Elastic Load Balancing

Application Load Balancers

Elastic Load Balancing: Application Load Balancers

Table of Contents

What Is an Application Load Balancer?

Elastic Load Balancing supports two types of load balancers: Application Load Balancers and Classic Load Balancers. This guide discusses Application Load Balancers. For more information about Classic Load Balancers, see the Classic Load Balancer Guide.

Application Load Balancer Overview

An Application Load Balancer functions at the application layer, the seventh layer of the Open Systems Interconnection (OSI) model. The load balancer makes routing decisions based on the content of the application traffic in the HTTP messages.

The load balancer distributes incoming application traffic across multiple targets, such as EC2 instances, in multiple Availability Zones. This increases the fault tolerance of your applications. Elastic Load Balancing detects unhealthy targets and routes traffic only to healthy targets.

The load balancer serves as a single point of contact for clients. This increases the availability of your application. 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*.

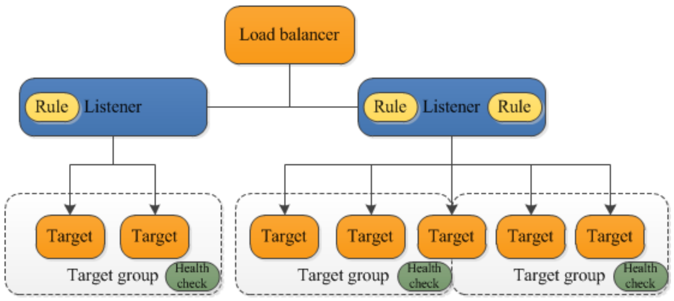
Components

The *load balancer* serves as the single point of contact for clients. You add one or more listeners to your load balancer.

A *listener* checks for connection requests from clients, using the protocol and port that you configure, and forwards requests to one or more target groups, based on the rules that you define. Each rule specifies a target group, condition, and priority. When the condition is met, the traffic is forwarded to the target group. You must define a default rule for each listener, and you can add rules that specify different target groups based on the content of the request (also known as *content-based routing*).

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. 12)
- Listeners (p. 19)
- Target Groups (p. 26)

Benefits

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

- Support for path-based routing. 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 routing requests to multiple services on a single EC2 instance by registering the instance using multiple ports.
- Support for containerized applications. Amazon EC2 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 Features of Elastic Load Balancing in the *Elastic Load Balancing User Guide*.

How to Get Started

To create an Application Load Balancer, try one of the following tutorials:

- Getting Started with Elastic Load Balancing in the *Elastic Load Balancing User Guide*.
- Tutorial: Use Path-Based Routing with Your Application Load Balancer (p. 4)
- Tutorial: Use Microservices as Targets with Your Application Load Balancer (p. 6)

Pricing

For more information, see Application Load Balancer Pricing.

Tutorials for Application Load Balancers

The following Elastic Load Balancing tutorials show you how to perform common tasks using an Application Load Balancer.

- Getting Started with Elastic Load Balancing (*Elastic Load Balancing User Guide*)
- Tutorial: Use Path-Based Routing with Your Application Load Balancer (p. 4)
- Tutorial: Use Microservices as Targets with Your Application Load Balancer (p. 6)
- Tutorial: Create an Application Load Balancer Using the AWS CLI (p. 8)

Tutorial: Use Path-Based Routing with Your Application Load Balancer

You can create a listener with rules to forward requests based on the URL path. This is known as *path-based routing*. If you are running microservices, you can route traffic to multiple back-end services using path-based routing. For example, you can route general requests to one target group and requests to render images to another target group.

Before You Begin

- 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. 33).
- Verify that your microservices are deployed on the EC2 instances that you plan to register.

Create Your Load Balancer

To create a load balancer that uses path-based routing

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation bar, select the same region that you selected for your EC2 instances.

3. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
4. Create a target group for the first set of targets as follows:

    a. Choose **Create target group**.

    b. Specify a name, protocol, port, and VPC for the target group, and then choose **Create**.

    c. Select the new target group.

    d. On the **Targets** tab, choose **Edit**.

    e. For **Instances**, select one or more instances. Specify a port for the instances, choose **Add to registered**, and then choose **Save**.

        Note that the status of the instances is `initial` until the instances are registered and have passed health checks, and then it is `unused` until you configure the target group to receive traffic from the load balancer.
5. Create a target group for the second set of targets as follows:

    a. Choose **Create target group**.

    b. Specify a name, protocol, port, and VPC for the target group, and then choose **Create**.

    c. On the **Targets** tab, choose **Edit**.

    d. For **Instances**, select one or more instances. Specify a port for the instances, choose **Add to registered**, and then choose **Save**.

        Note that the status of the instances is `initial` until the instances are registered and have passed health checks, and then it is `unused` until you configure the target group to receive traffic from the load balancer.
6. On the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
7. Choose **Create Load Balancer**.
8. For **Select load balancer type**, choose **Application Load Balancer**.
9. Choose **Continue**.
10. Complete the **Configure Load Balancer** page as follows:

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

        The name of your Application Load Balancer must be unique within your set of Application Load Balancers for the region, can have a maximum of 32 characters, can contain only alphanumeric characters and hyphens, and must not begin or end with a hyphen.

    b. 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.

    c. 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 port of the listener, or choose **Add** to add another listener.

    d. For **Availability Zones**, select the VPC that you used for your EC2 instances. Select at least two Availability Zones. If there is one subnet for an Availability Zone, it is selected. If there is more than one subnet for an Availability Zone, select one of the subnets. Note that you can select only one subnet per Availability Zone.

    e. Choose **Next: Configure Security Settings**.
11. (Optional) If you created a secure listener in the previous step, complete the **Configure Security Settings** page as follows:

    a. If you have a certificate from AWS Certificate Manager, choose **Choose an existing certificate from AWS Certificate Manager (ACM)**, and then choose the certificate from **Certificate name**.

    b. If you have already uploaded a certificate using IAM, choose **Choose an existing certificate from AWS Identity and Access Management (IAM)**, and then choose your certificate from **Certificate name**.

   c. If you have a certificate ready to upload, choose **Upload a new SSL Certificate to AWS Identity and Access Management (IAM)**. For **Certificate name**, type a name for the certificate. For **Private Key**, copy and paste the contents of the private key file (PEM-encoded). In **Public Key Certificate**, copy and paste the contents of the public key certificate file (PEM-encoded). In **Certificate Chain**, copy and paste the contents of the certificate chain file (PEM-encoded), unless you are using a self-signed certificate and it's not important that browsers implicitly accept the certificate.
   d. For **Select policy**, keep the one existing predefined security policy.

12. Choose **Next: Configure Security Groups**.
13. Complete the **Configure Security Groups** page as follows:

    a. Select **Create a new security group**.
    b. 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 port that you selected for your load balancer on the **Configure Load Balancer** page.
    c. Choose **Next: Configure Routing**.

14. Complete the **Configure Routing** page as follows:

    a. For **Target group**, choose `Existing target group`.
    b. For **Name**, choose the first target group that you created.
    c. Choose **Next: Register Targets**.

15. On the **Register Targets** page, the instances that you registered with the target group appear under **Registered instances**. You can't modify the targets registered with the target group until after you complete the wizard. Choose **Next: Review**.
16. On the **Review** page, choose **Create**.
17. After you are notified that your load balancer was created successfully, choose **Close**.
18. Select the newly created load balancer.
19. On the **Listeners** tab, use the arrow to view the rules for the listener, and then choose **Add rule**. Specify the rule as follows:

    a. For **Target group name**, choose the second target group that you created.
    b. For **Path pattern** specify the exact pattern to be used for path-based routing (for example, `/img/*`). For more information, see Listener Rules (p. 20).
    c. Choose **Save**.

Tutorial: Use Microservices as Targets with Your Application Load Balancer

You can use a microservices architecture to structure your application as services that you can develop and deploy independently. You can install one or more of these services on each EC2 instance, with each service accepting connections on a different port. You can use a single Application Load Balancer to route requests to all the services for your application. When you register an EC2 instance with a target group, you can register it multiple times; for each service, register the instance using the port for the service.

Important
If you are using an Application Load Balancer and deploying your services using Amazon EC2 Container Service (Amazon ECS), you can use dynamic port mapping to support multiple tasks from a single service on the same container instance. Amazon ECS manages updates to your services by automatically registering and deregistering containers with your target group using the instance ID and port for each container. For more information, see Service Load Balancing in the *Amazon EC2 Container Service Developer Guide*.

Before You Begin

- Launch your EC2 instances. Ensure that the security groups for the instances allow access from the load balancer security group on the listener ports and the health check ports. For more information, see Target Security Groups (p. 33).
- Deploy your services to your EC2 instances (for example, using containers.)

Create Your Load Balancer

To create a load balancer that uses multiple services as targets

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation bar, select the same region that you selected for your EC2 instances.
3. On the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
4. Choose **Create Load Balancer**.
5. For **Select load balancer type**, choose **Application Load Balancer**.
6. Choose **Continue**.
7. Complete the **Configure Load Balancer** page as follows:

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

       The name of your Application Load Balancer must be unique within your set of Application Load Balancers for the region, can have a maximum of 32 characters, can contain only alphanumeric characters and hyphens, and must not begin or end with a hyphen.

    b. 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.

    c. 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 port of the listener, or choose **Add** to add another listener.

    d. For **Availability Zones**, select the VPC that you used for your EC2 instances. Select at least two Availability Zones. If there is one subnet for an Availability Zone, it is selected. If there is more than one subnet for an Availability Zone, select one of the subnets. Note that you can select only one subnet per Availability Zone.

    e. Choose **Next: Configure Security Settings**.

8. (Optional) If you created a secure listener in the previous step, complete the **Configure Security Settings** page as follows:

    a. If you have a certificate from AWS Certificate Manager, choose **Choose an existing certificate from AWS Certificate Manager (ACM)**, and then choose the certificate from **Certificate name**.

    b. If you have already uploaded a certificate using IAM, choose **Choose an existing certificate from AWS Identity and Access Management (IAM)**, and then choose your certificate from **Certificate name**.

    c. If you have a certificate ready to upload, choose **Upload a new SSL Certificate to AWS Identity and Access Management (IAM)**. For **Certificate name**, type a name for the certificate. For **Private Key**, copy and paste the contents of the private key file (PEM-encoded). In **Public Key Certificate**, copy and paste the contents of the public key certificate file (PEM-encoded). In **Certificate Chain**, copy and paste the contents of the certificate chain file (PEM-encoded), unless you are using a self-signed certificate and it's not important that browsers implicitly accept the certificate.

    d. For **Select policy**, keep the one existing predefined security policy.

9. Choose **Next: Configure Security Groups**.

10. Complete the **Configure Security Groups** page as follows:

    a. Select **Create a new security group**.
    b. 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 port that you selected for your load balancer on the **Configure Load Balancer** page.
    c. Choose **Next: Configure Routing**.

11. Complete the **Configure Routing** page as follows:

    a. For **Target group**, keep the default, `New target group`.
    b. For **Name**, type a name for the new target group.
    c. Set **Protocol** and **Port** as needed.
    d. For **Health checks**, keep the default health check settings.
    e. Choose **Next: Register Targets**.

12. For **Register Targets**, do the following:

    a. For **Instances**, select an EC2 instance.
    b. Type the port used by the service, and then choose **Add to registered**.
    c. Repeat for each service to register. When you are finished, choose **Next: Review**.

13. On the **Review** page, choose **Create**.
14. After you are notified that your load balancer was created successfully, choose **Close**.

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. 33).

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-12345678 subnet-23456789 --security-groups sg-12345678
```

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

```
arn:aws:elasticloadbalancing:us-west-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-12345678
```

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

```
arn:aws:elasticloadbalancing:us-west-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-12345678 Id=i-23456789
```

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-west-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 the certificate using AWS Certificate Manager (ACM), which integrates with Elastic Load Balancing. For more information, see Request a Certificate in the *AWS Certificate Manager User Guide*.
- Create the certificate using SSL/TLS tools (for example, OpenSSL), and then upload the certificate using AWS Identity and Access Management (IAM). For more information, see Creating a Server Certificate 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. Note that you must specify an SSL server certificate and a security policy when you create an HTTPS listener.

```
aws elbv2 create-listener --load-balancer-arn loadbalancer-arn \
--protocol HTTPS --port 443  \
--certificates CertificateArn=arn:aws:iam::123456789012:server-
certificate/my-server-cert \
--ssl-policy ELBSecurityPolicy-2015-05 \
--default-actions Type=forward,TargetGroupArn=targetgroup-arn
```

Add Targets Using Port Overrides

If you have multiple ECS containers on a single instance, each container accepts connections on a different port. You can register the instance with the target group multiple times, each time with a different port.

To add targets using port overrides

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-12345678
```

2. Use the register-targets command to register your instances with your target group. Notice that the instance IDs are the same for each container, but the ports are different.

```
aws elbv2 register-targets --target-group-arn targetgroup-arn  \
--targets Id=i-12345678,Port=80 Id=i-12345678,Port=766
```

3. Use the create-rule command to add a rule to your listener that forwards requests to the target group:

```
aws elbv2 create-rule --listener-arn listener-arn --priority 10 \
--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-12345678
```

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-12345678 Id=i-23456789
```

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, in two or more Availability Zones. To configure your load balancer, you create target groups (p. 26), and then register targets with your target groups. You also create listeners (p. 19) to check for connection requests from clients, and listener rules to route requests from clients to the targets in one or more target groups.

Contents

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 associated with your load balancer security group 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. 15).

Load Balancer State

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

`provisioning`
: The load balancer is being set up.

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

`failed`
: The load balancer could not be set up.

Load Balancer Attributes

The following are the load balancer attributes:

`deletion_protection.enabled`
: Indicates whether deletion protection is enabled.

`idle_timeout.timeout_seconds`
: The idle timeout value, in seconds.

IP Address Type

You can set the IP address type of your Internet-facing load balancer when you create it or after it is active. Note that internal load balancers must use IPv4 addresses.

The following are the load balancer IP address types:

`ipv4`
: The load balancer supports only IPv4 addresses (for example, 192.0.2.1)

`dualstack`
: The load balancer supports both IPv4 and IPv6 addresses (for example, 2001:0db8:85a3:0:0:8a2e:0370:7334).

Clients that use IPv4 addresses resolve the A record and clients that use IPv6 addresses resolve the AAAA record.

For more information, see IP Address Types for Your Application Load Balancer (p. 16).

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 delete protection**.
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 delete protection**.
6. Choose **Save**.

To enable or disable deletion protection using the AWS CLI

Use the modify-load-balancer-attributes command.

Connection Idle Timeout

For each request that a client makes through a load balancer, the load balancer maintains two connections. A front-end connection is between a client and the load balancer, and a back-end connection is between the load balancer and a target. For each front-end connection, the load balancer manages an idle timeout that is triggered when no data is sent over the connection for a specified time period. If no data has been sent or received by the time that the idle timeout period elapses, the load balancer closes the front-end connection.

By default, Elastic Load Balancing sets the idle timeout value to 60 seconds. Therefore, if the target doesn't send some data at least every 60 seconds while the request is in flight, the load balancer can close the front-end 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 keep-alive option for your EC2 instances. You can enable keep-alive in your web server settings or in the kernel settings for your EC2 instances. Keep-alive, when enabled, enables the instances to tear down back-end connections when an operation is finished.

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, type a value for **Idle timeout**, in seconds. The valid range is 1-3600. The default is 60 seconds.
6. Choose **Save**.

To update the idle timeout value using the AWS CLI

Use the modify-load-balancer-attributes command.

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 an Availability Zone, select the check box for that Availability Zone. If there is one subnet for that Availability Zone, it is selected. If there is more than one subnet for that Availability Zone, select one of the subnets. Note that you can select only one subnet per Availability Zone.
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.

Recommended Rules

The recommended rules depend on the type of load balancer (Internet-facing or internal).

Internet-facing Load Balancer

Inbound		
Source	**Port Range**	**Comment**
0.0.0.0/0	*listener*	Allow all inbound traffic on the load balancer listener port

Outbound		
Destination	**Port Range**	**Comment**
instance security group	*instance listener*	Allow outbound traffic to instances on the instance listener port
instance security group	*health check*	Allow outbound traffic to instances on the health check port

Internal Load Balancer

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

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 to route IPv4 traffic only or to route both IPv4 and IPv6 traffic.

IPv6 Requirements

- An Internet-facing load balancer.
- Your virtual private cloud (VPC) has subnets with associated IPv6 CIDR blocks. For more information, see IPv6 Addresses in the *Amazon EC2 User Guide*.

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 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 load balancer using the AWS CLI

Use the add-tags and remove-tags commands.

Delete Your 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. 13).

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 Your Target Group (p. 34).

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 the targets in one or more target groups.

Contents

Listener Configuration

Listeners support the following protocols and ports:

- **Protocols**: HTTP, HTTPS
- **Ports**: 1-65535

You can use an HTTPS listener to offload the work of encryption and decryption to your load balancer so that your targets can focus on their main work. If the listener protocol is HTTPS, you must deploy exactly one SSL server certificate on the listener. For more information, see Create an HTTPS Listener for Your Application Load Balancer (p. 21).

You can use WebSockets with both HTTP and HTTPS listeners.

You can use HTTP/2 with HTTPS listeners. You can send up to 128 requests in parallel using one HTTP/2 connection. The load balancer converts these to individual HTTP/1.1 requests and distributes them across the healthy targets in the target group using the round robin routing algorithm. Because HTTP/2 uses front-end connections more efficiently, you might notice fewer connections between clients and the load balancer. Note that you can't use the server-push feature of HTTP/2.

Listener Rules

Each listener can have one or more rules for routing requests. Each rule can have one action and one condition. When the condition for a rule is met, then its action is taken. For example, you can define a rule that forwards requests to a target group based on the URL in the request (also known as *path-based routing*).

When you create a listener, you must define a default rule for the listener. You can add or remove other rules at any time. A default rule can't have a condition. If no conditions for any of a listener's rules are met, then the action specified by the default rule for the listener is taken.

Rule Actions

Each rule action has a type and a target group. Currently, the only supported type is `forward`. You can create a rule that forwards requests to the specified target group.

Rule Conditions

Each rule condition can have one *path pattern*. You can create multiple rules, each with a different path pattern, in order to route requests to different target groups based on the URLs in the requests. If the URL in a request matches a path pattern in a listener rule exactly, the request is routed using that rule; otherwise, the request is routed using the default rule.

A path pattern is case sensitive, can be up to 128 characters in length, and can contain any of the following characters. Note that you can include up to three wildcard characters in a path pattern.

- A-Z, a-z, 0-9
- _ - . $ / ~ " ' @ : +
- & (using &amp;)
- * (matches 0 or more characters)
- ? (matches exactly 1 character)

Example path patterns

- `/img/*`
- `/js/*`

Note that 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 would forward a request for `/img/picture.jpg` to the specified target group as a request for `/img/picture.jpg`.

Rule Priority

Each rule has a priority. Rules are evaluated in priority order, from the lowest value to the highest value. The default rule has the lowest priority. You can change the priorities of the nondefault rules at any time. For more information, see Update Listener Rules for Your Application Load Balancer (p. 24).

Create an HTTPS Listener for Your Application Load Balancer

You can create a listener that 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.

To use an HTTPS listener, you must deploy an SSL/TLS server certificate on your load balancer. The load balancer uses this certificate to terminate the connection and then decrypt requests from clients before sending them to the targets.

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.

SSL Certificates

The load balancer uses an X.509 certificate (SSL/TLS server certificate). This is a digital form of identification issued by a certificate authority (CA) and contains identification information, a validity period, a public key, a serial number, and the digital signature of the issuer.

You can create, import, and manage certificates 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.

Note
This option is available only in regions that support ACM. For more information, see AWS Certificate Manager Regions and Endpoints in the *AWS General Reference*.

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*.

Security Policies

You can choose the security policy that is used for front-end connections. The `ELBSecurityPolicy-2016-08` security policy is always used for back-end 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`
- `ELBSecurityPolicy-TLS-1-2-2017-01`
- `ELBSecurityPolicy-TLS-1-1-2017-01`
- `ELBSecurityPolicy-2015-05`

We recommend the `ELBSecurityPolicy-2016-08` policy for general use. You can use one of the `ELBSecurityPolicy-TLS` policies to meet compliance and security standards that require disabling certain TLS protocol versions. A small percentage of Internet clients use 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. 40).

The following table describes the security policies for Application Load Balancers.

Security Policy	**2016-08***	**TLS-1-1-2017-0**	**TLS-1-2-2017-01**
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	◆	◆	

* The `ELBSecurityPolicy-2016-08` and `ELBSecurityPolicy-2015-05` security policies for Application Load Balancers are identical.

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

Create an HTTPS Listener

You can add an HTTPS listener to an existing load balancer. When you create the listener, you must specify a certificate and a security policy.

To create 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 the load balancer.
4. On the **Listeners** tab, choose **Add listener**.
5. On the **Create Listener** page, do the following:

    a. For **Protocol**, select **HTTPS (Secure HTTP)**.

    b. (Optional) Modify **Port** and **Default target group** as needed.

    c. For **Select Certificate**, do one of the following:

    - If you have a certificate from AWS Certificate Manager, choose **Choose an existing certificate from AWS Certificate Manager (ACM)**, and then choose the certificate from **Certificate name**.
    - If you have already uploaded a certificate using IAM, choose **Choose an existing certificate from AWS Identity and Access Management (IAM)**, and then choose your certificate from **Certificate name**.
    - If you have a certificate ready to upload, choose **Upload a new SSL Certificate to AWS Identity and Access Management (IAM)**. For **Certificate name**, type a name for the certificate. For **Private Key**, copy and paste the contents of the private key file (PEM-encoded). In **Public Key Certificate**, copy and paste the contents of the public key certificate file (PEM-encoded). In **Certificate Chain**, copy and paste the contents of the certificate chain file (PEM-encoded), unless you are using a self-signed certificate and it's not important that browsers implicitly accept the certificate.

    d. For **Select Security Policy**, choose a security policy.

    e. Choose **Create**.

To create an HTTPS listener using the AWS CLI

Use the create-listener command.

Replace the SSL Certificate

Each certificate comes with a validity period. You must ensure that you renew or replace the certificate before its validity period ends.

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*.

To replace a certificate, you must first create a new certificate by following the same steps you used when you created the current certificate. Then, you can replace the certificate. Note that replacing a certificate does not affect requests that were received by a load balancer node and are pending routing to a healthy target, but the new certificate will be used with subsequent requests that are received.

To replace a 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.
4. On the **Listeners** tab, for the HTTPS listener, choose **Edit**.
5. For **Select Certificate**, do one of the following:
   - If you have a certificate from AWS Certificate Manager, choose **Choose an existing certificate from AWS Certificate Manager (ACM)**, and then choose the certificate from **Certificate name**.
   - If you have uploaded a certificate using IAM, choose **Choose an existing certificate from AWS Identity and Access Management (IAM)**, and then choose your certificate from **Certificate name**.
6. Choose **Save**.

To replace a certificate using the AWS CLI

Use the modify-listener 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. Note that Application Load Balancers do not support custom security policies.

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.
4. On the **Listeners** tab, for the HTTPS listener, choose **Edit**.
5. For **Select Security Policy**, choose a security policy.
6. Choose **Save**.

To update the security policy using the AWS CLI

Use the modify-listener command.

Update Listener Rules for Your Application Load Balancer

You define rules for your listeners that route requests to target groups. Each rule consists of a target group, priority, and an optional path for path-based routing. You define the default rule for a listener when you create the listener. For more information, see Listener Rules (p. 20).

To update the rules for your 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.
4. On the **Listener** tab, expand the section for the listener.
5. To add a rule, choose **Add rule**. Specify a target group and a priority for the rule. You can optionally specify a path pattern for path-routing routing. Choose **Save**.

6. To change the target group for a rule, choose **Edit**. Select a target group from **Target group name**, and then choose **Save**.
7. To change the path for path-based routing for a rule, choose **Edit**. Type the URI in **Path pattern**, and then choose **Save**. Note that you can't add a path pattern for the default rule.
8. To change the priority order of the rules, choose **Reorder rules**. Select a rule, and then use **Top** to give the rule the highest priority, **Up** to give it a higher priority than the previous rule, **Down** to give it a lower priority than the next rule, or **Bottom** to give it the second lowest priority. Note that the default rule always has the lowest priority. When you have finished, choose **Save**.
9. To delete a rule, choose **Delete**. When prompted for confirmation, choose **Yes, Delete**. Note that you can't delete the default rule for the listener.

To update the rules for your listener using the AWS CLI

Use the create-rule, modify-rule, and delete-rule commands.

Delete a Listener for your Application Load Balancer

Listeners are deleted when you delete the load balancer that they are attached to. Alternatively, you can delete listeners explicitly.

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 then choose the **Listeners** tab.
4. Choose **Delete** for the listener.
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

You register targets, such as EC2 instances, with a *target group*. To route requests to the targets in a target group, specify the target group in a rule for one of the listeners for your load balancer.

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

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, we use the security settings from the ELBSecurityPolicy2015-05 policy for these connections.

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. You can register the same EC2 instance with a target group multiple times using different ports, which enables the load balancer to route requests to ECS containers.

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.

When you plan to use your load balancer with an Auto Scaling group, you don't need to register your targets with a target group. After you attach a target group to an Auto Scaling group, Auto Scaling registers your targets with the target group when it launches them. For more information, see Attaching a Load Balancer to Your Auto Scaling Group in the *Auto Scaling User Guide*.

Target Group Attributes

The following are the target group attributes:

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

`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`.

Deregistration Delay

Elastic Load Balancing stops sending requests to instances that are deregistering. Connection draining ensures that in-flight requests complete before existing connections are closed. The initial state of a deregisterating target is `draining`. By default, the state of a deregistering target changes to `unused` after 300 seconds. To change the amount of time that Elastic Load Balancing waits before changing the state to `unused`, update the deregistration delay value.

To update the deregistration delay 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 **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.

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 and generates a cookie to include in the response to the client. The next request from that client 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.

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

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.

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. If you enable sticky sessions on multiple target groups, we recommend that you configure the same duration for all target groups.

To enable sticky sessions using the console

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

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.

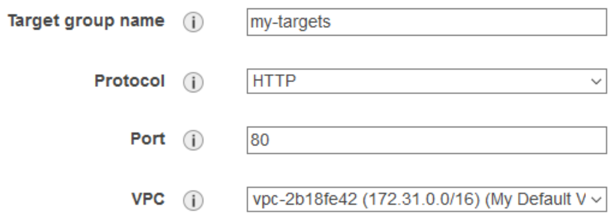
After you create a target group, you can add tags.

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. 20).

You can add or remove targets from your target group at any time. For more information, see Register Targets with Your Target Group (p. 32). 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. 32).

To create a target group using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Choose **Create target group**.
4. For **Target group name**, type a name for the target group.
5. (Optional) For **Protocol** and **Port**, modify the default values as needed.
6. For **VPC**, select a virtual private cloud (VPC). The targets for this target group must run in this VPC.



Target group name	my-targets
Protocol	HTTP
Port	80
VPC	vpc-2b18fe42 (172.31.0.0/16) (My Default V

7. (Optional) For **Health check settings** and **Advanced health check settings**, modify the default settings as needed.
8. Choose **Create**.
9. (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**.

To create a target group using the AWS CLI

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

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 group 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 no Availability Zone contains a healthy target, the load balancer nodes route requests to all targets.

Note that health checks do not support WebSockets.

Health Check Settings

You configure health checks for the targets in a target group using the following settings. The load balancer sends a health check request to each registered target every **HealthCheckIntervalSeconds** seconds, using the specified port, protocol, and ping path. It waits for the target to respond within the response timeout period. If the health checks exceed the threshold for consecutive failed responses, the load balancer takes the target out of service. When the health checks exceed the threshold for consecutive successful responses, 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 ping path that is the destination on the targets for health checks. The default is /.
HealthCheckTimeoutSeconds	The amount of time, in seconds, during which no response from a target means a failed health check. The range is 2 to 60 seconds. The default is 5 seconds.
HealthCheckIntervalSeconds	The approximate amount of time, in seconds, between health checks of an individual target. The range is 5 to 300 seconds. The default is 30 seconds.
HealthyThresholdCount	The number of consecutive successful health checks required before considering an unhealthy target healthy. The range is 2 to 10. The default is 5.
UnhealthyThresholdCount	The number of consecutive failed health checks required before considering a target unhealthy. The range is 2 to 10. The default is 2.

Setting	Description
Matcher	The HTTP codes to use when checking for a successful response from a target. You can specify values or ranges of values between 200 and 499. The default value is 200.

Target Health Status

Before the load balancer sends a health check request to a target, you must register it with a target group, specify its target group in a listener rule, and ensure that the Availability Zone of the target is enabled for the load balancer. Before a target can receive requests from the load balancer, it must pass the initial health checks. After a target passes the initial health checks, its status is `Healthy`.

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

Value	Description
`initial`	The load balancer is in the process of registering the target or performing the initial health checks on the target.
`healthy`	The target is healthy.
`unhealthy`	The target did not respond to a health check or failed the health check.
`unused`	The target is not registered with a target group, the target group is not used in a listener rule for the load balancer, or the target is in an Availability Zone that is not enabled for the load balancer.
`draining`	The target is deregistering and connection draining is in process.

Health Check Reason Codes

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

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

Reason code	Description
	Target is in an invalid state
`Target.NotInUse`	Target group is not configured to receive traffic from the load balancer Target is in an Availability Zone that is not enabled for the load balancer
`Target.NotRegistered`	Target is not registered to the target group
`Target.ResponseCodeMismatch`	Health checks failed with these codes: [*code*]
`Target.Timeout`	Request timed out

Check the Health of Your Targets

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

To check the health of your targets using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Select the target group.
4. On the **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, and it includes a reason code if the status is any value other than `Healthy`.

Modify the Health Check Settings of a Target Group

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

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

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Select the target group.
4. On the **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. For more information, see Target Groups for Your Application Load Balancers (p. 26).

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, Elastic Load Balancing waits until in-flight requests have completed. This is known as *connection draining*. The status of a target is `draining` while connection draining is in progress.

If your load balancer is attached to an Auto Scaling group and the group scales out, the targets added to the Auto Scaling group are automatically registered with the target group configured for the Auto Scaling group. If you detach the load balancer from the Auto Scaling group, the targets in the Auto Scaling group are automatically deregistered from the target group. For more information, see Attaching a Load Balancer to Your Auto Scaling Group in the *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

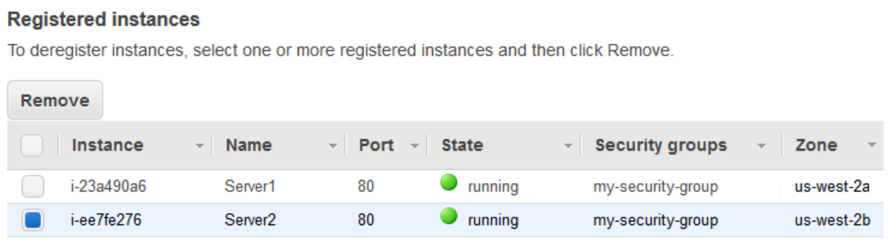
Inbound		
Source	**Port Range**	**Comment**
load balancer security group	*instance listener*	Allow traffic from the load balancer on the instance listener port
load balancer security group	*health check*	Allow traffic from the load balancer on the health check port

Register or Deregister Targets

A target must be in the virtual private cloud (VPC) that you specified for the target group. If the target is an EC2 instance, it must be in the `running` state when you register it.

To register or deregister targets using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Select your target group.
4. On the **Targets** tab, choose **Edit**.
5. (Optional) For **Registered instances**, select any instances to be deregistered and choose **Remove**.

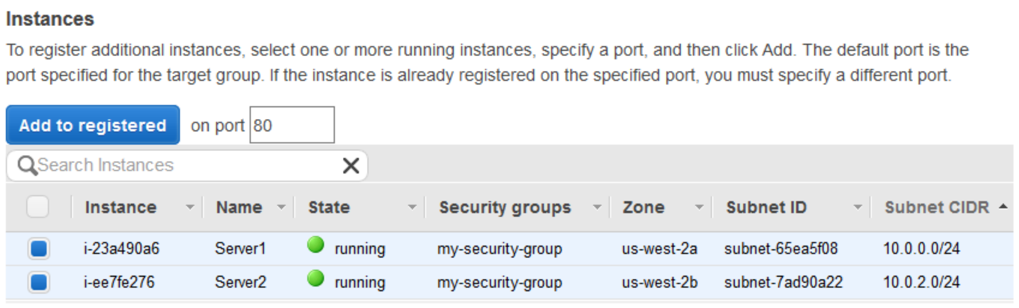


Registered instances

To deregister instances, select one or more registered instances and then click Remove.

	Instance	Name	Port	State	Security groups	Zone
☐	i-23a490a6	Server1	80	● running	my-security-group	us-west-2a
■	i-ee7fe276	Server2	80	● running	my-security-group	us-west-2b

6. (Optional) For **Instances**, select any running instances to be registered, modify the default instance port as needed, and then choose **Add to registered**.



Instances

To register additional instances, select one or more running instances, specify a port, and then click Add. The default port is the port specified for the target group. If the instance is already registered on the specified port, you must specify a different port.

	Instance	Name	State	Security groups	Zone	Subnet ID	Subnet CIDR
■	i-23a490a6	Server1	● running	my-security-group	us-west-2a	subnet-65ea5f08	10.0.0.0/24
■	i-ee7fe276	Server2	● running	my-security-group	us-west-2b	subnet-7ad90a22	10.0.2.0/24

7. Choose **Save**.

To register or deregister targets using the AWS CLI

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

Delete Your Target Group

You can delete a target group if it is not referenced by any actions. Note that deleting a target group does not affect the targets registered with the target group. If you no longer need the EC2 instances, you can stop or terminate them.

To delete a target group using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Select the target group, and then 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. 35).

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. 40).

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. 48).

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 AWS CloudTrail Logging for Your Application Load Balancer (p. 50).

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

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

Metric	Description
`ActiveConnectionCount`	The total number of concurrent TCP connections active from clients to the load balancer and from the load balancer to targets. **Statistics**: The most useful statistic is `Sum`.
`ClientTLSNegotiationErrorCount`	The number of TLS connections initiated by the client that did not establish a session with the load balancer. Possible causes include a mismatch of ciphers or protocols. **Statistics**: The most useful statistic is `Sum`.
`HealthyHostCount`	The number of targets that are considered healthy. **Statistics**: The most useful statistics are `Average`, `Minimum`, and `Maximum`.
`HTTPCode_ELB_4XX_Count`	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 have not been received by the target. This count does not include any response codes generated by the targets. **Statistics**: The most useful statistic is `Sum`. Note that `Minimum`, `Maximum`, and `Average` all return 1.
`HTTPCode_ELB_5XX_Count`	The number of HTTP 5XX server error codes that originate from the load balancer. This count does not include any response codes generated by the targets. **Statistics**: The most useful statistic is `Sum`. Note that `Minimum`, `Maximum`, and `Average` all return 1.
`HTTPCode_Target_2XX_Count`, `HTTPCode_Target_3XX_Count`, `HTTPCode_Target_4XX_Count`, `HTTPCode_Target_5XX_Count`	The number of HTTP response codes generated by the targets. This does not include any response codes generated by the load balancer. **Statistics**: The most useful statistic is `Sum`. Note that `Minimum`, `Maximum`, and `Average` all return 1.

Metric	Description
`IPv6ProcessedBytes`	The total number of bytes processed by the load balancer over IPv6.
`IPv6RequestCount`	The number of IPv6 requests received by the load balancer. **Statistics**: The most useful statistic is `Sum`. Note that `Minimum`, `Maximum`, and `Average` all return 1.
`NewConnectionCount`	The total number of new TCP connections established from clients to the load balancer and from the load balancer to targets. **Statistics**: The most useful statistic is `Sum`.
`ProcessedBytes`	The total number of bytes processed by the load balancer over IPv4 and IPv6.
`RejectedConnectionCount`	The number of connections that were rejected because the load balancer had reached its maximum number of connections. **Statistics**: The most useful statistic is `Sum`.
`RequestCount`	The number of requests received by the load balancer. This includes requests over IPv4 and IPv6. **Statistics**: The most useful statistic is `Sum`. Note that `Minimum`, `Maximum`, and `Average` all return 1.
`TargetConnectionErrorCount`	The number of connections that were not successfully established between the load balancer and target. **Statistics**: The most useful statistic is `Sum`.
`TargetResponseTime`	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 `target_processing_time` field in the access logs. **Statistics**: The most useful statistics are `Average` and `pNN.NN` (percentiles).
`TargetTLSNegotiationErrorCount`	The number of TLS connections initiated by the load balancer that did not establish a session with the target. Possible causes include a mismatch of ciphers or protocols. **Statistics**: The most useful statistic is `Sum`.
`UnHealthyHostCount`	The number of targets that are considered unhealthy. **Statistics**: The most useful statistics are `Average`, `Minimum`, and `Maximum`.

Metric Dimensions for Application Load Balancers

To filter the metrics for your Application Load Balancer, use the following dimensions.

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

Statistics for Application Load Balancer Metrics

CloudWatch provides statistics based on the metric data points published by 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. The following metrics are available:

      - Healthy Hosts — `HealthyHostCount`
      - Unhealthy Hosts — `UnHealthyHostCount`
      - Average Latency — `TargetResponseTime`
      - Sum Requests — `RequestCount`
      - Backend Connection Errors — `TargetConnectionErrorCount`
      - Target TLS Negotiation Errors — `TargetTLSNegotiationErrorCount`

- Sum HTTP 2XXs — `HTTPCode_Target_2XX`
- Sum HTTP 3XXs — `HTTPCode_Target_3XX`
- Sum HTTP 4XXs — `HTTPCode_Target_4XX`
- Sum HTTP 5XXs — `HTTPCode_Target_5XX`

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. The following metrics are available:

      - Average Latency — `TargetResponseTime`
      - Sum Requests — `RequestCount`
      - New Connection Count — `NewConnectionCount`
      - Active Connection Count — `ActiveConnectionCount`
      - Processed Bytes — `ProcessedBytes`
      - Target connection errors — `TargetConnectionErrorCount`
      - Sum rejected connections — `RejectedConnectionCount`
      - Client TLS Negotiation Errors — `ClientTLSNegotiationErrorCount`
      - Target TLS Negotiation Errors — `TargetTLSNegotiationErrorCount`
      - Sum HTTP 2XXs — `HTTPCode_Target_2XX_Count`
      - Sum HTTP 3XXs — `HTTPCode_Target_3XX_Count`
      - Sum HTTP 4XXs — `HTTPCode_Target_4XX_Count`
      - Sum HTTP 5XXs — `HTTPCode_Target_5XX_Count`
      - Sum ELB HTTP 4XXs — `HTTPCode_ELB_4XX_Count`
      - Sum ELB HTTP 5XXs — `HTTPCode_ELB_5XX_Count`

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, type 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, type 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, type 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, type its name in the search field. To view the metrics for a single Availability Zone, type 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, type its name in the search field. To view the metrics for a single target group, type its name in the search field. To view the metrics for a single Availability Zone, type its name in the search field.

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. Note that CloudWatch treats each unique combination of dimensions as a separate metric. You can't retrieve statistics using combinations of dimensions that were not specially published. You must specify the same dimensions that were used when the metrics were created.

```
aws cloudwatch get-metric-statistics --namespace AWS/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 or connections sent to your load balancer. Each log contains information such as the time it was received, the client's IP address, latencies, request paths, and server responses. You can use these access logs to analyze traffic patterns and to 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.

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 Entries (p. 42)
- Bucket Permissions (p. 45)
- Enable Access Logging (p. 47)
- Disable Access Logging (p. 48)
- Processing Access Log Files (p. 48)

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-
west-2/2016/05/01/123456789012_elasticloadbalancing_us-west-2_my-
loadbalancer_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. Note that Elastic Load Balancing does not log health check requests.

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.

Syntax

Each log entry contains the details of a single request (or connection in the case of WebSockets) made to the load balancer. All fields in a log entry are delimited by spaces. Each entry in a log file has the following format:

```
type timestamp elb client:port target:port request_processing_time
 target_processing_time response_processing_time elb_status_code
 target_status_code received_bytes sent_bytes "request" "user_agent"
 ssl_cipher ssl_protocol target_group_arn trace_id
```

You should ignore any fields at the end of the log entry that you were not expecting.

The following table describes the fields of an access log entry.

Field	Description
type	The type of request or connection. The possible values are as follows (ignore any other values): • `http` — HTTP • `https` — HTTP over SSL/TLS • `h2` — HTTP/2 over SSL/TLS • `ws` — WebSockets • `wss` — WebSockets over SSL/TLS
timestamp	The time when the load balancer received the request from 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. 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 `-`.
request_processing_time	The total time elapsed, in seconds, from the time the load balancer received the request until the time it sent it to a target.

Field	Description
	This value is set to `-1` if the load balancer can't dispatch the request to a target. This can happen if the target closes the connection before the idle timeout or if the client sends a malformed request.
target_processing_time	The total time elapsed, in seconds, from the time the load balancer sent the request to a target until the target started to send the response headers. This value is set to `-1` if the load balancer can't dispatch the request to a target. This can happen if the target closes the connection before the idle timeout or if the client sends a malformed request.
response_processing_time	The total time elapsed (in seconds) 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 target. 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.
elb_status_code	The status code of the response from the load balancer.
target_status_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, the value is set to `-`.
received_bytes	The size of the request, in bytes, received from the client (requester). For HTTP requests, this includes the headers. For WebSockets, this is the total number of bytes received from the client on the connection.
sent_bytes	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.
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.
user_agent	A User-Agent string that identifies the client that originated the request. 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 recorded only if the incoming connection was established after a successful negotiation. Otherwise, the value is set to `-`.
ssl_protocol	[HTTPS listener] The SSL protocol. This value is recorded only if the incoming connection was established after a successful negotiation. Otherwise, the value is set to `-`.
target_group_arn	The Amazon Resource Name (ARN) of the target group.
trace_id	The contents of the **X-Amzn-Trace-Id** header.

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 2016-08-10T22:08:42.945958Z app/my-loadbalancer/50dc6c495c0c9188
192.168.131.39:2817 10.0.0.1:80 0.0000 0.001 0.0000 200 200 34 366
"GET http://www.example.com:80/ HTTP/1.1" "curl/7.46.0" - -
arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/my-
targets/73e2d6bc24d8a067
"Root=1-58337262-36d228ad5d99923122bbe354"
```

Example HTTPS Entry

The following is an example log entry for an HTTPS listener (port 443 to port 80):

```
https 2016-08-10T23:39:43.065466Z app/my-loadbalancer/50dc6c495c0c9188
192.168.131.39:2817 10.0.0.1:80 0.000086 0.001048 0.001337 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-west-2:123456789012:targetgroup/my-
targets/73e2d6bc24d8a067
"Root=1-58337281-1d84f3d73c47ec4e58577259"
```

Example HTTP/2 Entry

The following is an example log entry for an HTTP/2 stream.

```
h2 2016-08-10T00:10:33.145057Z 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-west-2:123456789012:targetgroup/my-
targets/73e2d6bc24d8a067
"Root=1-58337327-72bd00b0343d75b906739c42"
```

Example WebSockets Entry

The following is an example log entry for a WebSockets connection.

```
ws 2016-08-10T00:32:08.923954Z 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-west-2:123456789012:targetgroup/my-
targets/73e2d6bc24d8a067
"Root=1-58337364-23a8c76965a2ef7629b185e3"
```

Example Secured WebSockets Entry

The following is an example log entry for a secured WebSockets connection.

```
wss 2016-08-10T00:42:46.423695Z 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"
```

Bucket Permissions

When you enable access logging, you must specify an S3 bucket for the access logs. This bucket must be located in the same region as the load balancer, and 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.

> **Important**
> If you will use the console to enable access logging, you can skip to Enable Access Logging (p. 47). If you will use the AWS CLI or an API to enable access logging, the bucket must exist and must have the required bucket policy.

If you need to create a bucket for your access logs, use the following procedure to create the bucket and add the required bucket policy. If you already have a bucket, start at step 4 to add or update the bucket policy for your bucket.

To create an Amazon S3 bucket with the required permissions

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose **Create Bucket**.
3. In the **Create a Bucket** dialog box, do the following:
    a. For **Bucket Name**, enter a name for your bucket (for example, `my-loadbalancer-logs`). 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 and choose **Properties**.
5. For **Permissions**, choose **Add bucket policy**. If your bucket already has an attached policy, choose **Edit bucket policy**. You can add the required statement to the existing policy.
6. In the **Bucket Policy Editor** dialog box, choose **AWS Policy Generator**.
7. In the **AWS Policy Generator** dialog box, do the following:
    a. For **Select Type of Policy**, choose **S3 Bucket Policy**.
    b. For **Effect**, choose **Allow**.
    c. For **Principal**, specify one of the following AWS account IDs to grant Elastic Load Balancing access to the S3 bucket. Use the account ID that corresponds to the region for your load balancer and bucket.

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

Region	Region Name	Elastic Load Balancing Account ID
`us-west-2`	US West (Oregon)	797873946194
`ca-central-1`	Canada (Central)	985666609251
`eu-west-1`	EU (Ireland)	156460612806
`eu-central-1`	EU (Frankfurt)	054676820928
`eu-west-2`	EU (London)	652711504416
`ap-northeast-1`	Asia Pacific (Tokyo)	582318560864
`ap-northeast-2`	Asia Pacific (Seoul)	600734575887
`ap-southeast-1`	Asia Pacific (Singapore)	114774131450
`ap-southeast-2`	Asia Pacific (Sydney)	783225319266
`ap-south-1`	Asia Pacific (Mumbai)	718504428378
`sa-east-1`	South America (São Paulo)	507241528517
`us-gov-west-1`*	AWS GovCloud (US)	048591011584
`cn-north-1`*	China (Beijing)	638102146993

* These regions requires a separate account. For more information, see AWS GovCloud (US) and China (Beijing).

d. For **Actions**, choose `PutObject` to allow Elastic Load Balancing to store objects in the S3 bucket.

e. For **Amazon Resource Name (ARN)**, enter the ARN of your S3 bucket in the following format. For *`aws-account-id`*, specify the ID of the AWS account that owns the load balancer (for example, *`123456789012`*).

```
arn:aws:s3:::bucket/prefix/AWSLogs/aws-account-id/*
```

Note that if you are using the `us-gov-west-1` region, specify `arn:aws-us-gov` instead of `arn:aws` in the ARN.

f. Choose **Add Statement**, **Generate Policy**. The policy document should be similar to the following:

```
{
  "Id": "Policy1429136655940",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1429136633762",
      "Action": [
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::my-loadbalancer-logs/my-app/
AWSLogs/123456789012/*",
      "Principal": {
        "AWS": [
          "797873946194"
        ]
      }
    }
  ]
}
```

   g. If you are creating a new bucket policy, copy the entire policy document, and then choose **Close**. Go back to the **Bucket Policy Editor** dialog box, paste the policy into the text area, and then choose **Save**.

      If you are editing an existing bucket policy, copy the new statement from the policy document (the text between the [and] of the `Statement` element), and then choose **Close**. Go back to the **Bucket Policy Editor** dialog box, paste the statement in the `Statement` element (making sure to separate the new statement from the existing statements using a comma), and then choose **Save**.

8. Under **Permissions**, 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. Choose **Enable access logs**.
   b. For **S3 location**, type the name of your S3 bucket, including any prefix (for example, `my-loadbalancer-logs/my-app`). You can specify the name of an existing bucket or a name for a new bucket.
   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. Note that 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 test log file. The path should be as follows:

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

Disable Access Logging

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

To disable access logging using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Load Balancers**.
3. Select your load balancer.
4. On the **Description** tab, choose **Edit attributes**.
5. On the **Edit load balancer attributes** page, clear **Enable access logs**.
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 EMR enables you to quickly and efficiently process vast amounts of data. For more information, see the Amazon EMR Developer Guide.
- 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. 40).

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.

AWS CloudTrail Logging for Your Application Load Balancer

Elastic Load Balancing is integrated with AWS CloudTrail, which captures API calls to AWS made by or on behalf of your AWS account, and delivers log files to an Amazon S3 bucket that you specify. There is no cost to use CloudTrail. However, the standard rates for Amazon S3 apply.

CloudTrail logs calls to the AWS APIs, including the Elastic Load Balancing API, whether you use them directly or indirectly through the AWS Management Console. You can use the information collected by CloudTrail to determine what API call was made, what source IP address was used, who made the call, when it was made, and so on.

To learn more about CloudTrail, including how to configure and enable it, see the AWS CloudTrail User Guide. For the complete list of Elastic Load Balancing API actions, see the Elastic Load Balancing API Reference version 2015-12-01.

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. 40).

Contents

Enable CloudTrail Event Logging

If you haven't done so already, use the following steps to enable CloudTrail event logging for your account.

To enable CloudTrail event logging

1. Open the CloudTrail console at https://console.aws.amazon.com/cloudtrail/.
2. Choose **Get Started Now**.
3. For **Trail name**, type a name for your trail.
4. Leave **Apply trail to all regions** as **Yes**.
5. Choose an existing S3 bucket for your CloudTrail log files, or create a new one. To create a new bucket, type a unique name for **S3 bucket**. To use an existing bucket, change **Create a new S3 bucket** to **No** and then select your bucket from **S3 bucket**.
6. Choose **Turn on**.

The log files are written to your S3 bucket in the following location:

```
my-bucket/AWSLogs/123456789012/CloudTrail/region/yyyy/mm/dd/
```

For more information, see the AWS CloudTrail User Guide.

Elastic Load Balancing Event Records in CloudTrail Log Files

The log files from CloudTrail contain event information in JSON format. An event record represents a single AWS API call and includes information about the requested action, such as the user that requested the action, the date and the time of the request, the request parameter, and the response elements.

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

The following example shows CloudTrail log records for a user who created a 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. For more information about the different elements and values in a CloudTrail log file, see CloudTrail Event Reference in the *AWS CloudTrail User Guide*.

```
{
   "Records": [
   . . .
   {
      "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": "Z2P70J7HTTTPLU",
            "createdTime": "Apr 11, 2016 5:23:50 PM",
            "loadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/ffcddace1759e1d0",
            "scheme": "internet-facing"
         }]
      },
      "requestID": "b9960276-b9b2-11e3-8a13-f1ef1EXAMPLE",
      "eventID": "6f4ab5bd-2daa-4d00-be14-d92efEXAMPLE",
      "eventType": "AwsApiCall",
      "apiVersion": "2015-12-01",
      "recipientAccountId": "123456789012"
   },
   . . .
   {
      "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"
   },
   . . .
]}
```

You can also use one of the Amazon partner solutions that integrate with CloudTrail to read and analyze your CloudTrail log files. For more information, see the AWS CloudTrail Partners page.

Troubleshoot Your Application Load Balancers

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

Issues

A registered target is not in service

If a target is taking longer than expected to enter the `InService` state, it might be failing health checks. Your target is not in service until it passes one health check. Verify that your instance is failing health checks and then check for the following:

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.

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 values when you configure health checks. Confirm the success codes that the load balancer is expecting and that your application is configured to return these success codes on success.

For more information, see Health Checks for Your Target Groups (p. 30).

Clients cannot connect to an Internet-facing load balancer

If the load balancer is not responding to requests, check for the following:

Your Internet-facing load balancer is attached to a private subnet
Verify that you specified public subnets for your load balancer. A public subnet has a route to the Internet Gateway for your 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.

HTTP 460

Indicates that the load balancer received a request from a client, but the client closed the connection with the load balancer before the idle timeout expired. The load balancer saves the request to the access log and increments the `HTTPCode_ELB_4XX_Count` metric.

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, or increase the client timeout period to match the load balancer idle timeout, if the client supports this.

HTTP 502: Bad Gateway

Indicates that the load balancer failed to establish a connection to a target or the target closed the connection while the load balancer has an outstanding request to the target. The load balancer sends the HTTP code to the client, saves the request to the access log, and increments the `HTTPCode_ELB_5XX_Count` metric.

HTTP 503: Service Unavailable

Indicates that the target group has no healthy registered targets. The load balancer sends the HTTP code to the client, saves the request to the access log, and increments the `HTTPCode_ELB_5XX_Count` metric.

Check whether there are healthy registered targets using the `HealthyHostCount` metric. Add targets so that each enabled Availability Zone has at least one healthy target. This ensures the best performance from your load balancer.

Limits for Your Application Load Balancer

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

Regional Limits *

- Load balancers per region: 20 ******
- Target groups per region: 200

Load Balancer Limits

- Listeners per load balancer: 10
- Targets per load balancer: 1000
- Subnets per Availability Zone per load balancer: 1
- Security groups per load balancer: 5
- Rules per load balancer (not counting default rules): 10
- Number of times a target can be registered per load balancer: 100

Listener Limits

- Certificates per listener: 1

Target Group Limits

- Load balancers per target group: 1
- Targets per target group: 1000

Rule Limits

- Conditions per rule: 1
- Actions per rule: 1
- Target groups per action: 1

***** If needed, you can request an increase in the number of load balancers for a region or the number of target groups for a region using the Elastic Load Balancing Limits form.

****** This limit includes both your Application Load Balancers and your Classic Load Balancers.

Document History for Application Load Balancers

The following table describes important additions to the documentation for Application Load Balancers.

- **API version:** 2015-12-01

Change	Description	Date
Security policies for TLS 1.1 and TLS 1.2	This release adds support for security policies for TLS 1.1 and TLS 1.2. For more information, see Security Policies (p. 21).	February 6, 2017
IPv6 support	This release adds support for IPv6 addresses. For more information, see IP Address Type (p. 13).	January 25, 2017
Request tracing	This release adds support for request tracing. For more information, see Request Tracing for Your Application Load Balancer (p. 48).	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. 38).	November 17, 2016
New load balancer type	This release of Elastic Load Balancing introduces Application Load Balancers.	August 11, 2016