
Elastic Load Balancing

Application Load Balancers



Elastic Load Balancing: Application Load Balancers

Copyright © 2021 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What is an Application Load Balancer?	1
Application Load Balancer components	1
Application Load Balancer overview	2
Benefits of migrating from a Classic Load Balancer	2
Related services	3
Pricing	3
Getting started	4
Before you begin	4
Step 1: Select a load balancer type	4
Step 2: Configure your load balancer and listener	5
Step 3: Configure a security group for your load balancer	5
Step 4: Configure your target group	5
Step 5: Register targets with your target group	6
Step 6: Create and test your load balancer	6
Step 7: Delete your load balancer (optional)	6
Tutorial: Create an Application Load Balancer using the AWS CLI	8
Before you begin	8
Create your load balancer	8
Add an HTTPS listener	9
Add path-based routing	9
Delete your load balancer	10
Load balancers	11
Subnets for your load balancer	11
Load balancer security groups	13
Load balancer state	13
Load balancer attributes	13
IP address type	14
Connection idle timeout	14
Deletion protection	15
Desync mitigation mode	16
Application Load Balancers and AWS WAF	17
Create a load balancer	17
Step 1: Configure a load balancer and a listener	18
Step 2: Configure security settings for an HTTPS listener	18
Step 3: Configure a security group	19
Step 4: Configure a target group	5
Step 5: Configure targets for the target group	19
Step 6: Create the load balancer	20
Update Availability Zones	20
Update security groups	21
Recommended rules	21
Update the associated security groups	22
Update the address type	22
Update tags	23
Delete a load balancer	24
Listeners	25
Listener configuration	25
Listener rules	26
Default rules	26
Rule priority	26
Rule actions	26
Rule conditions	26
Rule action types	26
Fixed-response actions	27

Forward actions	27
Redirect actions	29
Rule condition types	31
HTTP header conditions	31
HTTP request method conditions	32
Host conditions	32
Path conditions	33
Query string conditions	34
Source IP address conditions	34
Create an HTTP listener	35
Prerequisites	35
Add an HTTP listener	35
Create an HTTPS listener	36
SSL certificates	36
Security policies	38
Add an HTTPS listener	41
Update an HTTPS listener	41
Update listener rules	42
Requirements	42
Add a rule	42
Edit a rule	44
Reorder rules	45
Delete a rule	45
Update an HTTPS listener	46
Replace the default certificate	46
Add certificates to the certificate list	46
Remove certificates from the certificate list	47
Update the security policy	47
Authenticate users	48
Prepare to use an OIDC-compliant IdP	48
Prepare to use Amazon Cognito	48
Prepare to use Amazon CloudFront	49
Configure user authentication	49
Authentication flow	51
User claims encoding and signature verification	51
Authentication logout and session timeout	53
X-forwarded headers	54
X-Forwarded-For	54
X-Forwarded-Proto	54
X-Forwarded-Port	55
Delete a listener	55
Target groups	56
Routing configuration	56
Target type	57
Protocol version	57
Registered targets	58
Target group attributes	59
Routing algorithm	60
Deregistration delay	61
Slow start mode	61
Sticky sessions	62
Create a target group	64
Configure health checks	66
Health check settings	66
Target health status	67
Health check reason codes	68
Check the health of your targets	69

Modify the health check settings of a target group	70
Register targets	70
Target security groups	71
Register or deregister targets	71
Lambda functions as targets	74
Prepare the Lambda function	74
Create a target group for the Lambda function	73
Receive events from the load balancer	76
Respond to the load balancer	76
Multi-value headers	77
Enable health checks	79
Deregister the Lambda function	80
Update tags	81
Delete a target group	82
Monitor your load balancers	83
CloudWatch metrics	83
Application Load Balancer metrics	84
Metric dimensions for Application Load Balancers	94
Statistics for Application Load Balancer metrics	94
View CloudWatch metrics for your load balancer	95
Access logs	96
Access log files	97
Access log entries	98
Bucket permissions	107
Enable access logging	110
Disable access logging	111
Processing access log files	111
Request tracing	111
Syntax	112
Limitations	112
CloudTrail logs	113
Elastic Load Balancing information in CloudTrail	113
Understanding Elastic Load Balancing log file entries	114
Troubleshoot your load balancers	116
A registered target is not in service	116
Clients cannot connect to an internet-facing load balancer	117
The load balancer sends requests to unhealthy targets	117
The load balancer sends a response code of 000	117
The load balancer generates an HTTP error	117
HTTP 400: Bad request	118
HTTP 401: Unauthorized	118
HTTP 403: Forbidden	118
HTTP 405: Method not allowed	118
HTTP 408: Request timeout	118
HTTP 413: Payload too large	118
HTTP 414: URI too long	118
HTTP 460	119
HTTP 463	119
HTTP 464	119
HTTP 500: Internal server error	119
HTTP 501: Not implemented	119
HTTP 502: Bad gateway	119
HTTP 503: Service unavailable	120
HTTP 504: Gateway timeout	120
HTTP 505: Version not supported	120
HTTP 561: Unauthorized	120
A target generates an HTTP error	120

Quotas	121
Document history	122

What is an Application Load Balancer?

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

Elastic Load Balancing supports the following load balancers: Application Load Balancers, Network Load Balancers, Gateway Load Balancers, and Classic Load Balancers. You can select the type of load balancer that best suits your needs. This guide discusses 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](#).

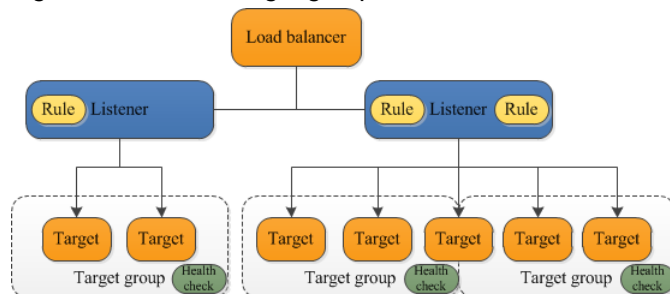
Application Load Balancer components

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 \(p. 11\)](#)
- [Listeners \(p. 25\)](#)

- [Target Groups \(p. 56\)](#)

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. Elastic Load Balancing scales your load balancer as traffic to your application changes over time. Elastic Load Balancing can scale to the vast majority of workloads automatically.

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

For more information, see [How Elastic Load Balancing works](#) in the *Elastic Load Balancing User Guide*.

Benefits of migrating from a Classic Load Balancer

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

- Support for [Path conditions \(p. 33\)](#). 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 \(p. 32\)](#). 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 \(p. 31\)](#) and methods, query parameters, and source IP addresses.
- Support for routing requests to multiple applications on a single EC2 instance. You can register each instance or IP address with the same target group using multiple ports.
- Support for 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.

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

Related services

Elastic Load Balancing 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 Elastic Load Balancing, instances that are launched by Auto Scaling are automatically registered with the load balancer, and instances that are terminated by Auto Scaling are automatically de-registered from the load balancer.
- **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](#) (p. 36).
- **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](#) (p. 83).
- **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.
- **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 [Application Load Balancers and AWS WAF](#) (p. 17).

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 [Elastic Load Balancing pricing](#).

Getting started with Application Load Balancers

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

Tasks

- [Before you begin](#) (p. 4)
- [Step 1: Select a load balancer type](#) (p. 4)
- [Step 2: Configure your load balancer and listener](#) (p. 5)
- [Step 3: Configure a security group for your load balancer](#) (p. 5)
- [Step 4: Configure your target group](#) (p. 5)
- [Step 5: Register targets with your target group](#) (p. 6)
- [Step 6: Create and test your load balancer](#) (p. 6)
- [Step 7: Delete your load balancer \(optional\)](#) (p. 6)

For demos of common load balancer configurations, see [Elastic Load Balancing demos](#).

Before you begin

- Decide which two Availability Zones you will use for your EC2 instances. Configure your virtual private cloud (VPC) with at least one public subnet in each of these Availability Zones. These public subnets are used to configure the load balancer. You can launch your EC2 instances in other subnets of these Availability Zones instead.
- Launch at least one EC2 instance in each Availability Zone. Be sure to install a web server, such as Apache or Internet Information Services (IIS), on each EC2 instance. Ensure that the security groups for these instances allow HTTP access on port 80.

Step 1: Select a load balancer type

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

To create an Application Load Balancer

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

Step 2: Configure your load balancer and listener

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

To configure your load balancer and listener

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

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

Step 3: Configure a security group for your load balancer

The security group for your load balancer must allow it to communicate with registered targets on both the listener port and the health check port. The console can create a security group for your load balancer on your behalf, with rules that specify the correct protocols and ports. If you prefer, you can create and select your own security group instead. For more information, see [Recommended rules \(p. 21\)](#).

On the **Configure Security Groups** page, complete the following procedure to have Elastic Load Balancing create a security group for your load balancer on your behalf.

To configure a security group for your load balancer

1. Choose **Create a new security group**.
2. Type a name and description for the security group, or keep the default name and description. This new security group contains a rule that allows traffic to the load balancer listener port that you selected on the **Configure Load Balancer** page.
3. Choose **Next: Configure Routing**.

Step 4: Configure your target group

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

To configure your target group

1. For **Target group**, keep the default, **New target group**.

2. For **Name**, enter a name for the new target group.
3. Keep the default target type (**Instance**), protocol (**HTTP**), and port (**80**).
4. For **Health checks**, keep the default settings.
5. Choose **Next: Register Targets**.

Step 5: Register targets with your target group

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

To register your instances with the target group

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

Step 6: Create and test your load balancer

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

To create and test your load balancer

1. On the **Review** page, choose **Create**.
2. After you are notified that your load balancer was created successfully, choose **Close**.
3. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
4. Select the newly created target group.
5. On the **Targets** tab, verify that your instances are ready. If the status of an instance is `initial`, it's probably because the instance is still in the process of being registered, or it has not passed the minimum number of health checks to be considered healthy. After the status of at least one instance is `healthy`, you can test your load balancer.
6. On the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
7. Select the newly created load balancer.
8. On the **Description** tab, copy the DNS name of the load balancer (for example, `my-load-balancer-1234567890.us-west-2.elb.amazonaws.com`). Paste the DNS name into the address field of an Internet-connected web browser. If everything is working, the browser displays the default page of your server.
9. (Optional) To define additional listener rules, see [Add a rule \(p. 42\)](#).

Step 7: Delete your load balancer (optional)

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

To delete your load balancer

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

2. On the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
3. Select the checkbox for the load balancer, and then choose **Actions, Delete**.
4. When prompted for confirmation, choose **Yes, Delete**.

Tutorial: Create an Application Load Balancer using the AWS CLI

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

Before you begin

- Use the following command to verify that you are running a version of the AWS CLI that supports Application Load Balancers.

```
aws elbv2 help
```

If you get an error message that elbv2 is not a valid choice, update your AWS CLI. For more information, see [Installing the AWS Command Line Interface](#) in the *AWS Command Line Interface User Guide*.

- Launch your EC2 instances in a virtual private cloud (VPC). Ensure that the security groups for these instances allow access on the listener port and the health check port. For more information, see [Target security groups](#) (p. 71).

Create your load balancer

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

To create a load balancer

1. Use the [create-load-balancer](#) command to create a load balancer. You must specify two subnets that are not from the same Availability Zone.

```
aws elbv2 create-load-balancer --name my-load-balancer \
--subnets subnet-0e3f5cac72EXAMPLE subnet-081ec835f3EXAMPLE --security-groups
sg-07e8ffd50fEXAMPLE
```

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

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

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

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

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

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

3. Use the [register-targets](#) command to register your instances with your target group:

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

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

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

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

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

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

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

Add an HTTPS listener

If you have a load balancer with an HTTP listener, you can add an HTTPS listener as follows.

To add an HTTPS listener to your load balancer

1. Create an SSL certificate for use with your load balancer using one of the following methods:
 - Create or import the certificate using AWS Certificate Manager (ACM). For more information, see [Request a certificate](#) or [Importing certificates](#) in the *AWS Certificate Manager User Guide*.
 - Upload the certificate using AWS Identity and Access Management (IAM). For more information, see [Working with server certificates](#) in the *IAM User Guide*.
2. Use the [create-listener](#) command to create the listener with a default rule that forwards requests to your target group. You must specify an SSL certificate when you create an HTTPS listener. Note that you can specify an SSL policy other than the default using the `--ssl-policy` option.

```
aws elbv2 create-listener --load-balancer-arn loadbalancer-arn \  
--protocol HTTPS --port 443 \  
--certificates CertificateArn=certificate-arn \  
--default-actions Type=forward,TargetGroupArn=targetgroup-arn
```

Add path-based routing

If you have a listener with a default rule that forwards requests to one target group, you can add a rule that forwards requests to another target group based on URL. For example, you can route general requests to one target group and requests to display images to another target group.

To add a rule to a listener with a path pattern

1. Use the `create-target-group` command to create a target group:

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

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

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

3. Use the `create-rule` command to add a rule to your listener that forwards requests to the target group if the URL contains the specified pattern:

```
aws elbv2 create-rule --listener-arn listener-arn --priority 10 \  
--conditions Field=path-pattern,Values='/img/*' \  
--actions Type=forward,TargetGroupArn=targetgroup-arn
```

Delete your load balancer

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

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


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](#) (p. 56), and then register targets with your target groups. You also create [listeners](#) (p. 25) 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 Elastic Load Balancing works](#) in the *Elastic Load Balancing User Guide*.

Contents

- [Subnets for your load balancer](#) (p. 11)
- [Load balancer security groups](#) (p. 13)
- [Load balancer state](#) (p. 13)
- [Load balancer attributes](#) (p. 13)
- [IP address type](#) (p. 14)
- [Connection idle timeout](#) (p. 14)
- [Deletion protection](#) (p. 15)
- [Desync mitigation mode](#) (p. 16)
- [Application Load Balancers and AWS WAF](#) (p. 17)
- [Create an Application Load Balancer](#) (p. 17)
- [Availability Zones for your Application Load Balancer](#) (p. 20)
- [Security groups for your Application Load Balancer](#) (p. 21)
- [IP address types for your Application Load Balancer](#) (p. 22)
- [Tags for your Application Load Balancer](#) (p. 23)
- [Delete an Application Load Balancer](#) (p. 24)

Subnets for your load balancer

When you create an Application Load Balancer, you must specify one of the following types of subnets: Availability Zone, Local Zone, or Outpost.

Availability Zones

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 8 free IP addresses. Your load balancer uses these IP addresses to establish connections with the targets.

Local Zones

You can specify a one or more Local Zone subnets. The following restrictions apply:

- You cannot use AWS WAF with the load balancer.
- You cannot use a Lambda function as a target.

Outposts

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 instances on the Outpost for the load balancer nodes. The supported instances are shown in the table below. Initially, the instances are `large` instances. The load balancer scales as needed, from `large` to `xlarge`, `xlarge` to `2xlarge`, and `2xlarge` to `4xlarge`. If you need additional capacity, the load balancer adds `4xlarge` instances. 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 Personal 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 available: Lambda functions as targets, AWS WAF integration, sticky sessions, authentication support, and integration with AWS Global Accelerator.

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		
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 \(p. 21\)](#).

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

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 [Bucket permissions \(p. 107\)](#).

`access_logs.s3.prefix`

The prefix for the location in the Amazon S3 bucket.

`deletion_protection.enabled`

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

`idle_timeout.timeout_seconds`

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

`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`. Elastic Load Balancing requires that message header names contain only alphanumeric characters and hyphens.

`routing.http2.enabled`

Indicates whether HTTP/2 is enabled. The default is `true`.

`waf.fail_open.enabled`

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

IP address type

You can set the types of IP addresses that clients can use with your internet-facing load balancer. Clients must use IPv4 addresses with internal load balancers.

The following are the IP address types:

`ipv4`

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

`dualstack`

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

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

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

For more information, see [IP address types for your Application Load Balancer \(p. 22\)](#).

Connection idle timeout

For each request that a client makes through a load balancer, the load balancer maintains two connections. The front-end connection is between a client and the load balancer. The back-end connection is between the load balancer and a target. The load balancer has a configured idle timeout period that applies to its connections. If no data has been sent or received by the time that the idle timeout period elapses, 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.

For back-end connections, we recommend that you enable the HTTP keep-alive option for your EC2 instances. You can enable HTTP keep-alive in the web server settings for your EC2 instances. If you enable HTTP keep-alive, the load balancer can reuse back-end connections until the keep-alive timeout expires. We also recommend that you configure the idle timeout of your application to be larger than the idle timeout configured for the load balancer.

By default, Elastic Load Balancing sets the idle timeout value for your load balancer to 60 seconds. Use the following procedure to set a different idle timeout value.

To update the idle timeout value using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
3. Select the load balancer.
4. On the **Description** tab, choose **Edit attributes**.
5. On the **Edit load balancer attributes** page, enter a value for **Idle timeout**, in seconds. The valid range is 1-4000.
6. Choose **Save**.

To update the idle timeout value using the AWS CLI

Use the [modify-load-balancer-attributes](#) command with the `idle_timeout.timeout_seconds` attribute.

Deletion protection

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

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

To enable deletion protection using the console

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

To disable deletion protection using the console

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

To enable or disable deletion protection using the AWS CLI

Use the [modify-load-balancer-attributes](#) command with the `deletion_protection.enabled` attribute.

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. For more information, see [Classification reasons \(p. 101\)](#).

- **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 \(p. 84\)](#).

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.

To update desync mitigation mode using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
3. Select the load balancer.
4. On the **Description** tab, choose **Edit attributes**.

5. For **Desync mitigation mode**, choose **Monitor**, **Defensive**, or **Strictest**.
6. Choose **Save**.

To update desync mitigation mode using the AWS CLI

Use the [modify-load-balancer-attributes](#) command with the `routing.http.desync_mitigation_mode` attribute set to `monitor`, `defensive`, or `strictest`.

Application Load Balancers and 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 [Working with web ACLs](#) in the *AWS WAF Developer Guide*.

To check whether your load balancer integrates with AWS WAF, select your load balancer in the AWS Management Console and choose the **Integrated services** tab.

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 the WAF fail open attribute.

To enable WAF fail open using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
3. Select the load balancer.
4. On the **Description** tab, choose **Edit attributes**.
5. For **WAF fail open**, choose **Enable**.
6. Choose **Save**.

To enable WAF fail open using the AWS CLI

Use the [modify-load-balancer-attributes](#) command with the `waf.fail_open.enabled` attribute set to `true`.

Create an Application Load Balancer

A load balancer takes requests from clients and distributes them across targets in a target group.

Before you begin, ensure that you have a virtual private cloud (VPC) with at least one public subnet in each of the Availability Zones used by your targets.

To create a load balancer using the AWS CLI, see [Tutorial: Create an Application Load Balancer using the AWS CLI](#) (p. 8).

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

Tasks

- [Step 1: Configure a load balancer and a listener](#) (p. 18)
- [Step 2: Configure security settings for an HTTPS listener](#) (p. 18)
- [Step 3: Configure a security group](#) (p. 19)
- [Step 4: Configure a target group](#) (p. 5)

- [Step 5: Configure targets for the target group \(p. 19\)](#)
- [Step 6: Create the load balancer \(p. 20\)](#)

Step 1: Configure a load balancer and a listener

First, provide some basic configuration information for your load balancer, such as a name, a network, and one or more listeners. A listener is a process that checks for connection requests. It is configured with a protocol and a port for connections from clients to the load balancer. For more information about supported protocols and ports, see [Listener configuration \(p. 25\)](#).

To configure your load balancer and listener

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
3. Choose **Create Load Balancer**.
4. For **Application Load Balancer**, choose **Create**.
5. For **Name**, enter a name for your load balancer. For example, **my-alb**.
6. For **Scheme**, an internet-facing load balancer routes requests from clients over the internet to targets. An internal load balancer routes requests to targets using private IP addresses.
7. For **IP address type**, choose **ipv4** if your clients use IPv4 addresses to communicate with the load balancer, or choose **dualstack** if your clients use both IPv4 and IPv6 addresses to communicate with the load balancer. If the load balancer is an internal load balancer, you must choose **ipv4**.
8. For **Listeners**, the default is a listener that accepts HTTP traffic on port 80. You can keep the default listener settings, modify the protocol, or modify the port. Choose **Add** to add another listener (for example, an HTTPS listener).
9. Select one subnet per zone to enable. If you enabled dual-stack mode for the load balancer, select subnets with associated IPv6 CIDR blocks. You can specify one of the following:
 - Subnets from at least two Availability Zones
 - Subnets from one or more Local Zones
 - One Outpost subnet
10. (Optional) You can use **Add-on services, AWS Global Accelerator** to create an accelerator and associate the load balancer with the accelerator.
11. (Optional) For **Tags**, specify the key and value for each tag to add to your load balancer.
12. Choose **Next: Configure Security Settings**.

Step 2: Configure security settings for an HTTPS listener

If you created an HTTPS listener in the previous step, configure the required security settings. Otherwise, go to the next page in the wizard.

When you use HTTPS for your load balancer listener, you must deploy an SSL certificate on your load balancer. The load balancer uses this certificate to terminate the connection and decrypt requests from clients before sending them to the targets. For more information, see [SSL certificates \(p. 36\)](#). You must also specify the security policy that the load balancer uses to negotiate SSL connections with the clients. For more information, see [Security policies \(p. 38\)](#).

To configure a certificate and security policy

1. For **Select default certificate**, do one of the following:

- If you created or imported a certificate using AWS Certificate Manager, select **Choose a certificate from ACM**, and then select the certificate from **Certificate name**.
 - If you uploaded a certificate using IAM, select **Choose a certificate from IAM**, and then select the certificate from **Certificate name**.
2. For **Security policy**, we recommend that you keep the default security policy.
 3. Choose **Next: Configure Security Groups**.

Step 3: Configure a security group

The security group for your load balancer must allow it to communicate with registered targets on both the listener port and the health check port. The console can create a security group for your load balancer on your behalf with rules that allow this communication. If you prefer, you can create a security group and select it instead. For more information, see [Recommended rules \(p. 21\)](#).

To configure a security group for your load balancer

1. Choose **Create a new security group**.
2. Enter a name and description for the security group, or keep the default name and description. This new security group contains a rule that allows traffic to the port that you selected for your load balancer on the **Configure Load Balancer** page.
3. Choose **Next: Configure Routing**.

Step 4: Configure a target group

You register targets with a target group. The target group that you configure in this step is used as the target group in the default listener rule, which forwards requests to the target group. For more information, see [Target groups for your Application Load Balancers \(p. 56\)](#).

To configure your target group

1. For **Target group**, keep the default, **New target group**.
2. For **Name**, enter a name for the target group.
3. For **Target type**, select **Instance** to register targets by instance ID, **IP** to register IP addresses, and **Lambda function** to register a Lambda function.
4. (Optional) If the target type is **Instance** or **IP**, modify the port and protocol as needed.
5. (Optional) If the target type is **Lambda function**, enable health checks as needed.
6. For **Health checks**, keep the default health check settings.
7. Choose **Next: Register Targets**.

Step 5: Configure targets for the target group

With an Application Load Balancer, the target type of your target group determines how you register targets with the target group.

To register targets by instance ID

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

To register IP addresses

1. For each IP address to register, do the following:
 - a. For **Network**, if the IP address is from a subnet of the target group VPC, select the VPC. Otherwise, select **Other private IP address**.
 - b. For **IP**, enter the IP address.
 - c. For **Port**, enter the port.
 - d. Choose **Add to list**.
2. When you have finished adding IP addresses to the list, choose **Next: Review**.

To register a Lambda function

1. For **Lambda function**, do one of the following:
 - Select the Lambda function
 - Create a new Lambda function and select it
 - Register the Lambda function after you create the target group
2. Choose **Next: Review**.

Step 6: Create the load balancer

After creating your load balancer, you can verify that your targets have passed the initial health check and then test that the load balancer is sending traffic to your targets. When you are finished with your load balancer, you can delete it. For more information, see [Delete an Application Load Balancer \(p. 24\)](#).

To create the load balancer

1. On the **Review** page, choose **Create**.
2. After the load balancer is created, choose **Close**.
3. (Optional) To define additional listener rules that forward requests based on a path pattern or a hostname, see [Add a rule \(p. 42\)](#).

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. 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.

To update Availability Zones using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
3. Select the load balancer.
4. On the **Description** tab, under **Basic Configuration**, choose **Edit Availability Zones**.

5. To enable a zone, select the check box for that zone and select one subnet. If there is only one subnet for that zone, it is selected. If there is more than one subnet for that zone, select one of the subnets.
6. To change the subnet for an enabled Availability Zone, choose **Change subnet** and select one of the other subnets.
7. To remove an Availability Zone, clear the check box for that Availability Zone.
8. Choose **Save**.

To update Availability Zones using the AWS CLI

Use the [set-subnets](#) command.

Security groups for your Application 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 do not, you can edit the rules for the currently associated security groups or associate different security groups with the load balancer. In a VPC, you provide the security group for your load balancer, which enables you to 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).

Recommended rules

The following rules are recommended for an internet-facing load balancer.

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

The following rules are recommended for an internal load balancer.

Inbound		
Source	Port Range	Comment

<i>VPC CIDR</i>	<i>listener</i>	Allow inbound traffic from the VPC CIDR on the load balancer listener port
Outbound		
Destination	Port Range	Comment
<i>instance security group</i>	<i>instance listener</i>	Allow outbound traffic to instances on the instance listener port
<i>instance security group</i>	<i>health check</i>	Allow outbound traffic to instances 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 for Linux Instances*.

Update the associated security groups

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

To update security groups using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
3. Select the load balancer.
4. On the **Description** tab, under **Security**, choose **Edit security groups**.
5. To associate a security group with your load balancer, select it. To remove a security group from your load balancer, clear it.
6. Choose **Save**.

To update security groups using the AWS CLI

Use the [set-security-groups](#) command.

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. The load balancer communicates with targets using IPv4 addresses, regardless of how the client communicates with the load balancer. For more information, see [IP address type](#) (p. 14).

IPv6 requirements

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

- The 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.

To set the IP address type at creation

Configure settings as described in [Create a Load Balancer \(p. 18\)](#).

To update the IP address type using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
3. Select the load balancer.
4. Choose **Actions, Edit IP address type**.
5. For **IP address type**, choose **ipv4** to support IPv4 addresses only or **dualstack** to support both IPv4 and IPv6 addresses.
6. Choose **Save**.

To update the IP address type using the AWS CLI

Use the [set-ip-address-type](#) command.

Tags for your 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. Tag keys must be unique for each load balancer. If you add a tag with a key that is already associated with the load balancer, it updates the value of that tag.

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

Restrictions

- Maximum number of tags per resource—50
- Maximum key length—127 Unicode characters
- Maximum value length—255 Unicode characters
- Tag keys and values are case sensitive. Allowed characters are letters, spaces, and numbers representable in UTF-8, plus the following special characters: + - = . _ : / @. Do not use leading or trailing spaces.
- Do not use the `aws :` prefix in your tag names or values because it is reserved for AWS use. You can't edit or delete tag names or values with this prefix. Tags with this prefix do not count against your tags per resource limit.

To update the tags for a load balancer using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
3. Select the load balancer.
4. On the **Tags** tab, choose **Add/Edit Tags**, and then do one or more of the following:

- a. To update a tag, edit the values of **Key** and **Value**.
 - b. To add a new tag, choose **Create Tag** and then enter values for **Key** and **Value**.
 - c. To delete a tag, choose the delete icon (X) next to the tag.
5. When you have finished updating tags, choose **Save**.

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

Use the [add-tags](#) and [remove-tags](#) commands.

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](#) (p. 15).

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 a target group](#) (p. 82).

To delete a load balancer using the console

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

To delete a load balancer using the AWS CLI

Use the [delete-load-balancer](#) command.

Listeners for your Application Load Balancers

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

Contents

- [Listener configuration \(p. 25\)](#)
- [Listener rules \(p. 26\)](#)
- [Rule action types \(p. 26\)](#)
- [Rule condition types \(p. 31\)](#)
- [Create an HTTP listener for your Application Load Balancer \(p. 35\)](#)
- [Create an HTTPS listener for your Application Load Balancer \(p. 36\)](#)
- [Listener rules for your Application Load Balancer \(p. 42\)](#)
- [Update an HTTPS listener for your Application Load Balancer \(p. 46\)](#)
- [Authenticate users using an Application Load Balancer \(p. 48\)](#)
- [HTTP headers and Application Load Balancers \(p. 54\)](#)
- [Delete a listener for your Application Load Balancer \(p. 55\)](#)

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 \(p. 36\)](#).

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. For more information, see [How the WebSocket Protocol Works](#) in the *Amazon CloudFront Developer Guide*.

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. By default, the load balancer converts these to individual HTTP/1.1 requests and distributes them across the healthy targets in the target group. However, you can use the protocol version to send the request to the targets using HTTP/2. For more information, see [Protocol version \(p. 57\)](#). 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.

For more information, see [Request routing](#) in the *Elastic Load Balancing User Guide*.

Listener rules

Each listener has a default rule, and you can optionally define additional rules. Each rule consists 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 [Edit a rule \(p. 44\)](#).

Default rules

When you create a listener, you define actions for the default rule. Default rules can't have conditions. If the conditions for none of a listener's rules are met, then the action for the default rule is performed.

The following is an example of a default rule as shown in the console:

last	HTTP 80: default action <i>This rule cannot be moved or deleted</i>	IF ✓ Requests otherwise not routed	THEN Forward to my-targets
------	---	--	--

Rule priority

Each rule has a priority. 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 cannot change the priority of the default rule. For more information, see [Reorder rules \(p. 45\)](#).

Rule actions

Each rule action has a type, an order, and the information required to perform the action. For more information, see [Rule action types \(p. 26\)](#).

Rule conditions

Each rule condition has a type and configuration information. When the conditions for a rule are met, then its actions are performed. For more information, see [Rule condition types \(p. 31\)](#).

Rule action types

The following are the supported action types for a listener rule:

`authenticate-cognito`

[HTTPS listeners] Use Amazon Cognito to authenticate users. For more information, see [Authenticate users using an Application Load Balancer \(p. 48\)](#).

`authenticate-oidc`

[HTTPS listeners] Use an identity provider that is compliant with OpenID Connect (OIDC) to authenticate users.

`fixed-response`

Return a custom HTTP response. For more information, see [Fixed-response actions \(p. 27\)](#).

`forward`

Forward requests to the specified target groups. For more information, see [Forward actions \(p. 27\)](#).

`redirect`

Redirect requests from one URL to another. For more information, see [Redirect actions \(p. 29\)](#).

The action with the lowest order value is performed first. 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.

If the protocol version is gRPC or HTTPS, the only supported actions are `forward` actions.

Fixed-response actions

You can use `fixed-response` actions to drop client requests and return 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 \(p. 98\)](#). The count of successful `fixed-response` actions is reported in the `HTTP_Fixed_Response_Count` metric. For more information, see [Application Load Balancer metrics \(p. 84\)](#).

Example Example fixed response action for the AWS CLI

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

You can use `forward` actions to route requests to one or more 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.

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, Elastic Load Balancing generates a second cookie, AWSALBTGCORS, which includes the same information as the original stickiness cookie plus this SameSite attribute. Clients receive both cookies.

Example 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 Example forward action with two 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 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](#) (p. 62) 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

You can use `redirect` actions to redirect 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:

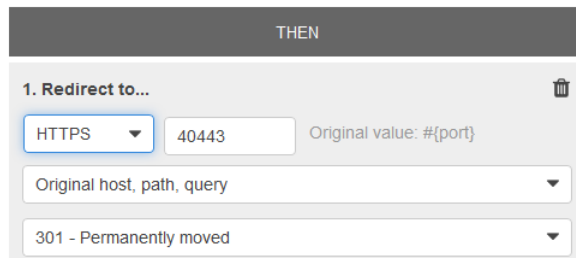
- `{protocol}` - Retains the protocol. Use in the protocol and query components.
- `{host}` - Retains the domain. Use in the hostname, path, and query components.
- `{port}` - Retains the port. Use in the port, path, and query components.

- `#{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 \(p. 98\)](#). The count of successful `redirect` actions is reported in the `HTTP_Redirect_Count` metric. For more information, see [Application Load Balancer metrics \(p. 84\)](#).

Example Example redirect actions using the console

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}"`.



THEN

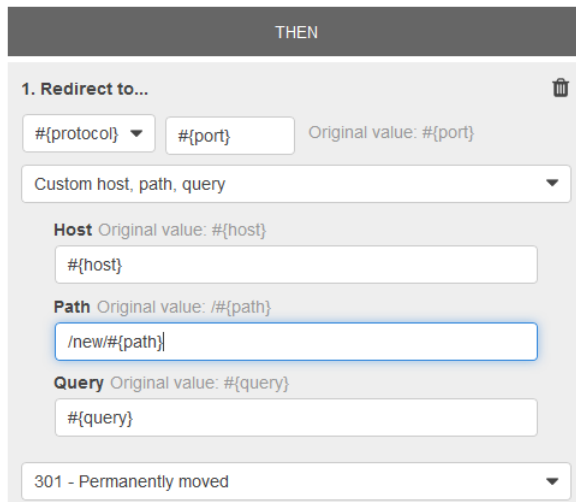
1. Redirect to...

Protocol: HTTPS Port: 40443 Original value: #{port}

Host, Path, Query: Original host, path, query

Status: 301 - Permanently moved

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}"`.



THEN

1. Redirect to...

Protocol: #{protocol} Port: #{port} Original value: #{port}

Host, Path, Query: Custom host, path, query

Host: #{host} Original value: #{host}

Path: /new/#{path} Original value: #{path}

Query: #{query} Original value: #{query}

Status: 301 - Permanently moved

Example Example redirect action for the AWS CLI

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.

```
[
  {
    "Type": "redirect",
    "RedirectConfig": {
      "Protocol": "HTTPS",
      "Port": "443",
      "Host": "#{host}",
      "Path": "/#{path}",
      "Query": "#{query}",
```

```
    "StatusCode": "HTTP_301"  
  }  
}  
]
```

Rule condition types

The following are the supported condition types for a rule:

host-header

Route based on the host name of each request. For more information, see [Host conditions \(p. 32\)](#).

http-header

Route based on the HTTP headers for each request. For more information, see [HTTP header conditions \(p. 31\)](#).

http-request-method

Route based on the HTTP request method of each request. For more information, see [HTTP request method conditions \(p. 32\)](#).

path-pattern

Route based on path patterns in the request URLs. For more information, see [Path conditions \(p. 33\)](#).

query-string

Route based on key/value pairs or values in the query strings. For more information, see [Query string conditions \(p. 34\)](#).

source-ip

Route based on the source IP address of each request. For more information, see [Source IP address conditions \(p. 34\)](#).

Each rule can optionally include up to one of each of the following conditions: `host-header`, `http-request-method`, `path-pattern`, and `source-ip`. Each rule can also optionally include one or more of each of the following conditions: `http-header` and `query-string`.

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.

For demos, see [Advanced request routing](#).

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.

Example Example HTTP header condition for the AWS CLI

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.

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

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 for the AWS CLI

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"]
    }
  }
]
```

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 host header condition for the AWS CLI

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 host header that matches the specified string.

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

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.

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 for the AWS CLI

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.

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

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 for the AWS CLI

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.

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 for the AWS CLI

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"]
    }
  }
]
```

Create an HTTP listener for your Application Load Balancer

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

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 \(p. 36\)](#).

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 \(p. 64\)](#).

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 \(p. 25\)](#).

To add an HTTP listener using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
 2. On the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
 3. Select a load balancer, and choose **Listeners**, **Add listener**.
 4. For **Protocol : port**, choose **HTTP** and keep the default port or enter a different port.
 5. For **Default actions**, do one of the following:
 - Choose **Add action**, **Forward to** and choose a target group.
 - Choose **Add action**, **Redirect to** and provide the URL for the redirect. For more information, see [Redirect actions \(p. 29\)](#).
 - Choose **Add action**, **Return fixed response** and provide a response code and optional response body. For more information, see [Fixed-response actions \(p. 27\)](#).
- To save the action, choose the checkmark icon.
6. Choose **Save**.
 7. (Optional) To define additional listener rules that forward requests based on a path pattern or a hostname, see [Add a rule \(p. 42\)](#).

To add an HTTP listener using the AWS CLI

Use the [create-listener](#) command to create the listener and default rule, and the [create-rule](#) command to define additional listener rules.

Create an HTTPS listener for your Application Load Balancer

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

You can create an HTTPS listener, which uses encrypted connections (also known as *SSL offload*). This feature enables traffic encryption between your load balancer and the clients that initiate SSL or TLS sessions.

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 \(p. 35\)](#).

Contents

- [SSL certificates \(p. 36\)](#)
 - [Default certificate \(p. 37\)](#)
 - [Certificate list \(p. 37\)](#)
 - [Certificate renewal \(p. 37\)](#)
- [Security policies \(p. 38\)](#)
 - [FS supported policies \(p. 38\)](#)
 - [TLS security policies \(p. 39\)](#)
- [Add an HTTPS listener \(p. 41\)](#)
- [Update an HTTPS listener \(p. 41\)](#)

SSL certificates

To use an HTTPS listener, you must deploy at least one SSL/TLS 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.

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.

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

Important

ACM supports RSA certificates with a 4096 key length and EC certificates. However, you cannot install these certificates on your load balancer through integration with ACM. You must upload these certificates to IAM in order to use them with your load balancer.

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 \(p. 46\)](#).

If you specify additional certificates in a [certificate list \(p. 37\)](#), 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, it has a default certificate and an empty certificate list. You can optionally add certificates to the certificate list for the listener. Using a certificate list enables the load balancer to support multiple domains on the same port and provide a different certificate for each domain. For more information, see [Add certificates to the certificate list \(p. 46\)](#).

The load balancer uses a smart certificate selection algorithm with support for SNI. If the hostname provided by a client matches a single certificate in the certificate list, the load balancer selects this certificate. If a hostname provided by a client matches multiple certificates in the certificate list, the load balancer selects the best certificate that the client can support. Certificate selection is based on the following criteria in the following order:

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

The load balancer access log entries indicate the hostname specified by the client and the certificate presented to the client. For more information, see [Access log entries \(p. 98\)](#).

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

Elastic Load Balancing 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.

Application Load Balancers do not support SSL renegotiation for client or target connections.

When you create a TLS listener, you must select a security policy. You can update the security policy as needed. For more information, see [Update the security policy \(p. 47\)](#).

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

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

- `ELBSecurityPolicy-2016-08` (default)
- `ELBSecurityPolicy-TLS-1-0-2015-04`
- `ELBSecurityPolicy-TLS-1-1-2017-01`
- `ELBSecurityPolicy-TLS-1-2-2017-01`
- `ELBSecurityPolicy-TLS-1-2-Ext-2018-06`
- `ELBSecurityPolicy-FS-2018-06`
- `ELBSecurityPolicy-FS-1-1-2019-08`
- `ELBSecurityPolicy-FS-1-2-2019-08`
- `ELBSecurityPolicy-FS-1-2-Res-2019-08`
- `ELBSecurityPolicy-2015-05` (identical to `ELBSecurityPolicy-2016-08`)
- `ELBSecurityPolicy-FS-1-2-Res-2020-10`

We recommend the `ELBSecurityPolicy-2016-08` policy for compatibility. You can use one of the `ELBSecurityPolicy-FS` policies if you require Forward Secrecy (FS). You can use one of the `ELBSecurityPolicy-TLS` policies to meet compliance and security standards that require disabling certain TLS protocol versions, or to support legacy clients that require deprecated ciphers. Only a small percentage of internet clients require TLS version 1.0. To view the TLS protocol version for requests to your load balancer, enable access logging for your load balancer and examine the access logs. For more information, see [Access Logs \(p. 96\)](#).

FS supported policies

The following table describes the default policy, `ELBSecurityPolicy-2016-08`, and the `ELBSecurityPolicy-FS` policies. The `ELBSecurityPolicy-` has been removed from policy names in the heading row so that they fit.

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

TLS security policies

The following table describes the default policy, `ELBSecurityPolicy-2016-08`, and the `ELBSecurityPolicy-TLS` policies. The `ELBSecurityPolicy-` has been removed from policy names in the heading row so that they fit.

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

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

To view the configuration of a security policy for Application Load Balancers using the AWS CLI, use the [describe-ssl-policies](#) command.

Add an HTTPS listener

You configure a listener with a protocol and a port for connections from clients to the load balancer, and a target group for the default listener rule. For more information, see [Listener configuration \(p. 25\)](#).

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 \(p. 64\)](#).
- To create an HTTPS listener, you must specify a certificate and a security policy. The load balancer uses the certificate to terminate the connection and decrypt requests from clients before routing them to targets. The load balancer uses the security policy when negotiating SSL connections with the clients.

To add an HTTPS listener using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
3. Select a load balancer, and choose **Listeners**, **Add listener**.
4. For **Protocol : port**, choose **HTTPS** and keep the default port or enter a different port.
5. (Optional) To authenticate users, for **Default actions**, choose **Add action**, **Authenticate** and provide the requested information. To save the action, choose the checkmark icon. For more information, see [Authenticate users using an Application Load Balancer \(p. 48\)](#).
6. For **Default actions**, do one of the following:
 - Choose **Add action**, **Forward to** and choose a target group.
 - Choose **Add action**, **Redirect to** and provide the URL for the redirect. For more information, see [Redirect actions \(p. 29\)](#).
 - Choose **Add action**, **Return fixed response** and provide a response code and optional response body. For more information, see [Fixed-response actions \(p. 27\)](#).

To save the action, choose the checkmark icon.

7. For **Security policy**, we recommend that you keep the default security policy.
8. For **Default SSL certificate**, do one of the following:
 - If you created or imported a certificate using AWS Certificate Manager, choose **From ACM** and choose the certificate.
 - If you uploaded a certificate using IAM, choose **From IAM** and choose the certificate.
9. Choose **Save**.
10. (Optional) To define additional listener rules that forward requests based on a path pattern or a hostname, see [Add a rule \(p. 42\)](#).
11. (Optional) To add a certificate list for use with the SNI protocol, see [Add certificates to the certificate list \(p. 46\)](#).

To add an HTTPS listener using the AWS CLI

Use the [create-listener](#) command to create the listener and default rule, and the [create-rule](#) command to define additional listener rules.

Update an HTTPS listener

After you create an HTTPS listener, you can replace the default certificate, update the certificate list, or replace the security policy. For more information, see [Update an HTTPS listener for your Application Load Balancer \(p. 46\)](#).

Listener rules for your Application Load Balancer

The rules that you define for your listener determine how the load balancer routes requests to the targets in one or more target groups.

Each rule consists of a priority, one or more actions, and one or more conditions. For more information, see [Listener rules \(p. 26\)](#).

Note

The console displays a relative sequence number for each rule, not the rule priority. You can get the priority of a rule by describing it using the AWS CLI or the Elastic Load Balancing API.

Requirements

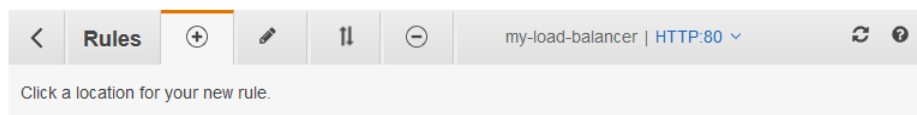
- Each rule must include exactly one of the following actions: `forward`, `redirect`, or `fixed-response`, and it must be the last action to be performed.
- Each rule can include zero or one of the following conditions: `host-header`, `http-request-method`, `path-pattern`, and `source-ip`, and zero or more of the following conditions: `http-header` and `query-string`.
- You can specify up to three comparison strings per condition and up to five per rule.
- 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 \(p. 64\)](#).

Add a rule

You define a default rule when you create a listener, and you can define additional nondefault rules at any time.

To add a rule using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
3. Select the load balancer and choose **Listeners**.
4. For the listener to update, choose **View/edit rules**.
5. Choose the **Add rules** icon (the plus sign) in the menu bar, which adds **Insert Rule** icons at the locations where you can insert a rule in the priority order.



6. Choose one of the **Insert Rule** icons added in the previous step.
7. Add one or more conditions as follows:
 - a. To add a host header condition, choose **Add condition, Host header** and enter the hostname (for example, `*.example.com`). To save the condition, choose the checkmark icon.

The maximum size of each string is 128 characters. The comparison is not case-sensitive. The following wildcard characters are supported: `*` and `?`.
 - b. To add a path condition, choose **Add condition, Path** and enter the path pattern (for example, `/img/*`). To save the condition, choose the checkmark icon.

The maximum size of each string is 128 characters. The comparison is case-sensitive. The following wildcard characters are supported: `*` and `?`.

- c. To add an HTTP header condition, choose **Add condition, Http header**. Enter the name of the header and add one or more comparison strings. To save the condition, choose the checkmark icon.

The maximum size of each header name is 40 characters, the header name is not case-sensitive, and wildcards are not supported. The maximum size of each comparison string is 128 characters and the following wildcard characters are supported: * and ?. The comparison is not case-sensitive.

- d. To add an HTTP request method condition, choose **Add condition, Http request method** and add one or more method names. To save the condition, choose the checkmark icon.

The maximum size of each name is 40 characters. The allowed characters are A-Z, hyphen (-), and underscore (_). The comparison is case sensitive. Wildcards are not supported.

- e. To add a query string condition, choose **Add condition, Query string** and add one or more key/value pairs. For each key/value pair, you can omit the key and specify only the value. To save the condition, choose the checkmark icon.

The maximum size of each string is 128 characters. The comparison is not case-sensitive. The following wildcard characters are supported: * and ?.

- f. To add a source IP condition, choose **Add condition, Source IP** and add one or more CIDR blocks. To save the condition, choose the checkmark icon.

You can use both IPv4 and IPv6 addresses. Wildcards are not supported.

8. (Optional, HTTPS listener) To authenticate users, choose **Add action, Authenticate** and provide the requested information. To save the action, choose the checkmark icon. For more information, see [Authenticate users using an Application Load Balancer \(p. 48\)](#).
9. Add one of the following actions:
- To add a forward action, choose **Add action, Forward to** and choose one or more target groups. If you use more than one target group, select a weight for each target group and optionally enable target group stickiness. If you enable target group stickiness and there is more than one target group, you must also enable sticky sessions on the target groups. To save the action, choose the checkmark icon. For more information, see [Forward actions \(p. 27\)](#).
 - To add a redirect action, choose **Add action, Redirect to** and provide the URL for the redirect. To save the action, choose the checkmark icon. For more information, see [Redirect actions \(p. 29\)](#).
 - To add a fixed-response action, choose **Add action, Return fixed response** and provide a response code and optional response body. To save the action, choose the checkmark icon. For more information, see [Fixed-response actions \(p. 27\)](#).

The screenshot shows the 'Insert Rule' configuration interface. It features two columns: 'IF (all match)' for conditions and 'THEN' for actions. In the 'IF' column, a condition is set for 'Host header...' with the value '*example.com'. In the 'THEN' column, the first action is '1. Forward to...' with 'my-targets' selected as the target group and a weight of 1. Below this, there is a 'Traffic distribution' section with a dropdown menu showing 'Select a target group' and a weight of 0. At the bottom of the 'THEN' section, there is a 'Group-level stickiness' section with a checkmark icon. The dialog also includes an 'Insert Rule' button at the top and an 'Add condition' button at the bottom of the 'IF' section, and an 'Add action' button at the bottom of the 'THEN' section.

10. Choose **Save**.
11. (Optional) To change the order of the rule, use the arrows and then choose **Save**. The default rule always has the **last** priority.
12. To leave this screen, choose the **Back to the load balancer** icon (the back button) in the menu bar.

To add a rule using the AWS CLI

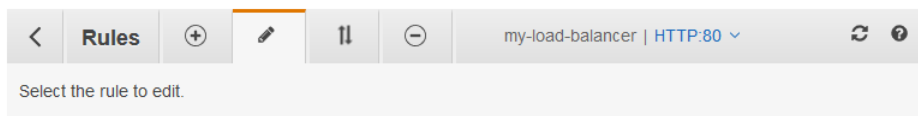
Use the [create-rule](#) command to create the rule. Use the [describe-rules](#) command to view information about the rule.

Edit a rule

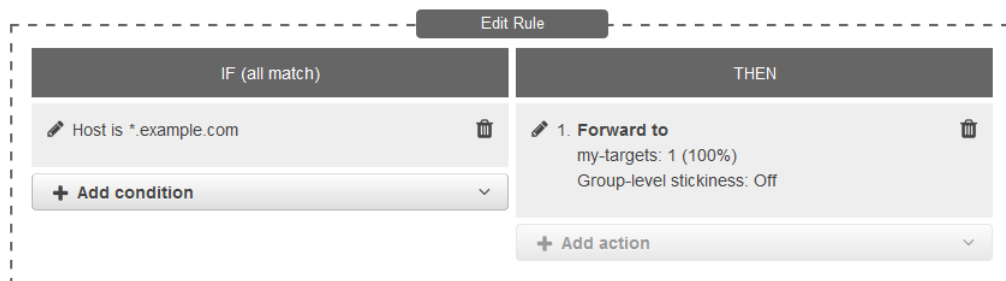
You can edit the action and conditions for a rule at any time. Rule updates do not take effect immediately, so requests could be routed using the previous rule configuration for a short time after you update a rule. Any in-flight requests are completed.

To edit a rule using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
3. Select the load balancer and choose **Listeners**.
4. For the listener to update, choose **View/edit rules**.
5. Choose the **Edit rules** icon (the pencil) in the menu bar.



6. For the rule to edit, choose the **Edit rules** icon (the pencil).
7. (Optional) Modify the conditions and actions as needed. For example, you can edit a condition or action (pencil icon), add a condition, add an authenticate action to a rule for an HTTPS listener, or delete a condition or action (trash can icon). You can't add conditions to the default rule.



8. Choose **Update**.
9. To leave this screen, choose the **Back to the load balancer** icon (the back button) in the menu bar.

To edit a rule using the AWS CLI

Use the [modify-rule](#) command.

Reorder rules

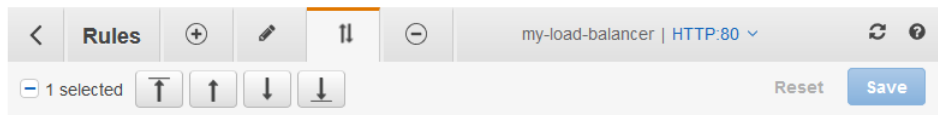
Rules are evaluated in priority order, from the lowest value to the highest value. The default rule is evaluated last. You can change the priority of a nondefault rule at any time. You cannot change the priority of the default rule.

Note

The console displays a relative sequence number for each rule, not the rule priority. When you reorder rules using the console, they get new rule priorities based on the existing rule priorities. To set the priority of a rule to a specific value, use the AWS CLI or the Elastic Load Balancing API.

To reorder rules using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
3. Select the load balancer and choose **Listeners**.
4. For the listener to update, choose **View/edit rules**.
5. Choose the **Reorder rules** icon (the arrows) in the menu bar.



6. Select the check box next to a rule, and then use the arrows to give the rule a new priority. The default rule always has the last priority.
7. When you have finished reordering rules, choose **Save**.
8. To leave this screen, choose the **Back to the load balancer** icon (the back button) in the menu bar.

To update rule priorities using the AWS CLI

Use the [set-rule-priorities](#) command.

Delete a rule

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

To delete a rule using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
3. Select the load balancer and choose **Listeners**.
4. For the listener to update, choose **View/edit rules**.
5. Choose the **Delete rules** icon (the minus sign) in the menu bar.
6. Select the check box for the rule and choose **Delete**. You can't delete the default rule for the listener.
7. To leave this screen, choose the **Back to the load balancer** icon (the back button) in the menu bar.

To delete a rule using the AWS CLI

Use the [delete-rule](#) command.

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.

Limitation

ACM supports RSA certificates with a 4096 key length and EC certificates. However, you cannot install these certificates on your load balancer through integration with ACM. You must upload these certificates to IAM in order to use them with your load balancer.

Tasks

- [Replace the default certificate \(p. 46\)](#)
- [Add certificates to the certificate list \(p. 46\)](#)
- [Remove certificates from the certificate list \(p. 47\)](#)
- [Update the security policy \(p. 47\)](#)

Replace the default certificate

You can replace the default certificate for your listener using the following procedure. For more information, see [SSL certificates \(p. 36\)](#).

To change the default certificate using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
3. Select the load balancer and choose **Listeners**.
4. Select the check box for the listener and choose **Edit**.
5. For **Default SSL certificate**, do one of the following:
 - If you created or imported a certificate using AWS Certificate Manager, choose **From ACM** and choose the certificate.
 - If you uploaded a certificate using IAM, choose **From IAM** and choose the certificate.
6. Choose **Update**.

To change the default certificate using the AWS CLI

Use the [modify-listener](#) command.

Add certificates to the certificate list

You can add certificates to the certificate list for your listener using the following procedure. When you first create an HTTPS listener, the certificate list is empty. You can add one or more certificates. You can optionally add the default certificate to ensure that this certificate is used with the SNI protocol even if it is replaced as the default certificate. For more information, see [SSL certificates \(p. 36\)](#).

To add certificates to the certificate list using the console

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

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

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

Use the `add-listener-certificates` command.

Remove certificates from the certificate list

You can remove certificates from the certificate list for an HTTPS listener using the following procedure. To remove the default certificate for an HTTPS listener, see [Replace the default certificate \(p. 46\)](#).

To remove certificates from the certificate list using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
3. Select the load balancer and choose **Listeners**.
4. For the listener to update, choose **View/edit certificates**, which displays the default certificate followed by any other certificates that you've added to the listener.
5. Choose the **Remove certificates** icon (the minus sign) in the menu bar.
6. Select the check boxes for the certificates and choose **Remove**.
7. To leave this screen, choose the **Back to the load balancer** icon (the back button) in the menu bar.

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

Use the `remove-listener-certificates` command.

Update the security policy

When you create 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 \(p. 38\)](#).

To update the security policy using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
3. Select the load balancer and choose **Listeners**.
4. Select the check box for the HTTPS listener and choose **Edit**.
5. For **Security policy**, choose a security policy.
6. Choose **Update**.

To update the security policy using the AWS CLI

Use the `modify-listener` command.

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 well-known social IdPs, such as Amazon, Facebook, or Google, through the user pools supported by Amazon Cognito.
- Authenticate users through corporate identities, using SAML, LDAP, or Microsoft AD, 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. 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 well-known config.
- 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

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 [Adding a Domain name for your user pool](#) 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 [Add social sign-in to a user pool](#) or [Add sign-in with a SAML IdP to a user pool](#) 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/oauth2/idpresponse`

To enable an IAM 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.

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.

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 [Authentication logout and session timeout \(p. 53\)](#).

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**—By default, after the authentication session cookie expires, the AJAX calls are redirected to the IdP and are blocked. If you use the `deny` option, the load balancer returns an HTTP 401 Unauthorized error to these AJAX calls.

The load balancer must be able to communicate with the IdP token endpoint (`TokenEndpoint`) and the IdP user info endpoint (`UserInfoEndpoint`). 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 the load balancer to access these endpoints.

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.

```
[{  
  "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": "123456789012345678901234567890",  
    "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 \(p. 26\)](#).

Authentication flow

Elastic Load Balancing uses the OIDC authorization code flow, which includes the following steps.

1. 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. 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.
2. After the user is authenticated, the IdP redirects the user back to the load balancer with an authorization grant code. The load balancer presents the code to the IdP token endpoint to get the ID token and access token.
3. After the load balancer validates the ID token, it exchanges the access token with the IdP user info endpoint to get the user claims.
4. The load balancer creates the authentication session cookie and sends it to the client so that the client's user agent can send the cookie to the load balancer when making requests. Because most browsers limit a cookie to 4K in size, 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.
5. The load balancer sends the user claims to the target in HTTP headers. For more information, see [User claims encoding and signature verification \(p. 51\)](#).
6. 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 [Authentication logout and session timeout \(p. 53\)](#).

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.

`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 authenticates the user and only passes access tokens and claims to the backend but does not pass the ID token information.

Applications that require the full user claims can use any standard JWT library to verify the JWT tokens. 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. The JWT signature is ECDSA + P-256 + SHA256.

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",
  ...
}
```

Because the load balancer does not encrypt the user claims, we recommend that you configure the target group to use HTTPS. If you configure your target group to use HTTP, be sure to restrict the traffic to your load balancer using security groups. We also recommend that you verify the signature before doing any authorization based on the claims. To get the public key, get the key ID from the JWT header and use it to look up the public key from the following regional endpoint:

```
https://public-keys.auth.elb.region.amazonaws.com/key-id
```

For AWS GovCloud (US-West), the endpoint is as follows:

```
https://s3-us-gov-west-1.amazonaws.com/aws-elb-public-keys-prod-us-gov-west-1/key-id
```

For AWS GovCloud (US-East), the endpoint is as follows:

```
https://s3-us-gov-east-1.amazonaws.com/aws-elb-public-keys-prod-us-gov-east-1/key-id
```

The following example shows how to get the public key in Python 3.x:

```
import jwt
import requests
import base64
import json

# Step 1: Get the key id from JWT headers (the kid field)
encoded_jwt = headers.dict['x-amzn-oidc-data']
jwt_headers = encoded_jwt.split('.')[0]
decoded_jwt_headers = base64.b64decode(jwt_headers)
decoded_jwt_headers = decoded_jwt_headers.decode("utf-8")
decoded_json = json.loads(decoded_jwt_headers)
kid = decoded_json['kid']

# Step 2: Get the public key from regional endpoint
url = 'https://public-keys.auth.elb.' + region + '.amazonaws.com/' + kid
req = requests.get(url)
pub_key = req.text

# Step 3: Get the payload
payload = jwt.decode(encoded_jwt, pub_key, algorithms=['ES256'])
```

The following example shows how to get the public key in Python 2.7:

```
import jwt
import requests
import base64
import json

# Step 1: Get the key id from JWT headers (the kid field)
encoded_jwt = headers.dict['x-amzn-oidc-data']
jwt_headers = encoded_jwt.split('.')[0]
decoded_jwt_headers = base64.b64decode(jwt_headers)
decoded_json = json.loads(decoded_jwt_headers)
kid = decoded_json['kid']

# Step 2: Get the public key from regional endpoint
url = 'https://public-keys.auth.elb.' + region + '.amazonaws.com/' + kid
req = requests.get(url)
pub_key = req.text

# Step 3: Get the payload
payload = jwt.decode(encoded_jwt, pub_key, algorithms=['ES256'])
```

Authentication logout and session timeout

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 a 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.

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 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 and then 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.

HTTP headers and Application Load Balancers

HTTP requests and HTTP responses use header fields to send information about the HTTP messages. 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 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](#) (p. 54)
- [X-Forwarded-Proto](#) (p. 54)
- [X-Forwarded-Port](#) (p. 55)

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 contain only the IP address of the load balancer. To see the IP address of the client, use the `X-Forwarded-For` request header. Elastic Load Balancing stores the IP address of the client in the `X-Forwarded-For` request header and passes the header to your server.

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
```

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. Elastic Load Balancing 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.

Delete a listener for your Application Load Balancer

You can delete a listener at any time. When you delete a load balancer, all its listeners are deleted.

To delete a listener using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
3. Select the load balancer and choose **Listeners**.
4. Select the check box for the HTTPS listener and choose **Delete**.
5. When prompted for confirmation, choose **Yes, Delete**.

To delete a listener using the AWS CLI

Use the [delete-listener](#) command.

Target groups for your Application Load Balancers

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. For more information, see [Application Load Balancer components \(p. 1\)](#).

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

Contents

- [Routing configuration \(p. 56\)](#)
- [Target type \(p. 57\)](#)
- [Protocol version \(p. 57\)](#)
- [Registered targets \(p. 58\)](#)
- [Target group attributes \(p. 59\)](#)
- [Routing algorithm \(p. 60\)](#)
- [Deregistration delay \(p. 61\)](#)
- [Slow start mode \(p. 61\)](#)
- [Sticky sessions \(p. 62\)](#)
- [Create a target group \(p. 64\)](#)
- [Health checks for your target groups \(p. 66\)](#)
- [Register targets with your target group \(p. 70\)](#)
- [Lambda functions as targets \(p. 74\)](#)
- [Tags for your target group \(p. 81\)](#)
- [Delete a target group \(p. 82\)](#)

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

If a target group is configured with the HTTPS protocol or uses HTTPS health checks, the TLS connections to the targets use the security settings from the `ELBSecurityPolicy-2016-08` policy.

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

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 cannot 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)

These supported CIDR blocks enable you to register the following with a target group: ClassicLink instances, 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), and on-premises resources linked to AWS through AWS Direct Connect or a Site-to-Site VPN connection.

Important

You can't specify publicly routable IP addresses.

If you specify targets using an instance ID, traffic is routed to instances using the primary private IP address specified in the primary network interface for the instance. 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 [Lambda functions as targets](#) (p. 74).

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 cannot 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 streaming from clients. The load balancer does not support streaming to the targets.

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 requests to a newly registered target as soon as the registration process completes and the target passes the initial health checks.

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 cannot 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.

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 Elastic Load Balancing to wait before deregistering a target. The range is 0–3600 seconds. The default value is 300 seconds.

`load_balancing.algorithm.type`

The load balancing algorithm determines how the load balancer selects targets when routing requests. The value is `round_robin` or `least_outstanding_requests`. The default is `round_robin`.

`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.

`stickiness.lb_cookie.duration_seconds`

The 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 value is `lb_cookie`.

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 \(p. 77\)](#).

Routing algorithm

By default, the round robin routing algorithm is used to route requests at the target group level. You can specify the least outstanding requests routing algorithm instead.

Consider using least outstanding requests when the requests for your application vary in complexity or your targets vary in processing capability. Round robin is a good choice when the requests and targets are similar, or if you need to distribute requests equally among targets. You can compare the effect of round robin versus least outstanding requests using the following CloudWatch metrics: **RequestCount**, **TargetConnectionErrorCount**, and **TargetResponseTime**.

Considerations

- You cannot enable both least outstanding requests and slow start mode.
- If you enable sticky sessions, this overrides the routing algorithm of the target group after the initial target selection.
- With HTTP/2, the load balancer converts the request to multiple HTTP/1.1 requests, so least outstanding request treats each HTTP/2 request as multiple requests.
- When you use least outstanding requests with WebSockets, the target is selected using least outstanding requests. The load balancer creates a connection to this target and sends all messages over this connection.

New console

To modify the routing algorithm using the new console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Choose the name of the target group to open its details page.
4. On the **Group details** tab, in the **Attributes** section, choose **Edit**.
5. On the **Edit attributes** page, for **Load balancing algorithm**, choose **Round robin** or **Least outstanding requests**.
6. Choose **Save changes**.

Old console

To modify the routing algorithm using the old console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Select the target group.
4. On the **Description** tab, choose **Edit attributes**.
5. On the **Edit attributes** page, for **Load balancing algorithm**, choose **Round robin** or **Least outstanding requests**, and then choose **Save**.

To modify the routing algorithm using the AWS CLI

Use the [modify-target-group-attributes](#) command with the `load_balancing.algorithm.type` attribute.

Deregistration delay

Elastic Load Balancing stops sending requests to targets that are deregistering. By default, Elastic Load Balancing 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 Elastic Load Balancing 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, Elastic Load Balancing 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 will be displayed as `draining` until the deregistration delay elapses.

If a deregistering target terminates the connection before the deregistration delay elapses, the client receives a 500-level error response.

New console

To update the deregistration delay value using the new console

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

Old console

To update the deregistration delay value using the old console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Select the target group.
4. On the **Description** tab, choose **Edit attributes**.
5. On the **Edit attributes** page, change the value of **Deregistration delay** as needed, and then choose **Save**.

To update the deregistration delay value using the AWS CLI

Use the `modify-target-group-attributes` command with the `deregistration_delay.timeout_seconds` attribute.

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 cannot enable both slow start mode and least outstanding requests.

New console

To update the slow start duration value using the new console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Choose the name of the target group to open its details page.
4. On the **Group details** tab, in the **Attributes** section, choose **Edit**.
5. On the **Edit attributes** page, change the value of **Slow start duration** as needed. To disable slow start mode, set the duration to 0.
6. Choose **Save changes**.

Old console

To update the slow start duration value using the old console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Select the target group.
4. On the **Description** tab, choose **Edit attributes**.
5. On the **Edit attributes** page, change the value of **Slow start duration** as needed, and then choose **Save**. To disable slow start mode, set the duration to 0.

To update the slow start duration value using the AWS CLI

Use the [modify-target-group-attributes](#) command with the `slow_start.duration_seconds` attribute.

Sticky sessions

Sticky sessions are a mechanism to route requests to the same target in a target group. This is useful for servers that maintain state information in order to provide a continuous experience to clients. To use sticky sessions, the clients must support cookies.

When a load balancer first receives a request from a client, it routes the request to a target, generates a cookie named `AWSALB` that encodes information about the selected target, 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 from a client that contains the cookie, if sticky sessions are enabled for the target group and the request goes to the same target group, the load balancer detects the cookie and routes the request to the same target. If the cookie is present but cannot 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.

You enable sticky sessions at the target group level. You can also set the duration for the stickiness of the load balancer-generated cookie, in seconds. The duration is set with each request. Therefore, if the client sends a request before each duration period expires, the sticky session continues.

With CORS (cross-origin resource sharing) requests, some browsers require `SameSite=None; Secure` to enable stickiness. In this case, Elastic Load Balancing generates a second stickiness cookie, `AWSALBCORS`, which includes the same information as the original stickiness cookie plus this `SameSite` attribute. Clients receive both cookies.

Application Load Balancers support load balancer-generated cookies only. The contents of these cookies are encrypted using a rotating key. You cannot decrypt or modify load balancer-generated cookies.

Considerations

- If you are using multiple layers of Application Load Balancers, you can enable sticky sessions on one layer only, because the load balancers would use the same cookie name.
- If you have a [forward action \(p. 27\)](#) with multiple target groups and one or more of the target groups has sticky sessions enabled, you must enable target group stickiness.
- WebSockets 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 header.
- Application Load Balancers do not support cookie values that are URL encoded.

New console

To enable sticky sessions using the new console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Choose the name of the target group to open its details page.
4. On the **Group details** tab, in the **Attributes** section, choose **Edit**.
5. On the **Edit attributes** page, do the following:
 - a. Select **Stickiness**.
 - b. For **Stickiness duration**, specify a value between 1 second and 7 days.
 - c. Choose **Save changes**.

Old console

To enable sticky sessions using the old console

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

2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Select the target group.
4. On the **Description** tab, choose **Edit attributes**.
5. On the **Edit attributes** page, do the following:
 - a. Select **Enable load balancer generated cookie stickiness**.
 - b. For **Stickiness duration**, specify a value between 1 second and 7 days.
 - c. Choose **Save**.

To enable sticky sessions using the AWS CLI

Use the [modify-target-group-attributes](#) command with the `stickiness.enabled` and `stickiness.lb_cookie.duration_seconds` attributes.

Create a target group

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](#) (p. 26).

You can add or remove targets from your target group at any time. For more information, see [Register targets with your target group](#) (p. 70). You can also modify the health check settings for your target group. For more information, see [Modify the health check settings of a target group](#) (p. 70).

New console

To create a target group using the new console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. 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. If the target type is **Instances** or **IP addresses**, do the following:
 - a. (Optional) For **Protocol** and **Port**, modify the default values as needed.
 - b. (Optional) For **Protocol version**, modify the default value as needed.
 - c. For **VPC**, select a virtual private cloud (VPC).
 - d. (Optional) In the **Health checks** section, modify the default settings as needed.
7. If the target type is **Lambda function**, you can enable health checks by selecting **Enable** in the **Health checks** section.
8. (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.
9. Choose **Next**.
10. (Optional) Add one or more targets as follows:
 - If the target type is **Instances**, select one or more instances, enter one or more ports, and then choose **Include as pending below**.
 - If the target type is **IP addresses**, select the network, enter the IP address and ports, and then choose **Include as pending below**.
 - If the target type is **Lambda function**, specify a single Lambda function or omit this step and specify a Lambda function later.
11. Choose **Create target group**.
12. (Optional) You can specify the target group in a listener rule. For more information, see [Listener Rules \(p. 42\)](#).

Old console

To create a target group using the old console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Choose **Create target group**.
4. For **Target group name**, type a name for the target group. This name must be unique per region per account, can have a maximum of 32 characters, must contain only alphanumeric characters or hyphens, and must not begin or end with a hyphen.
5. For **Target type**, select **Instance** to register targets by instance ID, **IP** to register IP addresses, and **Lambda function** to register a Lambda function.
6. If the target type is **Instance** or **IP**, do the following:
 - a. (Optional) For **Protocol** and **Port**, modify the default values as needed.
 - b. (Optional) For **Protocol version**, modify the default value as needed.
 - c. For **VPC**, select a virtual private cloud (VPC).
7. If the target type is **Lambda function**, do the following:
 - a. For **Lambda function**, do one of the following:
 - Select the Lambda function
 - Create a new Lambda function and select it
 - Register the Lambda function after you create the target group
 - b. (Optional) To enable health checks, choose **Health check, Enable**.
8. (Optional) For **Health check settings** and **Advanced health check settings**, modify the default settings as needed.
9. Choose **Create**.
10. (Optional) Add one or more tags as follows:
 - a. Select the newly created target group.
 - b. On the **Tags** tab, choose **Add/Edit Tags**.
 - c. On the **Add/Edit Tags** page, for each tag you add, choose **Create Tag** and then specify the tag key and tag value. When you have finished adding tags, choose **Save**.
11. (Optional) To add targets to the target group, see [Register targets with your target group \(p. 70\)](#).

12. (Optional) You can specify the target group in a listener rule. For more information, see [Listener Rules \(p. 42\)](#).

To create a target group using the AWS CLI

Use the [create-target-group](#) command to create the target group, the [add-tags](#) command to tag your target group, and the [register-targets](#) command to add targets.

Health checks for your target groups

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 nodes route requests across its unhealthy targets.

Health checks do not support WebSockets.

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 ping 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.

Setting	Description
HealthCheckProtocol	The protocol the load balancer uses when performing health checks on targets. The possible protocols are HTTP and HTTPS. The default is the HTTP protocol.
HealthCheckPort	The port the load balancer uses when performing health checks on targets. The default is to use the port on which each target receives traffic from the load balancer.
HealthCheckPath	The destination for health checks on the targets. If the protocol version is HTTP/1.1 or HTTP/2, specify a valid URI (<i>/path?query</i>). The default is <i>/</i> . If the protocol version is gRPC, specify the path of a custom health check method with the format

Setting	Description
	/Package.Class/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 <code>instance</code> or <code>ip</code> and 30 seconds if the target type is <code>lambda</code> .
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 <code>instance</code> or <code>ip</code> and 35 seconds if the target type is <code>lambda</code> .
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.
Matcher	<p>The codes to use when checking for a successful response from a target. These are called Success codes in the console.</p> <p>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.</p> <p>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.</p>

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
<code>initial</code>	<p>The load balancer is in the process of registering the target or performing the initial health checks on the target.</p> <p>Related reason codes: <code>Elb.RegistrationInProgress</code> <code>Elb.InitialHealthChecking</code></p>

Value	Description
healthy	The target is healthy. Related reason codes: None
unhealthy	The target did not respond to a health check or failed the health check. Related reason codes: <code>Target.ResponseCodeMismatch</code> <code>Target.Timeout</code> <code>Target.FailedHealthChecks</code> <code>Elb.InternalError</code>
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: <code>Target.NotRegistered</code> <code>Target.NotInUse</code> <code>Target.InvalidState</code> <code>Target.IpUnusable</code>
draining	The target is deregistering and connection draining is in process. Related reason code: <code>Target.DeregistrationInProgress</code>
unavailable	Health checks are disabled for the target group. Related reason code: <code>Target.HealthCheckDisabled</code>

Health check reason codes

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

Reason code	Description
<code>Elb.InitialHealthChecking</code>	Initial health checks in progress
<code>Elb.InternalError</code>	Health checks failed due to an internal error
<code>Elb.RegistrationInProgress</code>	Target registration is in progress
<code>Target.DeregistrationInProgress</code>	Target deregistration is in progress
<code>Target.FailedHealthChecks</code>	Health checks failed
<code>Target.HealthCheckDisabled</code>	Health checks are disabled
<code>Target.InvalidState</code>	Target is in the stopped state Target is in the terminated state Target is in the terminated or stopped state

Reason code	Description
	Target is in an invalid state
<code>Target.IpUnusable</code>	The IP address cannot be used as a target, as it is in use by a load balancer
<code>Target.NotInUse</code>	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
<code>Target.NotRegistered</code>	Target is not registered to the target group
<code>Target.ResponseCodeMismatch</code>	Health checks failed with these codes: <code>[code]</code>
<code>Target.Timeout</code>	Request timed out

Check the health of your targets

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

New console

To check the health of your targets using the new console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. 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, the **Status** column indicates the status of each target.
5. If the status is any value other than `Healthy`, the **Status details** column contains more information.

Old console

To check the health of your targets using the old console

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

To check the health of your targets using the AWS CLI

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

To receive email notifications about unhealthy targets

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

Modify the health check settings of a target group

You can modify the health check settings for your target group at any time.

New console

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

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Choose the name of the target group to open its details page.
4. On the **Group details** tab, in the **Health check settings** section, choose **Edit**.
5. On the **Edit health check settings** page, modify the settings as needed, and then choose **Save changes**.

Old console

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

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

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

Use the [modify-target-group](#) command.

Register targets with your 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](#) (p. 56).

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*.

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
<i>load balancer security group</i>	<i>instance listener</i>	Allow traffic from the load balancer on the instance listener port
<i>load balancer security group</i>	<i>health check</i>	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 for Linux Instances*.

Register or deregister targets

The target type of your target group determines how you register targets with that target group. For more information, see [Target type \(p. 57\)](#).

Contents

- [Register or deregister targets by instance ID \(p. 71\)](#)
- [Register or deregister targets by IP address \(p. 72\)](#)
- [Register or deregister a Lambda function \(p. 73\)](#)
- [Register or deregister targets using the AWS CLI \(p. 74\)](#)

Register or deregister targets by instance ID

The instance must be in the virtual private cloud (VPC) that you specified for the target group. The instance must also be in the `running` state when you register it.

New console

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

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Choose the name of the target group to open its details page.

4. Choose the **Targets** tab.
5. To register instances, choose **Register targets**. Select one or more instances, enter the default instance port as needed, and then choose **Include as pending below**. When you are finished adding instances, choose **Register pending targets**.
6. To deregister instances, select the instances and then choose **Deregister**.

Old console

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

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Select your target group.
4. On the **Targets** tab, choose **Edit**.
5. To register instances, select them from **Instances**, modify the default instance port as needed, and choose **Add to registered**.
6. To deregister instances, select them from **Registered instances** and choose **Remove**.
7. Choose **Save**.

Register or deregister targets by IP address

The IP addresses that you register must be from one of the following CIDR blocks:

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

Limits

- You cannot 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.

New console

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

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

Old console

To register or deregister targets by IP address

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

Register or deregister a Lambda function

You can register a single Lambda function with each target group. Elastic Load Balancing must have permissions to invoke 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, it is better to create a new target group instead. For more information, see [Lambda functions as targets \(p. 74\)](#).

New console

To register or deregister a Lambda function using the new console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Choose the name of the target group to open its details page.
4. Choose the **Targets** tab.
5. If there is no Lambda function registered, choose **Register**. Select the Lambda function and choose **Register**.
6. To deregister a Lambda function, choose **Deregister**. When prompted for confirmation, choose **Deregister**.

Old console

To register or deregister a Lambda function using the old console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Select your target group and choose the **Targets** tab.
4. If there is no Lambda function registered, choose **Register**. Select the Lambda function and choose **Register**.
5. To deregister a Lambda function, choose **Deregister**. When prompted for confirmation, choose **Deregister**.

Register or deregister targets using the AWS CLI

Use the [register-targets](#) command to add targets and the [deregister-targets](#) command to remove targets.

Lambda functions as targets

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.

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.

Contents

- [Prepare the Lambda function \(p. 74\)](#)
- [Create a target group for the Lambda function \(p. 73\)](#)
- [Receive events from the load balancer \(p. 76\)](#)
- [Respond to the load balancer \(p. 76\)](#)
- [Multi-value headers \(p. 77\)](#)
- [Enable health checks \(p. 79\)](#)
- [Deregister the Lambda function \(p. 80\)](#)

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 Elastic Load Balancing permission to invoke your Lambda function. We recommend that you include the `--source-arn` parameter to restrict function invocation to the specified target group.

```
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
```

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 versioning and aliases](#) and [Traffic shifting using 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 [Basic AWS Lambda function configuration](#). For information about the maximum timeout value you can configure, see [AWS Lambda limits](#).

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.

New console

To create a target group and register the Lambda function using the new console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Choose **Create target group**.
4. For **Choose a target type**, select **Lambda function**.
5. For **Target group name**, type a name for the target group.
6. (Optional) To enable health checks, choose **Enable** in the **Health checks** section.
7. (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.
8. Choose **Next**.
9. Specify a single Lambda function or omit this step and specify a Lambda function later.
10. Choose **Create target group**.

Old console

To create a target group and register the Lambda function using the old console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Choose **Create target group**.
4. For **Target group name**, type a name for the target group.
5. For **Target type**, select **Lambda function**.
6. For **Lambda function**, do one of the following:

- Select the Lambda function
 - Create a new Lambda function and select it
 - Register the Lambda function after you create the target group
7. (Optional) To enable health checks, choose **Health check, Enable**.
 8. Choose **Create**.

To create a target group and deregister the Lambda function using the AWS CLI

Use the [create-target-group](#) and [register-targets](#) commands.

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 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](#), create a function, and select one of the following from the AWS Serverless Application Repository:

- ALB-Lambda-Target-HelloWorld
- 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 \(p. 77\)](#)
- [Responses with multi-value headers \(p. 78\)](#)
- [Enable multi-value headers \(p. 79\)](#)

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"]  
  },  
}
```

```
}
```

Enable multi-value headers

You can enable or disable multi-value headers for a target group with the target type `lambda`.

New console

To enable multi-value headers using the new console

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

Old console

To enable multi-value headers using the old console

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

To enable multi-value headers using the AWS CLI

Use the [modify-target-group-attributes](#) command with the `lambda.multi_value_headers.enabled` attribute.

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
  }
}
```

New console

To enable health checks for a target group using the new console

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

Old console

To enable health checks for a target group using the old console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Select your target group.
4. On the **Health checks** tab, choose **Edit health check**.
5. For **Health check**, select **Enable**.
6. Choose **Save**.

To enable health checks for a target group using the AWS CLI

Use the `modify-target-group` command with the `--health-check-enabled` option.

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 instead of the existing one.

New console

To deregister the Lambda function using the new console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Choose the name of the target group to open its details page.
4. On the **Targets** tab, choose **Deregister**.

5. When prompted for confirmation, choose **Deregister**.

Old console

To deregister the Lambda function using the old console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Select your target group.
4. On the **Targets** tab, choose **Deregister**.
5. When prompted for confirmation, choose **Deregister**.

To deregister the Lambda function using the AWS CLI

Use the `deregister-targets` command.

Tags for your target group

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

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

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

Restrictions

- Maximum number of tags per resource—50
- Maximum key length—127 Unicode characters
- Maximum value length—255 Unicode characters
- Tag keys and values are case-sensitive. Allowed characters are letters, spaces, and numbers representable in UTF-8, plus the following special characters: + - = . _ : / @. Do not use leading or trailing spaces.
- Do not use the `aws:` prefix in your tag names or values because it is reserved for AWS use. You can't edit or delete tag names or values with this prefix. Tags with this prefix do not count against your tags per resource limit.

New console

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

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

Old console

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

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

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

Use the [add-tags](#) and [remove-tags](#) commands.

Delete a target group

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

New console

To delete a target group using the new console

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

Old console

To delete a target group using the old console

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

To delete a target group using the AWS CLI

Use the [delete-target-group](#) command.

Monitor your 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 \(p. 83\)](#).

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 \(p. 96\)](#).

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 \(p. 111\)](#).

CloudTrail logs

You can use AWS CloudTrail to capture detailed information about the calls made to the Elastic Load Balancing API and store them as log files in Amazon S3. You can use these CloudTrail logs to determine which calls were made, the source IP address where the call came from, who made the call, when the call was made, and so on. For more information, see [Logging API calls for your Application Load Balancer using AWS CloudTrail \(p. 113\)](#).

CloudWatch metrics for your Application Load Balancer

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

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

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

For more information, see the [Amazon CloudWatch User Guide](#).

Contents

- [Application Load Balancer metrics](#) (p. 84)
- [Metric dimensions for Application Load Balancers](#) (p. 94)
- [Statistics for Application Load Balancer metrics](#) (p. 94)
- [View CloudWatch metrics for your load balancer](#) (p. 95)

Application Load Balancer metrics

The AWS/ApplicationELB namespace includes the following metrics for load balancers.

Metric	Description
ActiveConnectionCount	<p>The total number of concurrent TCP connections active from clients to the load balancer and from the load balancer to targets.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: The most useful statistic is Sum.</p> <p>Dimensions</p> <ul style="list-style-type: none">• LoadBalancer
ClientTLSNegotiationErrorCount	<p>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.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: The most useful statistic is Sum.</p> <p>Dimensions</p> <ul style="list-style-type: none">• LoadBalancer• AvailabilityZone, LoadBalancer
ConsumedLCUs	<p>The number of load balancer capacity units (LCU) used by your load balancer. You pay for the number of LCUs that you use per hour. For more information, see Elastic Load Balancing pricing.</p> <p>Reporting criteria: Always reported</p> <p>Statistics: All</p> <p>Dimensions</p> <ul style="list-style-type: none">• LoadBalancer
DesyncMitigationMode_NonCompliantRequests_Count	<p>The number of requests that do not comply with RFC 7230.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: The most useful statistic is Sum.</p>

Metric	Description
	Dimensions <ul style="list-style-type: none"> LoadBalancer AvailabilityZone, LoadBalancer
DroppedInvalidHeaderRequests	<p>The number of requests where the load balancer removed HTTP headers with header fields that are not valid before routing the request. The load balancer removes these headers only if the <code>routing.http.drop_invalid_header_fields.enabled</code> attribute is set to true.</p> <p>Reporting criteria: Always reported</p> <p>Statistics: All</p> <p>Dimensions</p> <ul style="list-style-type: none"> AvailabilityZone, LoadBalancer
ForwardedInvalidHeaderRequests	<p>The number of requests routed by the load balancer that had HTTP headers with header fields that are not valid. The load balancer forwards requests with these headers only if the <code>routing.http.drop_invalid_header_fields.enabled</code> attribute is set to false.</p> <p>Reporting criteria: Always reported</p> <p>Statistics: All</p> <p>Dimensions</p> <ul style="list-style-type: none"> AvailabilityZone, LoadBalancer
GrpcRequestCount	<p>The number of gRPC requests processed over IPv4 and IPv6.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: The most useful statistic is Sum. Minimum, Maximum, and Average all return 1.</p> <p>Dimensions</p> <ul style="list-style-type: none"> LoadBalancer
HTTP_Fixed_Response_Count	<p>The number of fixed-response actions that were successful.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: The only meaningful statistic is Sum.</p> <p>Dimensions</p> <ul style="list-style-type: none"> LoadBalancer AvailabilityZone, LoadBalancer

Metric	Description
HTTP_Redirect_Count	<p>The number of redirect actions that were successful.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: The only meaningful statistic is Sum.</p> <p>Dimensions</p> <ul style="list-style-type: none"> LoadBalancer AvailabilityZone, LoadBalancer
HTTP_Redirect_Url_LimitExceeded_Count	<p>The number of redirect actions that couldn't be completed because the URL in the response location header is larger than 8K.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: The only meaningful statistic is Sum.</p> <p>Dimensions</p> <ul style="list-style-type: none"> LoadBalancer AvailabilityZone, LoadBalancer
HTTPCode_ELB_3XX_Count	<p>The number of HTTP 3XX redirection codes that originate from the load balancer.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: The only meaningful statistic is Sum.</p> <p>Dimensions</p> <ul style="list-style-type: none"> LoadBalancer AvailabilityZone, LoadBalancer
HTTPCode_ELB_4XX_Count	<p>The number of HTTP 4XX client error codes that originate from the load balancer. 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 (p. 119). This count does not include any response codes generated by the targets.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: The most useful statistic is Sum. Minimum, Maximum, and Average all return 1.</p> <p>Dimensions</p> <ul style="list-style-type: none"> LoadBalancer AvailabilityZone, LoadBalancer

Metric	Description
HTTPCode_ELB_5XX_Count	<p>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.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: The most useful statistic is Sum. Minimum, Maximum, and Average all return 1.</p> <p>Dimensions</p> <ul style="list-style-type: none"> LoadBalancer AvailabilityZone, LoadBalancer
HTTPCode_ELB_500_Count	<p>The number of HTTP 500 error codes that originate from the load balancer.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: The only meaningful statistic is Sum.</p> <p>Dimensions</p> <ul style="list-style-type: none"> LoadBalancer
HTTPCode_ELB_502_Count	<p>The number of HTTP 502 error codes that originate from the load balancer.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: The only meaningful statistic is Sum.</p> <p>Dimensions</p> <ul style="list-style-type: none"> LoadBalancer
HTTPCode_ELB_503_Count	<p>The number of HTTP 503 error codes that originate from the load balancer.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: The only meaningful statistic is Sum.</p> <p>Dimensions</p> <ul style="list-style-type: none"> LoadBalancer
HTTPCode_ELB_504_Count	<p>The number of HTTP 504 error codes that originate from the load balancer.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: The only meaningful statistic is Sum.</p> <p>Dimensions</p> <ul style="list-style-type: none"> LoadBalancer

Metric	Description
IPv6ProcessedBytes	<p>The total number of bytes processed by the load balancer over IPv6. This count is included in <code>ProcessedBytes</code>.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: The most useful statistic is <code>Sum</code>.</p> <p>Dimensions</p> <ul style="list-style-type: none"> • <code>LoadBalancer</code>
IPv6RequestCount	<p>The number of IPv6 requests received by the load balancer.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: The most useful statistic is <code>Sum</code>. <code>Minimum</code>, <code>Maximum</code>, and <code>Average</code> all return 1.</p> <p>Dimensions</p> <ul style="list-style-type: none"> • <code>LoadBalancer</code>
NewConnectionCount	<p>The total number of new TCP connections established from clients to the load balancer and from the load balancer to targets.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: The most useful statistic is <code>Sum</code>.</p> <p>Dimensions</p> <ul style="list-style-type: none"> • <code>LoadBalancer</code>
NonStickyRequestCount	<p>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.</p> <p>Reporting criteria: Stickiness is enabled on the target group.</p> <p>Statistics: The only meaningful statistic is <code>Sum</code>.</p> <p>Dimensions</p> <ul style="list-style-type: none"> • <code>LoadBalancer</code> • <code>AvailabilityZone</code>, <code>LoadBalancer</code>

Metric	Description
ProcessedBytes	<p>The total number of bytes processed by the load balancer over IPv4 and IPv6. This count includes traffic to and from clients and Lambda functions, and traffic from an Identity Provider (IdP) if user authentication is enabled.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: The most useful statistic is Sum.</p> <p>Dimensions</p> <ul style="list-style-type: none"> LoadBalancer
RejectedConnectionCount	<p>The number of connections that were rejected because the load balancer had reached its maximum number of connections.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: The most useful statistic is Sum.</p> <p>Dimensions</p> <ul style="list-style-type: none"> LoadBalancer AvailabilityZone, LoadBalancer
RequestCount	<p>The number of requests processed over IPv4 and IPv6.</p> <p>Reporting criteria: Always reported</p> <p>Statistics: The most useful statistic is Sum.</p> <p>Dimensions</p> <ul style="list-style-type: none"> LoadBalancer TargetGroup, LoadBalancer
RuleEvaluations	<p>The number of rules processed by the load balancer given a request rate averaged over an hour.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: The most useful statistic is Sum.</p> <p>Dimensions</p> <ul style="list-style-type: none"> LoadBalancer

The AWS/ApplicationELB namespace includes the following metrics for targets.

Metric	Description
HealthyHostCount	<p>The number of targets that are considered healthy.</p> <p>Reporting criteria: Reported if health checks are enabled</p>

Metric	Description
	<p>Statistics: The most useful statistics are Average, Minimum, and Maximum.</p> <p>Dimensions</p> <ul style="list-style-type: none"> • TargetGroup, LoadBalancer • TargetGroup, AvailabilityZone, LoadBalancer
HTTPCode_Target_2XX_Count, HTTPCode_Target_3XX_Count, HTTPCode_Target_4XX_Count, HTTPCode_Target_5XX_Count	<p>The number of HTTP response codes generated by the targets. This does not include any response codes generated by the load balancer.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: The most useful statistic is Sum. Minimum, Maximum, and Average all return 1.</p> <p>Dimensions</p> <ul style="list-style-type: none"> • LoadBalancer • AvailabilityZone, LoadBalancer • TargetGroup, LoadBalancer • TargetGroup, AvailabilityZone, LoadBalancer
RequestCountPerTarget	<p>The average number of requests received by each 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.</p> <p>Reporting criteria: Always reported</p> <p>Statistics: The only valid statistic is Sum. This represents the average not the sum.</p> <p>Dimensions</p> <ul style="list-style-type: none"> • TargetGroup • TargetGroup, LoadBalancer
TargetConnectionErrorCount	<p>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.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: The most useful statistic is Sum.</p> <p>Dimensions</p> <ul style="list-style-type: none"> • LoadBalancer • AvailabilityZone, LoadBalancer • TargetGroup, LoadBalancer • TargetGroup, AvailabilityZone, LoadBalancer

Metric	Description
TargetResponseTime	<p>The time elapsed, in seconds, after the request leaves the load balancer until a response from the target is received. This is equivalent to the <code>target_processing_time</code> field in the access logs.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: The most useful statistics are Average and pNN.NN (percentiles).</p> <p>Dimensions</p> <ul style="list-style-type: none"> • LoadBalancer • AvailabilityZone, LoadBalancer • TargetGroup, LoadBalancer • TargetGroup, AvailabilityZone, LoadBalancer
TargetTLSNegotiationErrorCount	<p>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.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: The most useful statistic is Sum.</p> <p>Dimensions</p> <ul style="list-style-type: none"> • LoadBalancer • AvailabilityZone, LoadBalancer • TargetGroup, LoadBalancer • TargetGroup, AvailabilityZone, LoadBalancer
UnHealthyHostCount	<p>The number of targets that are considered unhealthy.</p> <p>Reporting criteria: Reported if health checks are enabled</p> <p>Statistics: The most useful statistics are Average, Minimum, and Maximum.</p> <p>Dimensions</p> <ul style="list-style-type: none"> • TargetGroup, LoadBalancer • TargetGroup, AvailabilityZone, LoadBalancer

The AWS/ApplicationELB namespace includes the following metrics for Lambda functions that are registered as targets.

Metric	Description
LambdaInternalError	<p>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 <code>error_reason</code> field of the access log.</p> <p>Reporting criteria: There is a nonzero value</p>

Metric	Description
	<p>Statistics: The only meaningful statistic is Sum.</p> <p>Dimensions</p> <ul style="list-style-type: none"> TargetGroup TargetGroup, LoadBalancer
LambdaTargetProcessedBytes	<p>The total number of bytes processed by the load balancer for requests to and responses from a Lambda function.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: The only meaningful statistic is Sum.</p> <p>Dimensions</p> <ul style="list-style-type: none"> LoadBalancer
LambdaUserError	<p>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.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: The only meaningful statistic is Sum.</p> <p>Dimensions</p> <ul style="list-style-type: none"> TargetGroup TargetGroup, LoadBalancer

The AWS/ApplicationELB namespace includes the following metrics for user authentication.

Metric	Description
ELBAuthError	<p>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.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: The only meaningful statistic is Sum.</p> <p>Dimensions</p> <ul style="list-style-type: none"> LoadBalancer AvailabilityZone, LoadBalancer

Metric	Description
ELBAuthFailure	<p>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.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: The only meaningful statistic is Sum.</p> <p>Dimensions</p> <ul style="list-style-type: none"> LoadBalancer AvailabilityZone, LoadBalancer
ELBAuthLatency	<p>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.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: All statistics are meaningful.</p> <p>Dimensions</p> <ul style="list-style-type: none"> LoadBalancer AvailabilityZone, LoadBalancer
ELBAuthRefreshTokenSuccess	<p>The number of times the load balancer successfully refreshed user claims using a refresh token provided by the IdP.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: The only meaningful statistic is Sum.</p> <p>Dimensions</p> <ul style="list-style-type: none"> LoadBalancer AvailabilityZone, LoadBalancer
ELBAuthSuccess	<p>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.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: The most useful statistic is Sum.</p> <p>Dimensions</p> <ul style="list-style-type: none"> LoadBalancer AvailabilityZone, LoadBalancer

Metric	Description
ELBAuthUserClaimsSizeExceeded	<p>The number of times that a configured IdP returned user claims that exceeded 11K bytes in size.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Statistics: The only meaningful statistic is Sum.</p> <p>Dimensions</p> <ul style="list-style-type: none">LoadBalancerAvailabilityZone, 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: <code>app/load-balancer-name/1234567890123456</code> (the final portion of the load balancer ARN).
TargetGroup	Filters the metric data by target group. Specify the target group as follows: <code>targetgroup/target-group-name/1234567890123456</code> (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 Elastic Load Balancing. Statistics are metric data aggregations over specified period of time. When you request statistics, the returned data stream is identified by the metric name and dimension. A dimension is a name-value pair that uniquely identifies a metric. For example, you can request statistics for all the healthy EC2 instances behind a load balancer launched in a specific Availability Zone.

The `Minimum` and `Maximum` statistics reflect the minimum and maximum reported by the individual load balancer nodes. For example, suppose there are 2 load balancer nodes. 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.

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 Amazon EC2 console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. To view metrics filtered by target group, do the following:
 - a. In the navigation pane, choose **Target Groups**.
 - b. Select your target group, and 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://console.aws.amazon.com/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

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

Each access log file is automatically encrypted using SSE-S3 before it is stored in your S3 bucket and decrypted when you access it. You do not need to take any action; the encryption and decryption is performed transparently. Each log file is encrypted with a unique key, which is itself encrypted with a master key that is regularly rotated. For more information, see [Protecting data using server-side encryption with Amazon S3-managed encryption keys \(SSE-S3\)](#) in the *Amazon Simple Storage Service Developer Guide*.

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

Contents

- [Access log files \(p. 97\)](#)

- [Access log entries](#) (p. 98)
- [Bucket permissions](#) (p. 107)
- [Enable access logging](#) (p. 110)
- [Disable access logging](#) (p. 111)
- [Processing access log files](#) (p. 111)

Access log files

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

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

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

bucket

The name of the S3 bucket.

prefix

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

aws-account-id

The AWS account ID of the owner.

region

The Region for your load balancer and S3 bucket.

yyyy/mm/dd

The date that the log was delivered.

load-balancer-id

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

end-time

The date and time that the logging interval ended. For example, an end time of 20140215T2340Z contains entries for requests made between 23:35 and 23:40.

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:

```
s3://my-bucket/prefix/AWSLogs/123456789012/elasticloadbalancing/us-east-2/2016/05/01/123456789012_elasticloadbalancing_us-east-2_app.my-loadbalancer.1234567890abcdef_20140215T2340Z_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 Simple Storage Service Developer Guide*.

Access log entries

Elastic Load Balancing 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. Elastic Load Balancing does not log health check requests.

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

Elastic Load Balancing 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](#) (p. 98)
- [Actions taken](#) (p. 101)
- [Classification reasons](#) (p. 101)
- [Error reason codes](#) (p. 103)
- [Examples](#) (p. 105)

Syntax

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

Field	Description
type	The type of request or connection. The possible values are as follows (ignore any other values): <ul style="list-style-type: none">• <code>http</code> — HTTP• <code>https</code> — HTTP over TLS• <code>h2</code> — HTTP/2 over TLS• <code>grpcs</code> — gRPC over TLS• <code>ws</code> — WebSockets• <code>wss</code> — WebSockets over TLS
time	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	The resource ID of the load balancer. If you are parsing access log entries, note that resources IDs can contain forward slashes (/).
client:port	The IP address and port of the requesting client.
target:port	The IP address and port of the target that processed this request.

Field	Description
	<p>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 -.</p> <p>If the target is a Lambda function, this value is set to -.</p> <p>If the request is blocked by AWS WAF, this value is set to - and the value of <code>elb_status_code</code> is set to 403.</p>
<code>request_processing_time</code>	<p>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.</p> <p>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.</p> <p>This value can also be set to -1 if the registered target does not respond before the idle timeout.</p>
<code>target_processing_time</code>	<p>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.</p> <p>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.</p> <p>This value can also be set to -1 if the registered target does not respond before the idle timeout.</p>
<code>response_processing_time</code>	<p>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.</p> <p>This value is set to -1 if the load balancer can't send 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.</p>
<code>elb_status_code</code>	The status code of the response from the load balancer.
<code>target_status_code</code>	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 -.
<code>received_bytes</code>	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.
<code>sent_bytes</code>	The size of the response, in bytes, sent to the client (requester). For HTTP requests, this includes the headers. For WebSockets, this is the total number of bytes sent to the client on the connection.

Field	Description
"request"	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"	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	[HTTPS listener] The SSL cipher. This value is set to - if the listener is not an HTTPS listener.
ssl_protocol	[HTTPS listener] The SSL protocol. This value is set to - if the listener is not an HTTPS listener.
target_group_arn	The Amazon Resource Name (ARN) of the target group.
"trace_id"	The contents of the X-Amzn-Trace-Id header, enclosed in double quotes.
"domain_name"	[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"	[HTTPS listener] The ARN of the certificate presented to the client, enclosed in double quotes. This value is set to <code>session-reused</code> if the session is reused. This value is set to - if the listener is not an HTTPS listener.
matched_rule_priority	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	The time when the load balancer received the request from the client, in ISO 8601 format.
"actions_executed"	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 (p. 101) . If no action was taken, such as for a malformed request, this value is set to -.
"redirect_url"	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"	The error reason code, enclosed in double quotes. If the request failed, this is one of the error codes described in Error reason codes (p. 103) . If the actions taken do not include an authenticate action or the target is not a Lambda function, this value is set to -.

Field	Description
"target:port_list"	<p>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.</p> <p>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 -.</p> <p>If the target is a Lambda function, this value is set to -.</p> <p>If the request is blocked by AWS WAF, this value is set to - and the value of elb_status_code is set to 403.</p>
"target_status_code_list"	<p>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.</p> <p>This value is recorded only if a connection was established to the target and the target sent a response. Otherwise, it is set to -.</p>
"classification"	<p>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.</p> <p>If the request complies with RFC 7230, this value is set to -.</p>
"classification_reason"	<p>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 (p. 101). If the request complies with RFC 7230, this value is set to -.</p>

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.
- **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.
- **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 \(p. 16\)](#).

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
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
MultipleTransferEncoding	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
UndefinedContentLength	There is no Content-Length header defined for a GET or HEAD request.	Ambiguous
UndefinedTransferEncoding	There is no Transfer-Encoding header defined for 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](#) (p. 48).

Code	Description	Metric
<code>AuthInvalidCookie</code>	The authentication cookie is not valid.	<code>ELBAuthFailure</code>
<code>AuthInvalidGrantError</code>	The authorization grant code from the token endpoint is not valid.	<code>ELBAuthFailure</code>
<code>AuthInvalidIdToken</code>	The ID token is not valid.	<code>ELBAuthFailure</code>
<code>AuthInvalidStateParam</code>	The state parameter is not valid.	<code>ELBAuthFailure</code>
<code>AuthInvalidTokenResponse</code>	The response from the token endpoint is not valid.	<code>ELBAuthFailure</code>
<code>AuthInvalidUserInfoResponse</code>	The response from the user info endpoint is not valid.	<code>ELBAuthFailure</code>
<code>AuthMissingCodeParam</code>	The authentication response from the authorization endpoint is missing a query parameter named 'code'.	<code>ELBAuthFailure</code>
<code>AuthMissingHostHeader</code>	The authentication response from the authorization endpoint is missing a host header field.	<code>ELBAuthError</code>
<code>AuthMissingStateParam</code>	The authentication response from the authorization endpoint is missing a query parameter named 'state'.	<code>ELBAuthFailure</code>
<code>AuthTokenEpRequestFailure</code>	There is an error response (non-2XX) from the token endpoint.	<code>ELBAuthError</code>
<code>AuthTokenEpRequestTimeout</code>	The load balancer is unable to communicate with the token endpoint.	<code>ELBAuthError</code>
<code>AuthUnhandledException</code>	The load balancer encountered an unhandled exception.	<code>ELBAuthError</code>
<code>AuthUserInfoEpRequestFailure</code>	There is an error response (non-2XX) from the IdP user info endpoint.	<code>ELBAuthError</code>
<code>AuthUserInfoEpRequestTimeout</code>	The load balancer is unable to communicate with the IdP user info endpoint.	<code>ELBAuthError</code>
<code>AuthUserInfoResponseTooLarge</code>	The size of the claims returned by the IdP exceeded 11K bytes.	<code>ELBAuthUserClaimsSizeExceeded</code>

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
LambdaConnectionTimeout	An attempt to connect to Lambda timed out.	LambdaInternalError
LambdaEC2AccessDenied	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
LambdaENILimitReached	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
LambdaInvalidRuntime	The specified version of the Lambda runtime is not supported.	LambdaUserError
LambdaInvalidSecurityGroup	The security group ID specified in the configuration of the Lambda function is not valid.	LambdaUserError
LambdaInvalidSubnet	The subnet ID specified in the configuration of the Lambda function is not valid.	LambdaUserError
LambdaInvalidZipFile	Lambda could not unzip the specified function zip file.	LambdaUserError
LambdaKMSAccessDenied	Lambda could not decrypt environment variables because access to the KMS key was denied. Check the KMS permissions of the Lambda function.	LambdaUserError

Code	Description	Metric
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
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
LambdaSubnetIPAddressException	Lambda reached its default 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

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.
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.

Examples

The following are example log entries. Note that the 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 "-" "-"
```

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 "-" "-"
```

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
"_" "_"
```

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 "-" "-"
```

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 "-" "-"
```

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" "-" "-" "-" "-" "-" "
```

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" "-" "-" "-" "
```

Bucket permissions

When you enable access logging, you must specify an S3 bucket for the access logs. The bucket must meet the following requirements.

Requirements

- The bucket must be located in the same Region as the load balancer.
- Amazon S3-Managed Encryption Keys (SSE-S3) is required. No other encryption options are supported.
- The bucket must have a bucket policy that grants Elastic Load Balancing permission to write the access logs to your bucket. Bucket policies are a collection of JSON statements written in the access policy language to define access permissions for your bucket. Each statement includes information about a single permission and contains a series of elements.

Use one of the following options to prepare an S3 bucket for access logging.

Options

- To create a bucket and enable access logging using the Elastic Load Balancing console, skip to [Enable access logging \(p. 110\)](#) and select the option to have the console create the bucket and bucket policy for you.
- To use an existing bucket and add the required bucket policy using the Amazon S3 console, use the following procedure but skip the steps marked "[Skip to use existing bucket]".
- To create a bucket and add the required bucket policy using the Amazon S3 console (for example, if you are using the AWS CLI or an API to enable access logging), use the following procedure.

To prepare an Amazon S3 bucket for access logging

1. Open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
2. [Skip to use existing bucket] Choose **Create bucket**.
3. [Skip to use existing bucket] 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 Simple Storage Service Developer Guide*.
 - b. For **Region**, select the Region where you created your load balancer.
 - c. Choose **Create**.

4. Select the bucket. Choose **Permissions** and then choose **Bucket Policy**.
5. If you are creating a new bucket policy, copy the entire policy document to the policy editor, then replace the placeholders with the corresponding information. If you are editing an existing bucket policy, copy only the new statement from the policy document (the text between the [and] of the Statement element).

[Availability Zones and Local Zones] Use the following policy. Update the placeholders for the name and prefix for your bucket, the ID of the AWS account for Elastic Load Balancing (based on the Region for your load balancer), and the ID of your AWS account.

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

The following table contains the account IDs to use in place of *elb-account-id* in your bucket policy.

Region	Region name	Elastic Load Balancing account ID
us-east-1	US East (N. Virginia)	127311923021
us-east-2	US East (Ohio)	033677994240
us-west-1	US West (N. California)	027434742980
us-west-2	US West (Oregon)	797873946194
af-south-1	Africa (Cape Town)	098369216593
ca-central-1	Canada (Central)	985666609251

Region	Region name	Elastic Load Balancing account ID
eu-central-1	Europe (Frankfurt)	054676820928
eu-west-1	Europe (Ireland)	156460612806
eu-west-2	Europe (London)	652711504416
eu-south-1	Europe (Milan)	635631232127
eu-west-3	Europe (Paris)	009996457667
eu-north-1	Europe (Stockholm)	897822967062
ap-east-1	Asia Pacific (Hong Kong)	754344448648
ap-northeast-1	Asia Pacific (Tokyo)	582318560864
ap-northeast-2	Asia Pacific (Seoul)	600734575887
ap-northeast-3	Asia Pacific (Osaka-Local)	383597477331
ap-southeast-1	Asia Pacific (Singapore)	114774131450
ap-southeast-2	Asia Pacific (Sydney)	783225319266
ap-south-1	Asia Pacific (Mumbai)	718504428378
me-south-1	Middle East (Bahrain)	076674570225
sa-east-1	South America (São Paulo)	507241528517
us-gov-west-1*	AWS GovCloud (US-West)	048591011584
us-gov-east-1*	AWS GovCloud (US-East)	190560391635
cn-north-1*	China (Beijing)	638102146993
cn-northwest-1*	China (Ningxia)	037604701340

* These Regions requires a separate account. For more information, see [AWS GovCloud \(US-West\)](#) and [China \(Beijing\)](#).

[Outpost] Use the following policy. Update the placeholders for the name and prefix for your bucket and the ID of your AWS account.

```
{
  "Effect": "Allow",
  "Principal": {
    "Service": "logdelivery.elb.amazonaws.com"
  },
  "Action": "s3:PutObject",
  "Resource": "arn:aws:s3:::bucket-name/prefix/AWSLogs/your-aws-account-id/*",
  "Condition": {
    "StringEquals": {
```

```
    "s3:x-amz-acl": "bucket-owner-full-control"
  }
}
```

6. Choose **Save**.

Enable access logging

When you enable access logging for your load balancer, you must specify the name of the S3 bucket where the load balancer will store the logs. The bucket must be in the same Region as your load balancer, and must have a bucket policy that grants Elastic Load Balancing permission to write the access logs to the bucket. The bucket can be owned by a different account than the account that owns the load balancer.

To enable access logging using the console

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

To enable access logging using the AWS CLI

Use the `modify-load-balancer-attributes` command.

To verify that Elastic Load Balancing created a test file in your S3 bucket

After access logging is enabled for your load balancer, Elastic Load Balancing 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.

1. Open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
2. For **All Buckets**, select your S3 bucket.
3. Navigate to the bucket you specified for access logging and look for `ELBAccessLogTestFile`. For example, if you used the console to create the bucket and bucket policy, the path is as follows:

```
my-bucket/prefix/AWSLogs/123456789012/ELBAccessLogTestFile
```

To manage the S3 bucket for your access logs

After you enable access logging, be sure to disable access logging before you delete the bucket with your 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, Elastic Load Balancing could write the access logs for your load balancer to this new bucket.

Disable access logging

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

To disable access logging using the console

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

To disable access logging using the AWS CLI

Use the [modify-load-balancer-attributes](#) command.

Processing access log files

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

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

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

Request tracing for your Application Load Balancer

You can use request tracing to track HTTP requests from clients to targets or other services. 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.

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 \(p. 96\)](#).

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.

time

The epoch time, in seconds.

id

The trace identifier.

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=1-67891234-12456789abcdef012345678;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=1-67891234-12456789abcdef012345678;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.

Logging API calls for your Application Load Balancer using AWS CloudTrail

Elastic Load Balancing is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Elastic Load Balancing. CloudTrail captures all API calls for Elastic Load Balancing as events. The calls captured include calls from the AWS Management Console and code calls to the Elastic Load Balancing API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Elastic Load Balancing. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to Elastic Load Balancing, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the [AWS CloudTrail User Guide](#).

To monitor other actions for your load balancer, such as when a client makes a request to your load balancer, use access logs. For more information, see [Access logs for your Application Load Balancer](#) (p. 96).

Elastic Load Balancing information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in Elastic Load Balancing, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see [Viewing events with CloudTrail event history](#).

For an ongoing record of events in your AWS account, including events for Elastic Load Balancing, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- [Overview for creating a trail](#)
- [CloudTrail supported services and integrations](#)
- [Configuring Amazon SNS notifications for CloudTrail](#)
- [Receiving CloudTrail log files from multiple regions](#) and [Receiving CloudTrail log files from multiple accounts](#)

All Elastic Load Balancing actions for Application Load Balancers are logged by CloudTrail and are documented in the [Elastic Load Balancing API Reference version 2015-12-01](#). For example, calls to the `CreateLoadBalancer` and `DeleteLoadBalancer` actions generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the [CloudTrail `userIdentity` element](#).

Understanding Elastic Load Balancing log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The log files include events for all AWS API calls for your AWS account, not just Elastic Load Balancing API calls. You can locate calls to the Elastic Load Balancing API by checking for `eventSource` elements with the value `elasticloadbalancing.amazonaws.com`. To view a record for a specific action, such as `CreateLoadBalancer`, check for `eventName` elements with the action name.

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

Example Example: CreateLoadBalancer

```
{
  "eventVersion": "1.03",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "123456789012",
    "arn": "arn:aws:iam::123456789012:user/Alice",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "Alice"
  },
  "eventTime": "2016-04-01T15:31:48Z",
  "eventSource": "elasticloadbalancing.amazonaws.com",
  "eventName": "CreateLoadBalancer",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "198.51.100.1",
  "userAgent": "aws-cli/1.10.10 Python/2.7.9 Windows/7 botocore/1.4.1",
  "requestParameters": {
    "subnets": ["subnet-8360a9e7", "subnet-b7d581c0"],
    "securityGroups": ["sg-5943793c"],
    "name": "my-load-balancer",
    "scheme": "internet-facing"
  },
  "responseElements": {
    "loadBalancers": [{
      "type": "application",
      "loadBalancerName": "my-load-balancer",
      "vpcId": "vpc-3ac0fb5f",
      "securityGroups": ["sg-5943793c"],
      "state": {"code": "provisioning"},
      "availabilityZones": [
        {"subnetId": "subnet-8360a9e7", "zoneName": "us-west-2a"},
        {"subnetId": "subnet-b7d581c0", "zoneName": "us-west-2b"}
      ],
      "dnsName": "my-load-balancer-1836718677.us-west-2.elb.amazonaws.com",
      "canonicalHostedZoneId": "Z2P70J7HTTTPU",
      "createdTime": "Apr 11, 2016 5:23:50 PM",
      "loadBalancerArn": "arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/app/my-load-balancer/ffcdace1759e1d0",
      "scheme": "internet-facing"
    }]
  },
  "requestID": "b9960276-b9b2-11e3-8a13-f1ef1EXAMPLE",
}
```



```
"eventID": "6f4ab5bd-2daa-4d00-be14-d92efEXAMPLE",  
"eventType": "AwsApiCall",  
"apiVersion": "2015-12-01",  
"recipientAccountId": "123456789012"  
}
```

Example Example: DeleteLoadBalancer

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

Troubleshoot your Application Load Balancers

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

Issues

- [A registered target is not in service \(p. 116\)](#)
- [Clients cannot connect to an internet-facing load balancer \(p. 117\)](#)
- [The load balancer sends requests to unhealthy targets \(p. 117\)](#)
- [The load balancer sends a response code of 000 \(p. 117\)](#)
- [The load balancer generates an HTTP error \(p. 117\)](#)
- [A target generates an HTTP error \(p. 120\)](#)

A registered target is not in service

If a target is taking longer than expected to enter the `InService` state, it might be failing health checks. Your target is not in service until it passes one health check. For more information, see [Health checks for your target groups \(p. 66\)](#).

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.

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.

The load balancer sends requests to unhealthy targets

If there is at least one healthy target in a target group, the load balancer routes requests only to the healthy targets. If a target group contains only unhealthy targets, the load balancer routes requests to the unhealthy targets.

The load balancer sends a response code of 000

With HTTP/2 connections, if the compressed length of any of the headers exceeds 8K, 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 \(p. 118\)](#)
- [HTTP 401: Unauthorized \(p. 118\)](#)
- [HTTP 403: Forbidden \(p. 118\)](#)
- [HTTP 405: Method not allowed \(p. 118\)](#)
- [HTTP 408: Request timeout \(p. 118\)](#)
- [HTTP 413: Payload too large \(p. 118\)](#)
- [HTTP 414: URI too long \(p. 118\)](#)
- [HTTP 460 \(p. 119\)](#)
- [HTTP 463 \(p. 119\)](#)
- [HTTP 464 \(p. 119\)](#)
- [HTTP 500: Internal server error \(p. 119\)](#)

- [HTTP 501: Not implemented \(p. 119\)](#)
- [HTTP 502: Bad gateway \(p. 119\)](#)
- [HTTP 503: Service unavailable \(p. 120\)](#)
- [HTTP 504: Gateway timeout \(p. 120\)](#)
- [HTTP 505: Version not supported \(p. 120\)](#)
- [HTTP 561: Unauthorized \(p. 120\)](#)

HTTP 400: Bad request

Possible causes:

- The client sent a malformed request that does not meet the HTTP specification.
- The request header exceeded 16K per request line, 16K per single header, or 64K for the entire header.

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.

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

The target is a Lambda function and the request body exceeds 1 MB.

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 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.

HTTP 501: Not implemented

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.

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 load balancer encountered an SSL handshake error or SSL handshake timeout (10 seconds) when connecting to a target.
- 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.

HTTP 503: Service unavailable

The target groups for the load balancer have no registered targets.

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.

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 561: Unauthorized

You configured a listener rule to authenticate users, but the IdP returned an error code when authenticating the user.

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.

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 Elastic Load Balancing.

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

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

Regional

- Load balancers per Region: 50
- Target groups per Region: 3000 *

Load balancer

- Listeners per load balancer: 50
- Targets per load balancer: 1000
- Target groups per load balancer: 100
- Subnets per Availability Zone per load balancer: 1
- Security groups per load balancer: 5
- Rules per load balancer (not counting default rules): 100
- Certificates per load balancer (not counting default certificates): 25
- Number of times a target can be registered per load balancer: 100

Target group

- Load balancers per target group: 1
- Targets per target group (instances or IP addresses): 1000
- Targets per target group (Lambda functions): 1

Rule

- Target groups per action: 5
- Match evaluations per rule: 5
- Wildcards per rule: 5
- Actions per rule: 2 (one optional authentication action, one required action)

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

Document history for Application Load Balancers

The following table describes the releases for Application Load Balancers.

Feature	Description	Date
Security policy for FS supporting TLS version 1.2	This release adds a security policy for Forward Secrecy (FS) supporting TLS version 1.2. For more information, see Security policies (p. 38) .	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. For more information, see Application Load Balancers and AWS WAF (p. 17) .	November 13, 2020
gRPC and HTTP/2 support	This release adds support for gRPC workloads and end-to-end HTTP/2. For more information, see Protocol version (p. 57) .	October 29, 2020
Outpost support	You can provision an Application Load Balancer on your Outpost. The load balancer operates in a single subnet and scales automatically using capacity from the Outpost.	September 8, 2020
Desync mitigation mode	This release adds support for desync mitigation mode. For more information, see Desync mitigation mode (p. 16) .	August 17, 2020
Least outstanding requests	This release adds support for the least outstanding requests algorithm. For more information, see Routing algorithm (p. 60) .	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. For more information, see Forward actions (p. 27) .	November 19, 2019

Feature	Description	Date
New attribute	This release adds support for the <code>routing.http.drop_invalid_header_fields.enabled</code> attribute. For more information, see Load balancer attributes (p. 13) .	November 11, 2019
Advanced request routing	This release extends the existing support for host header and path-based routing by adding conditions for your listener rules based on standard and custom HTTP headers and methods, query parameters, and source IP addresses. For more information, see Rule condition types (p. 31) .	March 27, 2019
Lambda functions as a target	This release add support to register your Lambda functions as a target. For more information, see Lambda functions as targets (p. 74) .	November 29, 2018
Fixed-response actions	This release adds support for the load balancer to return a custom HTTP response. For more information, see Fixed-response actions (p. 27) .	July 25, 2018
Redirect actions	This release adds support for the load balancer to redirect requests to a different URL. For more information, see Redirect actions (p. 29) .	July 25, 2018
Security policies for FS and TLS 1.2	This release adds security policies for Forward Secrecy (FS) and TLS 1.2. For more information, see Security policies (p. 38) .	June 6, 2018
Authentication support	This release adds support for the load balancer to authenticate users of your applications using their corporate or social identities before routing requests. For more information, see Authenticate users using an Application Load Balancer (p. 48) .	May 30, 2018

Feature	Description	Date
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. For more information, see Slow start mode (p. 61) .	March 24, 2018
Resource-level permissions	This release adds support for resource-level permissions and tagging condition keys. For more information, see Authentication and access control in the <i>Elastic Load Balancing User Guide</i> .	May 10, 2018
SNI support	This release adds support for Server Name Indication (SNI). For more information, see SSL certificates (p. 36) .	October 10, 2017
IP addresses as targets	This release adds support for registering IP addresses as targets. For more information, see Target type (p. 57) .	August 31, 2017
Host-based routing	This release adds support for routing requests based on the host names in the host header. For more information, see Host conditions (p. 32) .	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. For more information, see Security policies (p. 38) .	February 6, 2017
IPv6 support	This release adds support for IPv6 addresses. For more information, see IP address type (p. 14) .	January 25, 2017
Request tracing	This release adds support for request tracing. For more information, see Request tracing for your Application Load Balancer (p. 111) .	November 22, 2016
Percentiles support for the TargetResponseTime metric	This release adds support for the new percentile statistics supported by Amazon CloudWatch. For more information, see Statistics for Application Load Balancer metrics (p. 94) .	November 17, 2016

Feature	Description	Date
New load balancer type	This release of Elastic Load Balancing introduces Application Load Balancers.	August 11, 2016