**Elastic Load Balancer**

**Related Services**

Elastic Load Balancing works with the following services to improve the availability and scalability of your applications.

* **Amazon EC2**
* **Amazon EC2 Auto Scaling**
* **AWS Certificate Manager** — When you create an HTTPS listener, you can specify certificates provided by ACM. The load balancer uses certificates to terminate connections and decrypt requests from clients.
* **Amazon CloudWatch**
* **Amazon ECS**
* **Route 53**
* **AWS WAF**

Elastic Load Balancing supports three types of load balancers: Application Load Balancers, Network Load Balancers, and Classic Load Balancers. There is a key difference between the way you configure these load balancers. With Application Load Balancers and Network Load Balancers, you register targets in target groups, and route traffic to the target groups. With Classic Load Balancers, you register instances with the load balancer.

## Availability Zones and Load Balancer Nodes

When you enable an Availability Zone for your load balancer, Elastic Load Balancing creates a load balancer node in the Availability Zone. If you register targets in an Availability Zone but do not enable the Availability Zone, these registered targets do not receive traffic. Note that your load balancer is most effective if you ensure that each enabled Availability Zone has at least one registered target.

We recommend that you enable multiple Availability Zones. (Note that with an Application Load Balancer, we require you to enable multiple Availability Zones.) With this configuration, if one Availability Zone becomes unavailable or has no healthy targets, the load balancer can continue to route traffic to the healthy targets in another Availability Zone.

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 traffic to them.

### Cross-Zone Load Balancing

The nodes for your load balancer distribute requests from clients to registered targets. When cross-zone load balancing is enabled, each load balancer node distributes traffic across the registered targets in all enabled Availability Zones. When cross-zone load balancing is disabled, each load balancer node distributes traffic across the registered targets in its Availability Zone only.

The following diagrams demonstrate the effect of cross-zone load balancing. There are two enabled Availability Zones, with 2 targets in Availability Zone A and 8 targets in Availability Zone B. Clients send requests, and Amazon Route 53 responds to each request with the IP address of one of the load balancer nodes. This distributes traffic such that each load balancer node receives 50% of the traffic from the clients. Each load balancer node distributes its share of the traffic across the registered targets in its scope.

If cross-zone load balancing is enabled, each of the 10 targets receives 10% of the traffic. This is because each load balancer node can route its 50% of the client traffic to all 10 targets.


                    When cross-zone load balancing is enabled
                

If cross-zone load balancing is disabled, each of the 2 targets in Availability Zone A receives 25% of the traffic and each of the 8 targets in Availability Zone B receives 6.25% of the traffic. This is because each load balancer node can route its 50% of the client traffic only to targets in its Availability Zone.


                    When cross-zone load balancing is disabled
                

With Application Load Balancers, cross-zone load balancing is always enabled.

With Network Load Balancers, cross-zone load balancing is disabled by default. After you create a Network Load Balancer, you can enable or disable cross-zone load balancing at any time. For more information, see [Cross-Zone Load Balancing](https://docs.aws.amazon.com/elasticloadbalancing/latest/network/network-load-balancers.html#cross-zone-load-balancing) in the User Guide for Network Load Balancers.

When you create a Classic Load Balancer, the default for cross-zone load balancing depends on how you create the load balancer. With the API or CLI, cross-zone load balancing is disabled by default. With the AWS Management Console, the option to enable cross-zone load balancing is selected by default. After you create a Classic Load Balancer, you can enable or disable cross-zone load balancing at any time. For more information, see [Enable Cross-Zone Load Balancing](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/enable-disable-crosszone-lb.html#enable-cross-zone) in the User Guide for Classic Load Balancers.

ad Balancers.

## Request Routing

Before a client sends a request to your load balancer, it resolves the load balancer's domain name using a Domain Name System (DNS) server. The DNS entry is controlled by Amazon, because your load balancers are in the amazonaws.com domain. The Amazon DNS servers return one or more IP addresses to the client, which are the IP addresses of the load balancer nodes for your load balancer. With Network Load Balancers, Elastic Load Balancing creates a network interface for each Availability Zone you enable. Each load balancer node in the Availability Zone uses this network interface to get a static IP address. You can optionally associate one Elastic IP address with each network interface when you create the load balancer.

As traffic to your application changes over time, Elastic Load Balancing scales your load balancer and updates the DNS entry. Note that the DNS entry also specifies the time-to-live (TTL) as 60 seconds, which ensures that the IP addresses can be remapped quickly in response to changing traffic.

The client determines which IP address to use to send requests to the load balancer. The load balancer node that receives the request selects a healthy registered target and sends the request to the target using its private IP address.

### Routing Algorithm

With Application Load Balancers 🡪 round robin routing algorithm.

With Network Load Balancers, 🡪 flow hash algorithm,

With Classic Load Balancers, 🡪 round robin routing algorithm for TCP listeners

and the least outstanding requests routing algorithm for HTTP and HTTPS listeners.

## Load Balancer Scheme

When you create a load balancer, you must choose whether to make it an internal load balancer or an Internet-facing load balancer. Note that when you create a Classic Load Balancer in EC2-Classic, it must be an Internet-facing load balancer.

The nodes of an Internet-facing load balancer have public IP addresses. The DNS name of an Internet-facing load balancer is publicly resolvable to the public IP addresses of the nodes. Therefore, Internet-facing load balancers can route requests from clients over the Internet.

The nodes of an internal load balancer have only private IP addresses. The DNS name of an internal load balancer is publicly resolvable to the private IP addresses of the nodes. Therefore, internal load balancers can only route requests from clients with access to the VPC for the load balancer.

Note that both Internet-facing and internal load balancers route requests to your targets using private IP addresses. Therefore, your targets do not need public IP addresses to receive requests from an internal or an Internet-facing load balancer

# **Classic Load Balancer**

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

A listener checks for connection requests from clients, using the protocol and port that you configure, and forwards requests to one or more registered instances using the protocol and port number that you configure. You add one or more listeners to your load balancer.

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


                    A load balancer routes traffic from clients to your EC2 instances.
                

To ensure that your registered instances are able to handle the request load in each Availability Zone, it is important to keep approximately the same number of instances in each Availability Zone registered with the load balancer. For example, if you have ten instances in Availability Zone us-west-2a and two instances in us-west-2b, the requests are distributed evenly between the two Availability Zones. As a result, the two instances in us-west-2b serve the same amount of traffic as the ten instances in us-west-2a. Instead, you should have six instances in each Availability Zone.

By default, the load balancer distributes traffic evenly across the Availability Zones that you enable for your load balancer. To distribute traffic evenly across all registered instances in all enabled Availability Zones, enable cross-zone load balancing on your load balancer. However, we still recommend that you maintain approximately equivalent numbers of instances in each Availability Zone for better fault tolerance.

## Benefits

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

* Support for EC2-Classic
* Support for TCP and SSL listeners
* Support for sticky sessions using application-generated cookies

# Internet-Facing Classic Load Balancers

An Internet-facing load balancer has a publicly resolvable DNS name, so it can route requests from clients over the Internet to the EC2 instances that are registered with the load balancer.


                An Internet-facing load balancer routes traffic from the Internet to your EC2 instances.
            

If a load balancer is in a VPC with ClassicLink enabled, its instances can be linked EC2-Classic instances. If a load balancer is in EC2-Classic, its instances must be in EC2-Classic.

## Public DNS Names for Your Load Balancer

When your load balancer is created, it receives a public DNS name that clients can use to send requests. The DNS servers resolve the DNS name of your load balancer to the public IP addresses of the load balancer nodes for your load balancer. Each load balancer node is connected to the back-end instances using private IP addresses.

|  |
| --- |
|  |

# Internal Classic Load Balancers

When you create a load balancer in a VPC, you must choose whether to make it an internal load balancer or an Internet-facing load balancer.

The nodes of an Internet-facing load balancer have public IP addresses. The DNS name of an Internet-facing load balancer is publicly resolvable to the public IP addresses of the nodes. Therefore, Internet-facing load balancers can route requests from clients over the Internet.

The nodes of an internal load balancer have only private IP addresses. The DNS name of an internal load balancer is publicly resolvable to the private IP addresses of the nodes. Therefore, internal load balancers can only route requests from clients with access to the VPC for the load balancer.


                An internal load balancer routes traffic to your EC2 instances in private subnets.
            

# Registered Instances for Your Classic Load Balancer

After you've created your Classic Load Balancer, you must register your EC2 instances with the load balancer. You can select EC2 instances from a single Availability Zone or multiple Availability Zones within the same region as the load balancer. Elastic Load Balancing routinely performs health checks on registered EC2 instances, and automatically distributes incoming requests to the DNS name of your load balancer across the registered, healthy EC2 instances.

## Best Practices for Your Instances

* Install a web server, such as Apache or Internet Information Services (IIS), on all instances that you plan to register with your load balancer.
* For HTTP and HTTPS listeners, we recommend that you enable the keep-alive option in your EC2 instances, which enables the load balancer to re-use the connections to your instances for multiple client requests. This reduces the load on your web server and improves the throughput of the load balancer. The keep-alive timeout should be at least 60 seconds
* Elastic Load Balancing supports Path Maximum Transmission Unit (MTU) Discovery. To ensure that Path MTU Discovery can function correctly, you must ensure that the security group for your instance allows ICMP fragmentation required (type 3, code 4) messages.

## Prepare Your VPC and EC2 Instances

**Load Balancers in a VPC**

Amazon Virtual Private Cloud (Amazon VPC) enables you to define a virtual networking environment in a private, isolated section of the AWS cloud. Within this virtual private cloud (VPC), you can launch AWS resources such as load balancers and EC2 instances.

**Subnets for Your Load Balancer**

To ensure that your load balancer can scale properly, verify that each subnet for your load balancer has a CIDR block with at least a /27 bitmask (for example, 10.0.0.0/27) and has at least 8 free IP addresses. Your load balancer uses these IP addresses to establish connections with the instances.

Create a subnet in each Availability Zone where you want to launch instances. Depending on your application, you can launch your instances in public subnets, private subnets, or a combination of public and private subnets. When you create a load balancer, you must add one or more public subnets to the load balancer. If your instances are in private subnets, create public subnets in the same Availability Zones as the subnets with your instances; you will add these public subnets to the load balancer.

**Security Groups**

You must ensure that the load balancer can communicate with your instances on both the listener port and the health check port..

**Network ACLs**

The network ACLs for your VPC must allow traffic in both directions on the listener port and the health check port.

**ClassicLink**

ClassicLink enables your EC2-Classic instances to communicate with VPC instances using private IP addresses, provided that the VPC security groups allow it. If you plan to register linked EC2-Classic instances with your load balancer, you must enable ClassicLink for your VPC, and then create your load balancer in the ClassicLink-enabled VPC.

|  |
| --- |
|  |

# Configure Health Checks for Your Classic Load Balancer

To discover the availability of your EC2 instances, a load balancer periodically sends pings, attempts connections, or sends requests to test the EC2 instances. These tests are called health checks. The status of the instances that are healthy at the time of the health check is InService. The status of any instances that are unhealthy at the time of the health check is OutOfService. The load balancer performs health checks on all registered instances, whether the instance is in a healthy state or an unhealthy state.

## Health Check Configuration

A health configuration contains the information that a load balancer uses to determine the health state of the registered instances. The following table describes the health check configuration fields.

|  |  |
| --- | --- |
| **Field** | **Description** |
| Ping Protocol |  |
| Ping Port |  |
| Ping Path | Default: /index.html |
| Response Timeout | The amount of time to wait when receiving a response from the health check, in seconds.  Valid values: 2 to 60  Default: 5 |
| HealthCheck Interval | The amount of time between health checks of an individual instance, in seconds.  Valid values: 5 to 300  Default: 30 |
| Unhealthy Threshold | The number of consecutive failed health checks that must occur before declaring an EC2 instance unhealthy.  Valid values: 2 to 10  Default: 2 |
| Healthy Threshold | The number of consecutive successful health checks that must occur before declaring an EC2 instance healthy.  Valid values: 2 to 10  Default: 10 |

* An HTTP/HTTPS health check succeeds if the instance returns a 200 response code within the health check interval.
* A TCP health check succeeds if the TCP connection succeeds.
* An SSL health check succeeds if the SSL handshake succeeds.

# Configure Security Groups for Your Classic Load Balancer

In both EC2-Classic and in a VPC, 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. In a VPC, your security groups and network access control lists (ACL) must allow traffic in both directions on these ports.

# Subnets for Your Classic Load Balancer in a VPC

Select subnets from the same Availability Zones as your instances. If your load balancer is an Internet-facing load balancer, you must select public subnets in order for your back-end instances to receive traffic from the load balancer (even if the back-end instances are in private subnets). If your load balancer is an internal load balancer, we recommend that you select private subnets.

After you add a subnet, the load balancer starts routing requests to the registered instances in the corresponding Availability Zone. By default, the load balancer routes requests evenly across the Availability Zones for its subnets. To route requests evenly across the registered instances in the Availability Zones for its subnets, enable cross-zone load balancing.

# Listeners for Your Classic Load Balancer

The HTTP requests and HTTP responses use header fields to send information about HTTP messages. Elastic Load Balancing supports X-Forwarded-For headers. Because load balancers intercept traffic between clients and servers, your server access logs contain only the IP address of the load balancer. To see the IP address of the client, use the X-Forwarded-For request header. For more information, see [X-Forwarded-For](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/x-forwarded-headers.html#x-forwarded-for).

When you use HTTP/HTTPS, you can enable sticky sessions on your load balancer. A sticky session binds a user's session to a specific back-end instance. This ensures that all requests coming from the user during the session are sent to the same back-end instance. For more information, see [Configure Sticky Sessions for Your Classic Load Balancer](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/elb-sticky-sessions.html).

## X-Forwarded-For

The X-Forwarded-For request header helps you identify the IP address of a client when you use an HTTP or HTTPS load balancer. Because load balancers intercept traffic between clients and servers, your server access logs contain only the IP address of the load balancer. To see the IP address of the client, use the X-Forwarded-For request header. Elastic Load Balancing stores the IP address of the client in the X-Forwarded-For request header and passes the header to your server.

## X-Forwarded-Proto

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

## X-Forwarded-Port

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

# HTTPS Listeners for Your Classic Load Balancer

You can create a load balancer that uses the SSL/TLS protocol for encrypted connections (also known as SSL offload). This feature enables traffic encryption between your load balancer and the clients that initiate HTTPS sessions, and for connections between your load balancer and your EC2 instances.

Elastic Load Balancing uses Secure Sockets Layer (SSL) negotiation configurations, known assecurity policies, to negotiate connections between the clients and the load balancer. When you use HTTPS/SSL for your front-end connections, you can use either a predefined security policy or a custom security policy. You must deploy an SSL 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 instances. The load balancer uses a static cipher suite for back-end connections. You can optionally choose to enable authentication on your instances.

A security policy is a combination of SSL protocols, SSL ciphers, and the Server Order Preference option.

# Create a Classic Load Balancer with an HTTPS Listener

A load balancer takes requests from clients and distributes them across the EC2 instances that are registered with the load balancer.

You can create a load balancer that listens on both the HTTP (80) and HTTPS (443) ports. If you specify that the HTTPS listener sends requests to the instances on port 80, the load balancer terminates the requests and communication from the load balancer to the instances is not encrypted. If the HTTPS listener sends requests to the instances on port 443, communication from the load balancer to the instances is encrypted.

If your load balancer uses an encrypted connection to communicate with the instances, you can optionally enable authentication of the instances. This ensures that the load balancer communicates with an instance only if its public key matches the key that you specified to the load balancer for this purpose.

If you plan to enable the keep-alive option on your EC2 instances, we recommend that you set the keep-alive settings to at least the idle timeout settings of your load balancer. If you want to ensure that the load balancer is responsible for closing the connections to your instance, make sure that the value set on your instance for the keep-alive time is greater than the idle timeout setting on your load balancer.

First, provide some basic configuration information for your load balancer, such as a name, a network, and one or more listeners.

A listener is a process that checks for connection requests. It is configured with a protocol and a port for front-end (client to load balancer) connections and a protocol and a port for back-end (load balancer to instance) connections. For information about the ports, protocols, and listener configurations supported by Elastic Load Balancing, see [Listeners for Your Classic Load Balancer](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/elb-listener-config.html).

In this example, you configure two listeners for your load balancer. The first listener accepts HTTP requests on port 80 and sends them to the instances on port 80 using HTTP. The second listener accepts HTTPS requests on port 443 and sends them to the instances using HTTP on port 80 (or using HTTPS on port 443 if you want to configure back-end instance authentication).

# Configure the Idle Connection Timeout for Your Classic Load Balancer

For each request that a client makes through a Classic Load Balancer, the load balancer maintains two connections. One connection is with the client and the other connection is with a registered EC2 instance. For each 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 connection.

By default, Elastic Load Balancing sets the idle timeout to 60 seconds for both connections. Therefore, if the instance doesn't send some data at least every 60 seconds while the request is in flight, the load balancer can close the connection. To ensure that lengthy operations such as file uploads have time to complete, send at least 1 byte of data before each idle timeout period elapses, and increase the length of the idle timeout period as needed.

If you use HTTP and HTTPS listeners, we recommend that you enable the HTTP keep-alive option for your instances. You can enable keep-alive in the web server settings for your instances. Keep-alive, when enabled, enables the load balancer to reuse connections to your instance, which reduces the CPU utilization. To ensure that the load balancer is responsible for closing the connections to your instance, make sure that the value you set for the HTTP keep-alive time is greater than the idle timeout setting on your load balancer.

Note that TCP keep-alive probes do not prevent the load balancer from terminating the connection because they do not send data in the payload.

# Configure Cross-Zone Load Balancing for Your Classic Load Balancer

With cross-zone load balancing, each load balancer node for your Classic Load Balancer distributes requests evenly across the registered instances in all enabled Availability Zones. If cross-zone load balancing is disabled, each load balancer node distributes requests evenly across the registered instances in its Availability Zone only. For more information, see [Cross-Zone Load Balancing](https://docs.aws.amazon.com/elasticloadbalancing/latest/userguide/how-elastic-load-balancing-works.html#cross-zone-load-balancing) in the Elastic Load Balancing User Guide.

Cross-zone load balancing reduces the need to maintain equivalent numbers of instances in each enabled Availability Zone, and improves your application's ability to handle the loss of one or more instances. However, we still recommend that you maintain approximately equivalent numbers of instances in each enabled Availability Zone for higher fault tolerance.

For environments where clients cache DNS lookups, incoming requests might favor one of the Availability Zones. Using cross-zone load balancing, this imbalance in the request load is spread across all available instances in the region, reducing the impact of misbehaving clients.

When you create a Classic Load Balancer, the default for cross-zone load balancing depends on how you create the load balancer. With the API or CLI, cross-zone load balancing is disabled by default. With the AWS Management Console, the option to enable cross-zone load balancing is selected by default. After you create a Classic Load Balancer, you can enable or disable cross-zone load balancing at any time.

# Configure Connection Draining for Your Classic Load Balancer

To ensure that a Classic Load Balancer stops sending requests to instances that are de-registering or unhealthy, while keeping the existing connections open, use connection draining. This enables the load balancer to complete in-flight requests made to instances that are de-registering or unhealthy.

When you enable connection draining, you can specify a maximum time for the load balancer to keep connections alive before reporting the instance as de-registered. The maximum timeout value can be set between 1 and 3,600 seconds (the default is 300 seconds). When the maximum time limit is reached, the load balancer forcibly closes connections to the de-registering instance.

While in-flight requests are being served, the load balancer reports the state of a de-registering instance as InService: Instance deregistration currently in progress. When the de-registering instance is finished serving all in-flight requests, or when the maximum timeout limit is reached, the load balancer reports the instance state as OutOfService: Instance is not currently registered with the LoadBalancer.

If an instance becomes unhealthy, the load balancer reports the instance state as OutOfService. If there are in-flight requests made to the unhealthy instance, they are completed. The maximum timeout limit does not apply to connections to unhealthy instances.

If your instances are part of an Auto Scaling group and connection draining is enabled for your load balancer, Auto Scaling waits for the in-flight requests to complete, or for the maximum timeout to expire, before terminating instances due to a scaling event or health check replacement.

# Configure Proxy Protocol Support for Your Classic Load Balancer

Proxy Protocol is an Internet protocol used to carry connection information from the source requesting the connection to the destination for which the connection was requested. Elastic Load Balancing uses Proxy Protocol version 1, which uses a human-readable header format.

By default, when you use Transmission Control Protocol (TCP) for both front-end and back-end connections, your Classic Load Balancer forwards requests to the instances without modifying the request headers. If you enable Proxy Protocol, a human-readable header is added to the request header with connection information such as the source IP address, destination IP address, and port numbers. The header is then sent to the instance as part of the request.

**Note**

The AWS Management Console does not support enabling Proxy Protocol.

## Proxy Protocol Header

The Proxy Protocol header helps you identify the IP address of a client when you have a load balancer that uses TCP for back-end connections. Because load balancers intercept traffic between clients and your instances, the access logs from your instance contain the IP address of the load balancer instead of the originating client. You can parse the first line of the request to retrieve your client's IP address and the port number.

# Configure Sticky Sessions for Your Classic Load Balancer

By default, a Classic Load Balancer routes each request independently to the registered instance with the smallest load. However, you can use the sticky session feature (also known as session affinity), which enables the load balancer to bind a user's session to a specific instance. This ensures that all requests from the user during the session are sent to the same instance.

## Duration-Based Session Stickiness

The load balancer uses a special cookie to track the instance for each request to each listener. When the load balancer receives a request, it first checks to see if this cookie is present in the request. If so, the request is sent to the instance specified in the cookie. If there is no cookie, the load balancer chooses an instance based on the existing load balancing algorithm. A cookie is inserted into the response for binding subsequent requests from the same user to that instance. The stickiness policy configuration defines a cookie expiration, which establishes the duration of validity for each cookie. The load balancer does not refresh the expiry time of the cookie and does not check whether the cookie is expired before using it. After a cookie expires, the session is no longer sticky. The client should remove the cookie from its cookie store upon expiry.

# Configure a Custom Domain Name for Your Classic Load Balancer

Each Classic Load Balancer receives a default Domain Name System (DNS) name. This DNS name includes the name of the AWS region in which the load balancer is created. For example, if you create a load balancer named my-loadbalancer in the US West (Oregon) region, your load balancer receives a DNS name such as my-loadbalancer-1234567890.us-west-2.elb.amazonaws.com. To access the website on your instances, you paste this DNS name into the address field of a web browser. However, this DNS name is not easy for your customers to remember and use.

If you'd prefer to use a friendly DNS name for your load balancer, such as www.example.com, instead of the default DNS name, you can create a custom domain name and associate it with the DNS name for your load balancer. When a client makes a request using this custom domain name, the DNS server resolves it to the DNS name for your load balancer.

## Configure DNS Failover for Your Load Balancer

If you use Route 53 to route DNS queries to your load balancer, you can also configure DNS failover for your load balancer using Route 53. In a failover configuration, Route 53 checks the health of the registered EC2 instances for the load balancer to determine whether they are available. If there are no healthy EC2 instances registered with the load balancer, or if the load balancer itself is unhealthy, Route 53 routes traffic to another available resource, such as a healthy load balancer or a static website in Amazon S3.

# Monitor Your Classic Load Balancer

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

**CloudWatch metrics**

**Elastic Load Balancing access logs**

The access logs for Elastic Load Balancing capture detailed information for requests made to your load balancer and stores them as log files in the Amazon S3 bucket that you specify. Each log contains details such as the time a request was received, the client's IP address, latencies, request path, and server responses. You can use these access logs to analyze traffic patterns and to troubleshoot your back-end applications.

**CloudTrail logs**

AWS CloudTrail enables you to keep track of the calls made to the Elastic Load Balancing API by or on behalf of your AWS account. CloudTrail stores the information in log files in the Amazon S3 bucket that you specify. You can use these log files to monitor activity of your load balancers by determining which requests were made, the source IP addresses where the requests came from, who made the request, when the request was made, and so on.

# Troubleshoot Your Classic Load Balancer

The following tables list the troubleshooting resources that you'll find useful as you work with a Classic Load Balancer.

**API Errors**

|  |
| --- |
| **Error** |
| [CertificateNotFound: undefined](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/ts-elb-error-api-response.html#ts-elb-error-message-certificate) |
| [OutofService: A Transient Error Occurred](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/ts-elb-error-api-response.html#ts-elb-error-message-service) |

**HTTP Errors**

|  |
| --- |
| **Error** |
| [HTTP 400: BAD\_REQUEST](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/ts-elb-error-message.html#ts-elb-errorcodes-http400) |
| [HTTP 405: METHOD\_NOT\_ALLOWED](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/ts-elb-error-message.html#ts-elb-errorcodes-http405) |
| [HTTP 408: Request Timeout](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/ts-elb-error-message.html#ts-elb-errorcodes-http408) |
| [HTTP 502: Bad Gateway](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/ts-elb-error-message.html#ts-elb-errorcodes-http502) |
| [HTTP 503: Service Unavailable](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/ts-elb-error-message.html#ts-elb-errorcodes-http503) |
| [HTTP 504: Gateway Timeout](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/ts-elb-error-message.html#ts-elb-errorcodes-http504) |

**Response Code Metrics**

|  |
| --- |
| **Response Code Metric** |
| [HTTPCode\_ELB\_4XX](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/ts-elb-http-errors.html#ts-elb-error-metrics-ELB_4XX) |
| [HTTPCode\_ELB\_5XX](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/ts-elb-http-errors.html#ts-elb-error-metrics-ELB_5XX) |
| [HTTPCode\_Backend\_2XX](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/ts-elb-http-errors.html#ts-elb-error-metrics-Backend_2XX) |
| [HTTPCode\_Backend\_3XX](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/ts-elb-http-errors.html#ts-elb-error-metrics-Backend_3XX) |
| [HTTPCode\_Backend\_4XX](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/ts-elb-http-errors.html#ts-elb-error-metrics-Backend_4XX) |
| [HTTPCode\_Backend\_5XX](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/ts-elb-http-errors.html#ts-elb-error-metrics-Backend_5XX) |

**Health Check Issues**

|  |
| --- |
| **Issue** |
| [Health check target page error](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/ts-elb-healthcheck.html#ts-elb-healthcheck-targetpage) |
| [Connection to the instances has timed out](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/ts-elb-healthcheck.html#ts-elb-healthcheck-failed) |
| [Public key authentication is failing](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/ts-elb-healthcheck.html#ts-elb-healthcheck-publickey) |
| [Instance is not receiving traffic from the load balancer](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/ts-elb-healthcheck.html#ts-elb-healthcheck-securitygroup) |
| [Ports on instance are not open](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/ts-elb-healthcheck.html#ts-elb-healthcheck-ports) |
| [Instances in an Auto Scaling group are failing the ELB health check](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/ts-elb-healthcheck.html#ts-elb-healthcheck-autoscaling) |

**Connectivity Issues**

|  |
| --- |
| **Issue** |
| [Clients cannot connect to the load balancer](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/ts-elb-connection-failed.html) |

**Instance Registration Issues**

|  |
| --- |
| **Issue** |
| [Taking too long to register an EC2 instance](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/ts-elb-register-instance.html#ts-elb-register-too-long) |
| [Unable to register an instance launched from a paid AMI](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/ts-elb-register-instance.html#ts-elb-paid-ami-instance) |

# Troubleshoot a Classic Load Balancer: Client Connectivity

If your Internet-facing load balancer in a VPC 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 virtual private cloud (VPC).

**A security group or network ACL does not allow traffic**

The security group for the load balancer and any network ACLs for the load balancer subnets must allow inbound traffic from the clients and outbound traffic to the clients on the listener ports. For more information, see [Security Groups for Load Balancers in a VPC](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/elb-security-groups.html#elb-vpc-security-groups).

# **Application Load Balancer**

## Application Load Balancer Overview

An Application Load Balancer functions at the application layer, the seventh layer of the Open Systems Interconnection (OSI) model.

## Benefits of Migrating from a Classic Load Balancer

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

* Support for path-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 host-based routing. You can configure rules for your listener that forward requests based on the host field in the HTTP header. This enables you to route requests to multiple domains using a single load balancer.
* Support for routing based on fields in the request, such as standard and custom HTTP headers and methods, query parameters, and source IP addresses.
* Support for routing requests to multiple applications on a single EC2 instance. You can register each instance or IP address with the same target group using multiple ports.
* Support for redirecting requests from one URL to another.
* Support for returning a custom HTTP response.
* Support for registering targets by IP address, including targets outside the VPC for the load balancer.
* Support for registering Lambda functions as targets.
* Support for the load balancer to authenticate users of your applications through their corporate or social identities before routing requests.
* Support for containerized applications. Amazon Elastic Container Service (Amazon ECS) can select an unused port when scheduling a task and register the task with a target group using this port. This enables you to make efficient use of your clusters.
* Support for monitoring the health of each service independently, as health checks are defined at the target group level and many CloudWatch metrics are reported at the target group level. Attaching a target group to an Auto Scaling group enables you to scale each service dynamically based on demand.
* Access logs contain additional information and are stored in compressed format.
* Improved load balancer performance.

## Related Services

Elastic Load Balancing works with the following services to improve the availability and scalability of your applications.

* **Amazon EC2**
* **Amazon EC2 Auto Scaling**
* **AWS Certificate Manager**
* **Amazon CloudWatch**
* **Amazon ECS**
* **Route 53**
* **AWS WAF**

# 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.

# 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.

# Application Load Balancers

## Subnets for Your Load Balancer

When you create a load balancer, you must specify one public subnet from at least two Availability Zones. You can specify only one public subnet per Availability Zone.

To ensure that your load balancer can scale properly, verify that each subnet for your load balancer has a CIDR block with at least a /27 bitmask (for example, 10.0.0.0/27) and has at least 8 free IP addresses. Your load balancer uses these IP addresses to establish connections with the targets.

## 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.

## 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:

access\_logs.s3.enabled Indicates whether access logs stored in Amazon S3 are enabled. The default is false.

access\_logs.s3.bucket The name of the S3 bucket for the access logs. This attribute is required if access logs are enabled.

access\_logs.s3.prefix The prefix for the location in the S3 bucket.

deletion\_protection.enabled Indicates whether deletion protection is enabled. The default is false.

idle\_timeout.timeout\_seconds The idle timeout value, in seconds. The default is 60 seconds.

routing.http2.enabled Indicates whether HTTP/2 is enabled. The default is true.

## 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 communicate with the load balancer using IPv4 addresses resolve the A record and clients that communicate with the load balancer using IPv6 addresses resolve the AAAA record. However, the load balancer communicates with its targets using IPv4 addresses, regardless of how the client communicates with the load balancer.

## 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.

# 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.

## Listener Configuration

Listeners support the following protocols and ports:

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

## Listener Rules

Each listener has a default rule, and you can optionally define additional rules. Each rule consists of a priority, one or more actions, and one or more conditions. You can add or edit rules at any time. For more information

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

### Rule Priority Each rule has a priority. Rules are evaluated in priority order, from the lowest value to the highest value. The default rule is evaluated last. You can change the priority of a nondefault rule at any time. You cannot change the priority of the default rule. For more information

### Rule Actions Each rule action has a type, an order, and the information required to perform the action.

### Rule Conditions Each rule condition has a type and configuration information. When the conditions for a rule are met, then its actions are performed.

## Rule Action Types

The following are the supported action types for a rule:

authenticate-cognito [HTTPS listeners] Use Amazon Cognito to authenticate users

Authenticate users through well-known social IdPs, such as Amazon, Facebook, or Google, through the user pools supported by Amazon Cognito.

Authenticate users through corporate identities, using SAML, LDAP, or Microsoft AD, through the user pools supported by Amazon Cognito.

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

fixed-response Return a custom HTTP response. For more information, see [Fixed-Response Actions](https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-listeners.html#fixed-response-actions).

forwardForward requests to the specified target group.

redirectRedirect requests from one URL to another. For more information, You can use redirect actions to redirect client requests from one URL to another. You can configure redirects as either temporary (HTTP 302) or permanent (HTTP 301) based on your needs.

The action with the lowest order value is performed first. Each rule must include exactly one of the following actions: forward, redirect, or fixed-response, and it must be the last action to be performed.

## Rule Condition Types

The following are the supported condition types for a rule:

host-header Route based on the host name of each request..

http-header Route based on the HTTP headers for each request.

http-request-method Route based on the HTTP request method of each request.

path-pattern Route based on path patterns in the request URLs.

query-string Route based on key/value pairs or values in the query strings.

source-ip Route based on the source IP address of each request.

Each rule can include zero or one of the following conditions: host-header, http-request-method, path-pattern, and source-ip, and zero or more of the following conditions: http-header and query-string.

You can include wildcard characters in the match evaluations for the http-header, host-header, path-pattern, and query-string conditions. There is a limit of five wildcard characters per rule.

# Target Groups for Your Application Load Balancers

Each target group is used to route requests to one or more registered targets. When you create each listener rule, you specify a target group and conditions. When a rule condition is met, traffic is forwarded to the corresponding target group. You can create different target groups for different types of requests.

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.

## 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

## Target Type

When you create a target group, you specify its target type, which determines the type of target you specify when registering targets with this target group. After you create a target group, you cannot change its target type.

The following are the possible target types:

instance

The targets are specified by instance ID.

ip

The targets are IP addresses.

lambda

The target is a Lambda function.

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

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

These supported CIDR blocks enable you to register the following with a target group:

* ClassicLink instances,
* instances in a peered VPC,
* AWS resources that are addressable by IP address and port (for example, databases),
* and on-premises resources linked to AWS through AWS Direct Connect or a VPN connection.

**Important**

You can't specify publicly routable IP addresses.

If you specify targets using an instance ID, traffic is routed to instances using the primary private IP address specified in the primary network interface for the instance. If you specify targets using IP addresses, you can route traffic to an instance using any private IP address from one or more network interfaces. This enables multiple applications on an instance to use the same port. Each network interface can have its own security group.

If the target type of your target group is lambda, you can register a single Lambda function. When the load balancer receives a request for the Lambda function, it invokes the Lambda function.

## 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 each EC2 instance or IP address with the same target group multiple times using different ports, which enables the load balancer to route requests to microservices.

If demand on your application increases, you can register additional targets with one or more target groups in order to handle the demand.

If demand on your application decreases, or you need to service your targets, you can deregister targets from your target groups. Deregistering a target removes it from your target group, but does not affect the target otherwise. The load balancer stops routing requests to a target as soon as it is deregistered. The target enters the draining state until in-flight requests have completed. You can register the target with the target group again when you are ready for it to resume receiving requests.

If you are registering targets by instance ID, you can use your load balancer with an Auto Scaling group. After you attach a target group to an Auto Scaling group, Auto Scaling registers your targets with the target group for you when it launches them. For more information, see [Attaching a Load Balancer to Your Auto Scaling Group](https://docs.aws.amazon.com/autoscaling/ec2/userguide/attach-load-balancer-asg.html) in the Amazon EC2 Auto Scaling User Guide.

## Target Group Attributes

The following target group attributes are supported if the target group type is instance or ip:

deregistration\_delay.timeout\_seconds

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

slow\_start.duration\_seconds

The time period, in seconds, during which the load balancer sends a newly registered target a linearly increasing share of the traffic to the target group. The range is 30–900 seconds (15 minutes). The default is 0 seconds (disabled).

stickiness.enabled

Indicates whether sticky sessions are enabled.

stickiness.lb\_cookie.duration\_seconds

The cookie expiration period, in seconds. After this period, the cookie is considered stale. The minimum value is 1 second and the maximum value is 7 days (604800 seconds). The default value is 1 day (86400 seconds).

stickiness.type

The type of stickiness. The possible value is lb\_cookie.

The following target group attribute is supported if the target group type is lambda:

lambda.multi\_value\_headers.enabled

Indicates whether the request and response headers exchanged between the load balancer and the Lambda function include arrays of values or strings. The possible values are true or false. The default value is false. For more information, see [Multi-Value Headers](https://docs.aws.amazon.com/elasticloadbalancing/latest/application/lambda-functions.html#multi-value-headers).

## 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 |

# Lambda Functions as Targets

## Enable Health Checks

By default, health checks are disabled for target groups of type lambda. You can enable health checks in order to implement DNS failover with Amazon Route 53. The Lambda function can check the health of a downstream service before responding to the health check request. If the response from the Lambda function indicates a health check failure, the health check failure is passed to Route 53. You can configure Route 53 to fail over to a backup application stack.

You are charged for health checks as you are for any Lambda function invocation.

## Deregister the Lambda Function

If you no longer need to send traffic to your Lambda function, you can deregister it. After you deregister a Lambda function, in-flight requests fail with HTTP 5XX errors.

To replace a Lambda function, we recommend that you create a new target group, register the new function with the new target group, and update the listener rules to use the new target group instead of the existing one.

# 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**

**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. Access logging is an optional feature of Elastic Load Balancing that is disabled by default. Each access log file is automatically encrypted before it is stored in your S3 bucket and decrypted when you access it. You do not need to take any action; the encryption and decryption is performed transparently.

**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.

**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..

# Troubleshoot Your Application Load Balancers

## 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**

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

**The ping path does not exist**

**The connection times out**

**The target did not return a successful response code**

## 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**

**A security group or network ACL does not allow traffic**

## The load balancer sends requests to unhealthy targets

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

## The load balancer generates an HTTP error

The following HTTP errors are generated by the load balancer. The load balancer sends the HTTP code to the client, saves the request to the access log, and increments theHTTPCode\_ELB\_4XX\_Count or HTTPCode\_ELB\_5XX\_Count metric.

**Errors**

* [HTTP 400: Bad Request](https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-troubleshooting.html#http-400-issues)
* [HTTP 401: Unauthorized](https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-troubleshooting.html#http-401-issues)
* [HTTP 403: Forbidden](https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-troubleshooting.html#http-403-issues)
* [HTTP 408: Request Timeout](https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-troubleshooting.html#http-408-issues)
* [HTTP 413: Payload Too Large](https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-troubleshooting.html#http-413-issues)
* [HTTP 414: URI Too Long](https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-troubleshooting.html#http-414-issues)
* [HTTP 460](https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-troubleshooting.html#http-460-issues)
* [HTTP 463](https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-troubleshooting.html#http-463-issues)
* [HTTP 500: Internal Server Error](https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-troubleshooting.html#http-500-issues)
* [HTTP 501: Not Implemented](https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-troubleshooting.html#http-501-issues)
* [HTTP 502: Bad Gateway](https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-troubleshooting.html#http-502-issues)
* [HTTP 503: Service Unavailable](https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-troubleshooting.html#http-503-issues)
* [HTTP 504: Gateway Timeout](https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-troubleshooting.html#http-504-issues)
* [HTTP 561: Unauthorized](https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-troubleshooting.html#http-561-issues)

### HTTP 400: Bad Request

Possible causes:

* The client sent a malformed request that does not meet the HTTP specification.
* The client used the HTTP CONNECT method, which is not supported by Application Load Balancers.
* The request header exceeded 16K per request line, 16K per single header, or 64K for the entire header.

### HTTP 401: Unauthorized

You configured a listener rule to authenticate users. Either you configured OnUnauthenticatedRequest to deny unauthenticated users or the IdP denied access.

### HTTP 403: Forbidden

You configured an AWS WAF web access control list (web ACL) to monitor requests to your Application Load Balancer and it blocked a request.

### HTTP 408: Request Timeout

The client did not send data before the idle timeout period expired. Sending a TCP keep-alive does not prevent this timeout. Send at least 1 byte of data before each idle timeout period elapses. Increase the length of the idle timeout period as needed.

### HTTP 413: Payload Too Large

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

### HTTP 414: URI Too Long

The request URL or query string parameters are too large.

### HTTP 460

The load balancer received a request from a client, but the client closed the connection with the load balancer before the idle timeout period elapsed.

Check whether the client timeout period is greater than the idle timeout period for the load balancer. Ensure that your target provides a response to the client before the client timeout period elapses, or increase the client timeout period to match the load balancer idle timeout, if the client supports this.

### HTTP 463

The load balancer received an **X-Forwarded-For** request header with more than 30 IP addresses.

### HTTP 500: Internal Server Error

Possible causes:

* You configured an AWS WAF web access control list (web ACL) and there was an error executing the web ACL rules.
* You configured a listener rule to authenticate users, but one of the following is true:
  + The load balancer is unable to communicate with the IdP token endpoint or the IdP user info endpoint. Verify that the security groups for your load balancer and the network ACLs for your VPC allow outbound access to these endpoints. Verify that your VPC has internet access. If you have an internal-facing load balancer, use a NAT gateway to enable internet access.
  + The size of the claims returned by the IdP exceeded the maximum size supported by the load balancer.
  + A client submitted an HTTP/1.0 request without a host header, and the load balancer was unable to generate a redirect URL.
  + A client submitted a request without an HTTP protocol, and the load balancer was unable to generate a redirect URL.
  + The requested scope doesn't return an ID token.

### HTTP 501: Not Implemented

The load balancer received a **Transfer-Encoding** header with an unsupported value. The supported values for **Transfer-Encoding** are chunked and identity. As an alternative, you can use the **Content-Encoding** header.

### HTTP 502: Bad Gateway

Possible causes:

* The load balancer received a TCP RST from the target when attempting to establish a connection.
* The load balancer received an unexpected response from the target, such as "ICMP Destination unreachable (Host unreachable)", when attempting to establish a connection. Check whether traffic is allowed from the load balancer subnets to the targets on the target port.
* The target closed the connection with a TCP RST or a TCP FIN while the load balancer had an outstanding request to the target. Check whether the keep-alive duration of the target is shorter than the idle timeout value of the load balancer.
* The target response is malformed or contains HTTP headers that are not valid.
* The load balancer encountered an SSL handshake error or SSL handshake timeout (10 seconds) when connecting to a target.
* The deregistration delay period elapsed for a request being handled by a target that was deregistered. Increase the delay period so that lengthy operations can complete.
* The target is a Lambda function and the response body exceeds 1 MB.
* The target is a Lambda function that did not respond before its configured timeout was reached.

### HTTP 503: Service Unavailable

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

### HTTP 504: Gateway Timeout

Possible causes:

* The load balancer failed to establish a connection to the target before the connection timeout expired (10 seconds).
* The load balancer established a connection to the target but the target did not respond before the idle timeout period elapsed.
* The network ACL for the subnet did not allow traffic from the targets to the load balancer nodes on the ephemeral ports (1024-65535).
* The target returns a content-length header that is larger than the entity body. The load balancer timed out waiting for the missing bytes.
* The target is a Lambda function that did not respond before its possible maximum configured timeout was reached.

### HTTP 561: Unauthorized

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

## A target generates an HTTP error

The load balancer forwards valid HTTP responses from targets to the client, including HTTP errors. The HTTP errors generated by a target are recorded in the HTTPCode\_Target\_4XX\_Countand HTTPCode\_Target\_5XX\_Count metrics.

# **Network Load Balancer**

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

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

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

## Benefits of Migrating from a Classic Load Balancer

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

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

# Network Load Balancers

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

## 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.

When you create an Internet-facing load balancer, you can optionally specify one Elastic IP address per subnet. This provides your load balancer with static IP addresses. You cannot add or change the Elastic IP addresses for your subnets after you create the load balancer.

### Cross-Zone Load Balancing

By default, each load balancer node distributes traffic across the registered targets in its Availability Zone only. If you enable cross-zone load balancing, each load balancer node distributes traffic across the registered targets in all enabled Availability Zones. For more information, see [Cross-Zone Load Balancing](https://docs.aws.amazon.com/elasticloadbalancing/latest/userguide/how-elastic-load-balancing-works.html#cross-zone-load-balancing) in the Elastic Load Balancing User Guide.

## 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.

## Connection Idle Timeout

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

Elastic Load Balancing sets the idle timeout value for TCP flows to 350 seconds. You cannot modify this value. For TCP listeners, clients and targets can use TCP keepalive packets to reset the idle timeout. TCP keepalive packets are not supported for TLS listeners.

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

# Listeners for Your Network Load Balancers

## Listener Configuration

Listeners support the following protocols and ports:

* **Protocols**: TCP, TLS, UDP, TCP\_UDP
* **Ports**: 1-65535

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

# Update a Listener for Your Network Load Balancer

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

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

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

# Target Groups for Your Network Load Balancers

**Routing Configuration**

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

* **Protocols**: TCP, TLS, UDP, TCP\_UDP
* **Ports**: 1-65535

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Listener Protocol** | **Target Group Protocol** | **Target Group Type** | **Health Check Protocol** |
| TCP | TCP | TCP\_UDP | instance | ip | HTTP | HTTPS | TCP |
| TLS | TCP | TLS | instance | ip | HTTP | HTTPS | TCP |
| UDP | UDP | TCP\_UDPs | instance | HTTP | HTTPS | TCP |
| TCP\_UDP | TCP\_UDP | instance | HTTP | HTTPS | TCP |

## Target Type

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

The following are the possible target types:

instance

The targets are specified by instance ID.

ip

The targets are specified by IP address.

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

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

**Important**

You can't specify publicly routable IP addresses.

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

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

If the target group protocol is UDP or TCP\_UDP, the target type must be instance.

Network Load Balancers do not support the lambda target type, only Application Load Balancers support the lambda target type. For more information, see [Lambda Functions as Targets](https://docs.aws.amazon.com/elasticloadbalancing/latest/application/lambda-functions.html) in the User Guide for Application Load Balancers.

### Request Routing and IP Addresses

If you specify targets using an instance ID, traffic is routed to instances using the primary private IP address specified in the primary network interface for the instance. If you specify targets using IP addresses, you can route traffic to an instance using any private IP address from one or more network interfaces. This enables multiple applications on an instance to use the same port. Note that each network interface can have its own security group.

### Source IP Preservation

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

If you specify targets by IP address, the source IP addresses are the private IP addresses of the load balancer nodes. If you need the IP addresses of the clients, enable Proxy Protocol and get the client IP addresses from the Proxy Protocol header.

If you have micro services on instances registered with a Network Load Balancer, you cannot use the load balancer to provide communication between them unless the load balancer is internet-facing or the instances are registered by IP address.

## Registered Targets

If you are registering targets by instance ID, you can use your load balancer with an Auto Scaling group. After you attach a target group to an Auto Scaling group, Auto Scaling registers your targets with the target group for you when it launches them.

## Deregistration Delay

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

The initial state of a deregistering target is draining. By default, the load balancer changes the state of a deregistering target to unused after 300 seconds. To change the amount of time that the load balancer waits before changing the state of a deregistering target to unused, update the deregistration delay value. We recommend that you specify a value of at least 120 seconds to ensure that requests are completed.

# Health Checks for Your Target Groups

Network Load Balancers use active and passive health checks to determine whether a target is available to handle requests.

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

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

# Monitor Your Network Load Balancers

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

**CloudWatch metrics**

**VPC Flow Logs**

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

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

**CloudTrail logs**

# Troubleshoot Your Network Load Balancer

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

## A registered target is not in service

If a target is taking longer than expected to enter the InService state, it might be failing health checks. Your target is not in service until it passes one health check.

**A security group does not allow traffic**

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

## Requests are not routed to targets

**A security group does not allow traffic**

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

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

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

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

**The instance is in a peered VPC**

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

## Targets receive more health check requests than expected

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

## Targets receive fewer health check requests than expected

Check whether net.ipv4.tcp\_tw\_recycle is enabled. This setting is known to cause issues with load balancers. The net.ipv4.tcp\_tw\_reuse setting is considered a safer alternative.

## Unhealthy targets receive requests from the load balancer

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

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

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

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

* Register instances by IP address instead of instance ID. When using Amazon Elastic Container Service, use the awsvpc network mode with your tasks to ensure that target groups require registration by IP address.
* Ensure that containers that must communicate are on different container instances.
* Use an Internet-facing load balancer.

## Performance decreases when moving targets to a Network Load Balancer

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

## Port allocation errors connecting through AWS PrivateLink

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