Elastic Load Balancing

User Guide

Elastic Load Balancing: User Guide

Table of Contents

What Is Elastic Load Balancing?

Elastic Load Balancing distributes incoming application or network traffic across multiple targets, such as Amazon EC2 instances, containers, and IP addresses, in multiple Availability Zones. Elastic Load Balancing scales your load balancer as traffic to your application changes over time. It can automatically scale to the vast majority of workloads.

Load Balancer Benefits

A load balancer distributes workloads across multiple compute resources, such as virtual servers. Using a load balancer increases the availability and fault tolerance of your applications.

You can add and remove compute resources from your load balancer as your needs change, without disrupting the overall flow of requests to your applications.

You can configure health checks, which monitor the health of the compute resources, so that the load balancer sends requests only to the healthy ones. You can also offload the work of encryption and decryption to your load balancer so that your compute resources can focus on their main work.

Features of Elastic Load Balancing

Elastic Load Balancing supports three types of load balancers: Application Load Balancers, Network Load Balancers, and Classic Load Balancers. You can select a load balancer based on your application needs. For more information, see Comparison of Elastic Load Balancing Products.

For more information about using each load balancer, see the User Guide for Application Load Balancers, the User Guide for Network Load Balancers, and the User Guide for Classic Load Balancers.

Accessing Elastic Load Balancing

You can create, access, and manage your load balancers using any of the following interfaces:

- **AWS Management Console**— Provides a web interface that you can use to access Elastic Load Balancing.
- **AWS Command Line Interface (AWS CLI)** — Provides commands for a broad set of AWS services, including Elastic Load Balancing. The AWS CLI is supported on Windows, macOS, and Linux. For more information, see AWS Command Line Interface.
- **AWS SDKs** — Provide language-specific APIs and take care of many of the connection details, such as calculating signatures, handling request retries, and error handling. For more information, see AWS SDKs.
- **Query API**— Provides low-level API actions that you call using HTTPS requests. Using the Query API is the most direct way to access Elastic Load Balancing. However, the Query API requires that your application handle low-level details such as generating the hash to sign the request, and error handling. For more information, see the following:
  - Application Load Balancers and Network Load Balancers — API version 2015-12-01
  - Classic Load Balancers — API version 2012-06-01

Related Services

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

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

Pricing

With your load balancer, you pay only for what you use. For more information, see Elastic Load Balancing Pricing.

How Elastic Load Balancing Works

A load balancer accepts incoming traffic from clients and routes requests to its registered targets (such as EC2 instances) in one or more Availability Zones. The load balancer also monitors the health of its registered targets and ensures that it routes traffic only to healthy targets. When the load balancer detects an unhealthy target, it stops routing traffic to that target. It then resumes routing traffic to that target when it detects that the target is healthy again.

You configure your load balancer to accept incoming traffic by specifying one or more *listeners*. A listener is a process that checks for connection requests. It is configured with a protocol and port number for connections from clients to the load balancer. Likewise, it is configured with a protocol and port number for connections from the load balancer to the targets.

Elastic Load Balancing supports three types of load balancers:

- Application Load Balancers
- Network Load Balancers
- Classic Load Balancers

There is a key difference in how the load balancer types are configured. 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. Your load balancer is most effective when you ensure that each enabled Availability Zone has at least one registered target.

We recommend that you enable multiple Availability Zones. (With an Application Load Balancer, we require you to enable multiple Availability Zones.) This configuration helps ensure that the load balancer can continue to route traffic. If one Availability Zone becomes unavailable or has no healthy targets, the load balancer can 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. However, even though they remain registered, the load balancer does not route traffic to them.

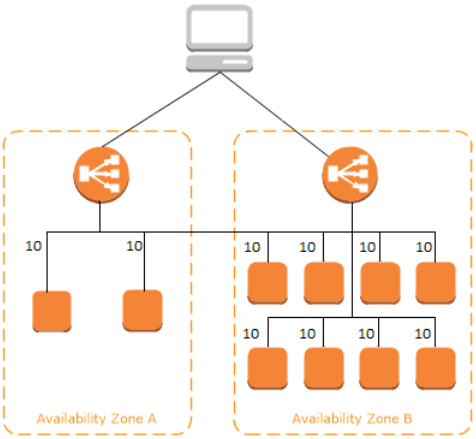
Cross-Zone Load Balancing

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 only across the registered targets in its Availability Zone.

The following diagrams demonstrate the effect of cross-zone load balancing. There are two enabled Availability Zones, with two targets in Availability Zone A and eight 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.

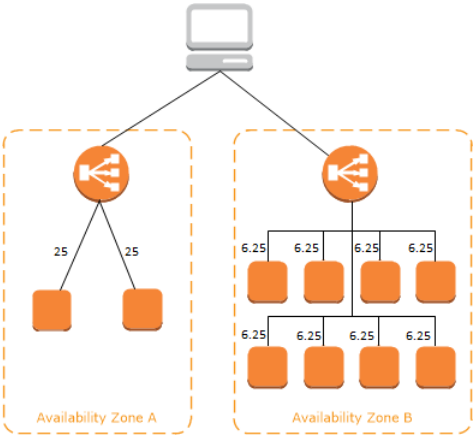
If cross-zone load balancing is enabled, each of the 10 targets receives 10% of the traffic. This is because each load balancer node can route its 50% of the client traffic to all 10 targets.



If cross-zone load balancing is disabled:

- Each of the two targets in Availability Zone A receives 25% of the traffic.
- Each of the eight 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.



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 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 in the *User Guide for Classic Load 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. These 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 that 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. The DNS entry also specifies the time-to-live (TTL) of 60 seconds. This helps ensure 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**, the load balancer node that receives the request uses the following process:

1. Evaluates the listener rules in priority order to determine which rule to apply.
2. Selects a target from the target group for the rule action, using the routing algorithm configured for the target group. The default routing algorithm is round robin. Routing is performed independently for each target group, even when a target is registered with multiple target groups.

With **Network Load Balancers**, the load balancer node that receives the connection uses the following process:

1. Selects a target from the target group for the default rule using a flow hash algorithm. It bases the algorithm on:
   - The protocol
   - The source IP address and source port
   - The destination IP address and destination port
   - The TCP sequence number
2. Routes each individual TCP connection to a single target for the life of the connection. The TCP connections from a client have different source ports and sequence numbers, and can be routed to different targets.

With **Classic Load Balancers**, the load balancer node that receives the request selects a registered instance as follows:

- Uses the round robin routing algorithm for TCP listeners
- Uses the least outstanding requests routing algorithm for HTTP and HTTPS listeners

HTTP Connections

Classic Load Balancers use pre-open connections, but Application Load Balancers do not. Both Classic Load Balancers and Application Load Balancers use connection multiplexing. This means that requests

from multiple clients on multiple front-end connections can be routed to a given target through a single backend connection. Connection multiplexing improves latency and reduces the load on your applications. To prevent connection multiplexing, disable HTTP `keep-alives` by setting the `Connection: close` header in your HTTP responses.

Classic Load Balancers support the following protocols on front-end connections (client to load balancer): HTTP/0.9, HTTP/1.0, and HTTP/1.1.

Application Load Balancers support the following protocols on front-end connections: HTTP/0.9, HTTP/1.0, HTTP/1.1, and HTTP/2. You can use HTTP/2 only with HTTPS listeners, and can send up to 128 requests in parallel using one HTTP/2 connection. Application Load Balancers also support connection upgrades from HTTP to WebSockets.

Both Application Load Balancers and Classic Load Balancers use HTTP/1.1 on backend connections (load balancer to registered target). `Keep-alive` is supported on backend connections by default. For HTTP/1.0 requests from clients that do not have a host header, the load balancer generates a host header for the HTTP/1.1 requests sent on the backend connections. For Application Load Balancer, the host header contains the DNS name of the load balancer. For Classic Load Balancer, the host header contains the IP address of the load balancer node.

You can set an idle timeout value for both Application Load Balancers and Classic Load Balancers. The default value is 60 seconds. With an Application Load Balancer, the idle timeout value applies only to front-end connections. With a Classic Load Balancer, if a connection is idle for longer than the idle timeout value, the connection is torn down and the client receives an error response. This is true for both front-end and backend connections. A registered target can use a `keep-alive` timeout to keep a backend connection open until it is ready to tear it down.

Application Load Balancers and Classic Load Balancers support pipelined HTTP on front-end connections. They do not support pipelined HTTP on backend connections.

HTTP Headers

Application Load Balancers and Classic Load Balancers add **X-Forwarded-For**, **X-Forwarded-Proto**, and **X-Forwarded-Port** headers to the request.

For front-end connections that use HTTP/2, the header names are in lowercase. Before the request is sent to the target using HTTP/1.1, the following header names are converted to mixed case: **X-Forwarded-For**, **X-Forwarded-Proto**, **X-Forwarded-Port**, **Host**, **X-Amzn-Trace-Id**, **Upgrade**, and **Connection**. All other header names are in lowercase.

Application Load Balancers and Classic Load Balancers honor the connection header from the incoming client request after proxying the response back to the client.

HTTP Header Limits

The following size limits for Application Load Balancers are hard limits that cannot be changed.

HTTP/1.x Headers

- Request line: 16 K
- Single header: 16 K
- Whole header: 64 K

HTTP/2 Headers

- Request line: 8 K

- Single header: 8 K
- Whole header: 64 K

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.

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.

If your application has multiple tiers, you can design an architecture that uses both internal and internet-facing load balancers. For example, this is true if your application uses web servers that must be connected to the internet, and database servers that are only connected to the web servers. Create an internet-facing load balancer and register the web servers with it. Create an internal load balancer and register the database servers with it. The web servers receive requests from the internet-facing load balancer and send requests for the database servers to the internal load balancer. The database servers receive requests from the internal load balancer.

Getting Started with Elastic Load Balancing

There are three types of load balancers: Application Load Balancers, Network Load Balancers, and Classic Load Balancers. You can select a load balancer based on your application needs. For more information, see Comparison of Elastic Load Balancing Products.

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

If you have an existing Classic Load Balancer, you can migrate to an Application Load Balancer or a Network Load Balancer. For more information, see Migrate Your Classic Load Balancer (p. 24).

Create an Application Load Balancer

To create an Application Load Balancer using the AWS Management Console, see Getting Started with Application Load Balancers in the *User Guide for Application Load Balancers*.

To create an Application Load Balancer using the AWS CLI, see Create an Application Load Balancer Using the AWS CLI in the *User Guide for Application Load Balancers*.

Create a Network Load Balancer

To create a Network Load Balancer using the AWS Management Console, see Getting Started with Network Load Balancers in the *User Guide for Network Load Balancers*.

To create a Network Load Balancer using the AWS CLI, see Create a Network Load Balancer Using the AWS CLI in the *User Guide for Network Load Balancers*.

Create a Classic Load Balancer

To create a Classic Load Balancer using the AWS Management Console, see Create a Classic Load Balancer in the *User Guide for Classic Load Balancers*.

Security in Elastic Load Balancing

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that are built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security of the cloud and security in the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to Elastic Load Balancing, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Elastic Load Balancing. It shows you how to configure Elastic Load Balancing to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Elastic Load Balancing resources.

Contents

Data Protection in Elastic Load Balancing

Elastic Load Balancing conforms to the AWS shared responsibility model, which includes regulations and guidelines for data protection. AWS is responsible for protecting the global infrastructure that runs all AWS services. AWS maintains control over data hosted on this infrastructure, including the security configuration controls for handling customer content and personal data. AWS customers and APN Partners, acting either as data controllers or data processors, are responsible for any personal data that they put in the AWS Cloud.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM), so that each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use TLS to communicate with AWS resources.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.

We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into free-form fields or metadata, such as function names and tags. Any data that you enter into metadata might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don't include credential information in the URL to validate your request to that server.

For more information about data protection, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

Encryption at Rest

If you enable server-side encryption with Amazon S3-managed encryption keys (SSE-S3) for your S3 bucket for Elastic Load Balancing access logs, Elastic Load Balancing automatically encrypts each access log file before it is stored in your S3 bucket. Elastic Load Balancing also decrypts the access log files when you access them. Each log file is encrypted with a unique key, which is itself encrypted with a master key that is regularly rotated.

Encryption in Transit

Elastic Load Balancing simplifies the process of building secure web applications by terminating HTTPS and TLS traffic from clients at the load balancer. The load balancer performs the work of encrypting and decrypting the traffic, instead of requiring each EC2 instance to handle the work for TLS termination. When you configure a secure listener, you specify the cipher suites and protocol versions that are supported by your application, and a server certificate to install on your load balancer. You can use AWS Certificate Manager (ACM) or AWS Identity and Access Management (IAM) to manage your server certificates. Application Load Balancers support HTTPS listeners. Network Load Balancers support TLS listeners. Classic Load Balancers support both HTTPS and TLS listeners.

Identity and Access Management for Elastic Load Balancing

AWS uses security credentials to identify you and to grant you access to your AWS resources. You can use features of AWS Identity and Access Management (IAM) to allow other users, services, and applications to use your AWS resources fully or in a limited way. You can do this without sharing your security credentials.

By default, IAM users don't have permission to create, view, or modify AWS resources. To allow an IAM user to access resources such as a load balancer, and to perform tasks, you:

1. Create an IAM policy that grants the IAM user permission to use the specific resources and API actions they need.
2. Attach the policy to the IAM user or the group that the IAM user belongs to.

When you attach a policy to a user or group of users, it allows or denies the users permission to perform the specified tasks on the specified resources.

For example, you can use IAM to create users and groups under your AWS account. An IAM user can be a person, a system, or an application. Then you grant permissions to the users and groups to perform specific actions on the specified resources using an IAM policy.

Grant Permissions Using IAM Policies

When you attach a policy to a user or group of users, it allows or denies the users permission to perform the specified tasks on the specified resources.

An IAM policy is a JSON document that consists of one or more statements. Each statement is structured as shown in the following example.

```
{
  "Version": "2012-10-17",
  "Statement":[{
    "Effect": "effect",
    "Action": "action",
    "Resource": "resource-arn",
    "Condition": {
      "condition": {
        "key":"value"
      }
    }
  }]
}
```

- **Effect**— The *effect* can be `Allow` or `Deny`. By default, IAM users don't have permission to use resources and API actions, so all requests are denied. An explicit allow overrides the default. An explicit deny overrides any allows.
- **Action**— The *action* is the specific API action for which you are granting or denying permission. For more information about specifying *action*, see API Actions for Elastic Load Balancing (p. 11).
- **Resource**— The resource that's affected by the action. With many Elastic Load Balancing API actions, you can restrict the permissions granted or denied to a specific load balancer. To do so, specify its Amazon Resource Name (ARN) in this statement. Otherwise, you can use the * wildcard to specify all of your load balancers. For more information, see Elastic Load Balancing Resources (p. 12).
- **Condition**— You can optionally use conditions to control when your policy is in effect. For more information, see Condition Keys for Elastic Load Balancing (p. 15).

For more information, see the IAM User Guide.

API Actions for Elastic Load Balancing

In the **Action** element of your IAM policy statement, you can specify any API action that Elastic Load Balancing offers. You must prefix the action name with the lowercase string `elasticloadbalancing:`, as shown in the following example.

```
"Action": "elasticloadbalancing:DescribeLoadBalancers"
```

To specify multiple actions in a single statement, enclose them in square brackets and separate them with a comma, as shown in the following example.

```
"Action": [
    "elasticloadbalancing:DescribeLoadBalancers",
    "elasticloadbalancing:DeleteLoadBalancer"
]
```

You can also specify multiple actions using the * wildcard. The following example specifies all API action names for Elastic Load Balancing that start with `Describe`.

```
"Action": "elasticloadbalancing:Describe*"
```

To specify all API actions for Elastic Load Balancing, use the * wildcard, as shown in the following example.

```
"Action": "elasticloadbalancing:*"
```

For the complete list of the API actions for Elastic Load Balancing, see the following documentation:

- Application Load Balancers and Network Load Balancers — API Reference version 2015-12-01
- Classic Load Balancers — API Reference version 2012-06-01

Elastic Load Balancing Resources

Resource-level permissions refers to the ability to specify which resources users are allowed to perform actions on. Elastic Load Balancing has partial support for resource-level permissions. For API actions that support resource-level permissions, you can control the resources that users are allowed to use with the action. To specify a resource in a policy statement, you must use its Amazon Resource Name (ARN). When specifying an ARN, you can use the * wildcard in your paths. For example, you can use the * wildcard when you do not want to specify the exact load balancer name.

The ARN for an Application Load Balancer has the format shown in the following example.

```
arn:aws:elasticloadbalancing:region-code:account-id:loadbalancer/app/load-balancer-name/load-balancer-id
```

The ARN for a Network Load Balancer has the format shown in the following example.

```
arn:aws:elasticloadbalancing:region-code:account-id:loadbalancer/net/load-balancer-name/load-balancer-id
```

The ARN for a Classic Load Balancer has the format shown in the following example.

```
arn:aws:elasticloadbalancing:region-code:account-id:loadbalancer/load-balancer-name
```

The ARNs for a listener and a listener rule for an Application Load Balancer have the format shown in the following example.

```
arn:aws:elasticloadbalancing:region-code:account-id:listener/app/load-balancer-name/load-balancer-id/listener-id
arn:aws:elasticloadbalancing:region-code:account-id:listener-rule/app/load-balancer-name/load-balancer-id/listener-id/rule-id
```

The ARN for a listener for a Network Load Balancer has the format shown in the following example.

```
arn:aws:elasticloadbalancing:region-code:account-id:listener/net/load-balancer-name/load-balancer-id/listener-id
```

The ARN for a target group has the format shown in the following example.

```
arn:aws:elasticloadbalancing:region-code:account-id:targetgroup/target-group-name/target-group-id
```

API Actions with No Support for Resource-Level Permissions

The following Elastic Load Balancing actions do not support resource-level permissions:

- API version 2015-12-01:
  - `DescribeAccountLimits`

  - `DescribeListenerCertificates`
  - `DescribeListeners`
  - `DescribeLoadBalancerAttributes`
  - `DescribeLoadBalancers`
  - `DescribeRules`
  - `DescribeSSLPolicies`
  - `DescribeTags`
  - `DescribeTargetGroupAttributes`
  - `DescribeTargetGroups`
  - `DescribeTargetHealth`
- API version 2012-06-01:
  - `DescribeInstanceHealth`
  - `DescribeLoadBalancerAttributes`
  - `DescribeLoadBalancerPolicyTypes`
  - `DescribeLoadBalancers`
  - `DescribeLoadBalancerPolicies`
  - `DescribeTags`

For API actions that don't support resource-level permissions, you must specify the resource statement shown in the following example.

```
"Resource": "*"
```

Resource-Level Permissions for Elastic Load Balancing

The following tables describe the Elastic Load Balancing actions that support resource-level permissions, and the supported resources for each action.

API version 2015-12-01

API Action	Resource ARNs
`AddListenerCertificates`	listener
`AddTags`	load balancer, target group
`CreateListener`	load balancer
`CreateLoadBalancer`	load balancer
`CreateRule`	listener
`CreateTargetGroup`	target group
`DeleteListener`	listener
`DeleteLoadBalancer`	load balancer
`DeleteRule`	listener rule
`DeleteTargetGroup`	target group

API Action	Resource ARNs
DeregisterTargets	target group
ModifyListener	listener
ModifyLoadBalancerAttributes	load balancer
ModifyRule	listener rule
ModifyTargetGroup	target group
ModifyTargetGroupAttributes	target group
RegisterTargets	target group
RemoveListenerCertificates	listener
RemoveTags	load balancer, target group
SetIpAddressType	load balancer
SetRulePriorities	listener rule
SetSecurityGroups	load balancer
SetSubnets	load balancer

API version 2012-06-01

API Action	Resource ARNs
AddTags	load balancer
ApplySecurityGroupsToLoadBalancer	load balancer
AttachLoadBalancerToSubnets	load balancer
ConfigureHealthCheck	load balancer
CreateAppCookieStickinessPolicy	load balancer
CreateLBCookieStickinessPolicy	load balancer
CreateLoadBalancer	load balancer
CreateLoadBalancerListeners	load balancer
CreateLoadBalancerPolicy	load balancer
DeleteLoadBalancer	load balancer
DeleteLoadBalancerListeners	load balancer
DeleteLoadBalancerPolicy	load balancer
DeregisterInstancesFromLoadBalancer	load balancer
DetachLoadBalancerFromSubnets	load balancer
DisableAvailabilityZonesForLoadBalancer	load balancer

API Action	Resource ARNs
EnableAvailabilityZonesForLoadBalancer	load balancer
ModifyLoadBalancerAttributes	load balancer
RegisterInstancesWithLoadBalancer	load balancer
RemoveTags	load balancer
SetLoadBalancerListenerSSLCertificate	load balancer
SetLoadBalancerPoliciesForBackendServer	load balancer
SetLoadBalancerPoliciesOfListener	load balancer

Condition Keys for Elastic Load Balancing

When you create a policy, you can specify the conditions that control when the policy is in effect. Each condition contains one or more key-value pairs. There are global condition keys and service-specific condition keys.

You cannot use the `aws:SourceIp` condition key with Elastic Load Balancing.

The `elasticloadbalancing:ResourceTag/`*`key`* condition key is specific to Elastic Load Balancing. The following actions support this condition key:

API version 2015-12-01

- AddTags
- CreateListener
- CreateLoadBalancer
- DeleteLoadBalancer
- DeleteTargetGroup
- DeregisterTargets
- ModifyLoadBalancerAttributes
- ModifyTargetGroup
- ModifyTargetGroupAttributes
- RegisterTargets
- RemoveTags
- SetIpAddressType
- SetSecurityGroups
- SetSubnets

API version 2012-06-01

- AddTags
- ApplySecurityGroupsToLoadBalancer
- AttachLoadBalancersToSubnets
- ConfigureHealthCheck
- CreateAppCookieStickinessPolicy

- `CreateLBCookieStickinessPolicy`
- `CreateLoadBalancer`
- `CreateLoadBalancerListeners`
- `CreateLoadBalancerPolicy`
- `DeleteLoadBalancer`
- `DeleteLoadBalancerListeners`
- `DeleteLoadBalancerPolicy`
- `DeregisterInstancesFromLoadBalancer`
- `DetachLoadBalancersFromSubnets`
- `DisableAvailabilityZonesForLoadBalancer`
- `EnableAvailabilityZonesForLoadBalancer`
- `ModifyLoadBalancerAttributes`
- `RegisterInstancesWithLoadBalancer`
- `RemoveTags`
- `SetLoadBalancerListenerSSLCertificate`
- `SetLoadBalancerPoliciesForBackendServer`
- `SetLoadBalancerPoliciesOfListener`

For more information about global condition keys, see AWS Global Condition Context Keys in the *IAM User Guide*.

The following actions support the `aws:RequestTag/key` and `aws:TagKeys` condition keys:

- `AddTags`
- `CreateLoadBalancer`
- `RemoveTags`

Predefined AWS Managed Policies

The managed policies created by AWS grant the required permissions for common use cases. You can attach these policies to your IAM users, based on the access to Elastic Load Balancing that they require:

- **ElasticLoadBalancingFullAccess** — Grants full access required to use Elastic Load Balancing features.
- **ElasticLoadBalancingReadOnly** — Grants read-only access to Elastic Load Balancing features.

For more information about the permissions required by each Elastic Load Balancing action, see Elastic Load Balancing API Permissions (p. 16).

Elastic Load Balancing API Permissions

You must grant IAM users permission to call the Elastic Load Balancing API actions they need, as described in API Actions for Elastic Load Balancing (p. 11). In addition, for some Elastic Load Balancing actions, you must grant IAM users permission to call specific actions from the Amazon EC2 API.

Required Permissions for the 2015-12-01 API

When calling the following actions from the 2015-12-01 API, you must grant IAM users permission to call the specified actions.

`CreateLoadBalancer`

- `elasticloadbalancing:CreateLoadBalancer`
- `ec2:DescribeAccountAttributes`
- `ec2:DescribeAddresses`
- `ec2:DescribeInternetGateways`
- `ec2:DescribeSecurityGroups`
- `ec2:DescribeSubnets`
- `ec2:DescribeVpcs`
- `iam:CreateServiceLinkedRole`

`CreateTargetGroup`

- `elasticloadbalancing:CreateTargetGroup`
- `ec2:DescribeInternetGateways`
- `ec2:DescribeVpcs`

`RegisterTargets`

- `elasticloadbalancing:RegisterTargets`
- `ec2:DescribeInstances`
- `ec2:DescribeInternetGateways`
- `ec2:DescribeSubnets`
- `ec2:DescribeVpcs`

`SetIpAddressType`

- `elasticloadbalancing:SetIpAddressType`
- `ec2:DescribeSubnets`

`SetSubnets`

- `elasticloadbalancing:SetSubnets`
- `ec2:DescribeSubnets`

Required Permissions for the 2012-06-01 API

When calling the following actions from the 2012-06-01 API, you must grant IAM users permission to call the specified actions.

`ApplySecurityGroupsToLoadBalancer`

- `elasticloadbalancing:ApplySecurityGroupsToLoadBalancer`
- `ec2:DescribeAccountAttributes`
- `ec2:DescribeSecurityGroups`

`AttachLoadBalancerToSubnets`

- `elasticloadbalancing:AttachLoadBalancerToSubnets`
- `ec2:DescribeSubnets`

`CreateLoadBalancer`

- `elasticloadbalancing:CreateLoadBalancer`
- `ec2:CreateSecurityGroup`
- `ec2:DescribeAccountAttributes`
- `ec2:DescribeInternetGateways`
- `ec2:DescribeSecurityGroups`
- `ec2:DescribeSubnets`
- `ec2:DescribeVpcs`

- iam:CreateServiceLinkedRole

DeregisterInstancesFromLoadBalancer

- elasticloadbalancing:DeregisterInstancesFromLoadBalancer
- ec2:DescribeClassicLinkInstances
- ec2:DescribeInstances

DescribeInstanceHealth

- elasticloadbalancing:DescribeInstanceHealth
- ec2:DescribeClassicLinkInstances
- ec2:DescribeInstances

DescribeLoadBalancers

- elasticloadbalancing:DescribeLoadBalancers
- ec2:DescribeSecurityGroups

DisableAvailabilityZonesForLoadBalancer

- elasticloadbalancing:DisableAvailabilityZonesForLoadBalancer
- ec2:DescribeAccountAttributes
- ec2:DescribeInternetGateways
- ec2:DescribeVpcs

EnableAvailabilityZonesForLoadBalancer

- elasticloadbalancing:EnableAvailabilityZonesForLoadBalancer
- ec2:DescribeAccountAttributes
- ec2:DescribeInternetGateways
- ec2:DescribeSubnets
- ec2:DescribeVpcs

RegisterInstancesWithLoadBalancer

- elasticloadbalancing:RegisterInstancesWithLoadBalancer
- ec2:DescribeAccountAttributes
- ec2:DescribeClassicLinkInstances
- ec2:DescribeInstances
- ec2:DescribeVpcClassicLink

Elastic Load Balancing Service-Linked Role

Elastic Load Balancing uses a service-linked role for the permissions that it requires to call other AWS services on your behalf. For more information, see Using Service-Linked Roles in the *IAM User Guide*.

Permissions Granted by the Service-Linked Role

Elastic Load Balancing uses the service-linked role named **AWSServiceRoleForElasticLoadBalancing** to call the following actions on your behalf:

- ec2:DescribeAddresses
- ec2:DescribeInstances
- ec2:DescribeNetworkInterfaces
- ec2:DescribeSubnets
- ec2:DescribeSecurityGroups
- ec2:DescribeVpcs

- `ec2:DescribeInternetGateways`
- `ec2:DescribeAccountAttributes`
- `ec2:DescribeClassicLinkInstances`
- `ec2:DescribeVpcClassicLink`
- `ec2:CreateSecurityGroup`
- `ec2:CreateNetworkInterface`
- `ec2:DeleteNetworkInterface`
- `ec2:ModifyNetworkInterfaceAttribute`
- `ec2:AuthorizeSecurityGroupIngress`
- `ec2:AssociateAddress`
- `ec2:DisassociateAddress`
- `ec2:AttachNetworkInterface`
- `ec2:DetachNetworkInterface`
- `ec2:AssignPrivateIpAddresses`
- `ec2:AssignIpv6Addresses`
- `ec2:UnassignIpv6Addresses`
- `logs:CreateLogDelivery`
- `logs:GetLogDelivery`
- `logs:UpdateLogDelivery`
- `logs:DeleteLogDelivery`
- `logs:ListLogDeliveries`

AWSServiceRoleForElasticLoadBalancing trusts the `elasticloadbalancing.amazonaws.com` service to assume the role.

Create the Service-Linked Role

You don't need to manually create the **AWSServiceRoleForElasticLoadBalancing** role. Elastic Load Balancing creates this role for you when you create a load balancer.

For Elastic Load Balancing to create a service-linked role on your behalf, you must have the required permissions. For more information, see Service-Linked Role Permissions in the *IAM User Guide*.

If you created a load balancer before January 11, 2018, Elastic Load Balancing created **AWSServiceRoleForElasticLoadBalancing** in your AWS account. For more information, see A New Role Appeared in My AWS Account in the *IAM User Guide*.

Edit the Service-Linked Role

You can edit the description of **AWSServiceRoleForElasticLoadBalancing** using IAM. For more information, see Editing a Service-Linked Role in the *IAM User Guide*.

Delete the Service-Linked Role

If you no longer need to use Elastic Load Balancing, we recommend that you delete **AWSServiceRoleForElasticLoadBalancing**.

You can delete this service-linked role only after you delete all load balancers in your AWS account. This ensures that you can't inadvertently remove permission to access your load balancers. For more information, see Delete an Application Load Balancer, Delete a Network Load Balancer, and Delete a Classic Load Balancer.

You can use the IAM console, the IAM CLI, or the IAM API to delete service-linked roles. For more information, see Deleting a Service-Linked Role in the *IAM User Guide*.

After you delete **AWSServiceRoleForElasticLoadBalancing**, Elastic Load Balancing creates the role again if you create a load balancer.

Compliance Validation for Elastic Load Balancing

Third-party auditors assess the security and compliance of Elastic Load Balancing as part of multiple AWS compliance programs. These include SOC, PCI, FedRAMP, HIPAA, and others.

For a list of AWS services in scope of specific compliance programs, see AWS Services in Scope by Compliance Program. For general information, see AWS Compliance Programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

Your compliance responsibility when using Elastic Load Balancing is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- Security and Compliance Quick Start Guides – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- Architecting for HIPAA Security and Compliance Whitepaper – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- Evaluating Resources with Rules in the *AWS Config Developer Guide* – AWS Config; assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

Resilience in Elastic Load Balancing

The AWS global infrastructure is built around AWS Regions and Availability Zones. Regions provide multiple physically separated and isolated Availability Zones, which are connected through low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

Infrastructure Security in Elastic Load Balancing

As a managed service, Elastic Load Balancing is protected by the AWS global network security procedures that are described in the Amazon Web Services: Overview of Security Processes whitepaper.

You use AWS published API calls to access Elastic Load Balancing through the network. Clients must support Transport Layer Security (TLS) 1.0 or later. We recommend TLS 1.2 or later. Clients must also

support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

Network Isolation

A virtual private cloud (VPC) is a virtual network in your own logically isolated area in the AWS Cloud. A subnet is a range of IP addresses in a VPC. When you create a load balancer, you can specify one or more subnets for the load balancer nodes. You can deploy EC2 instances in the subnets of your VPC and register them with your load balancer. For more information about VPC and subnets, see the Amazon VPC User Guide.

When you create a load balancer in a VPC, it can be either internet-facing or internal. An internal load balancer can only route requests that come from clients with access to the VPC for the load balancer.

Your load balancer sends requests to its registered targets using private IP addresses. Therefore, your targets do not need public IP addresses in order to receive requests from a load balancer.

To call the Elastic Load Balancing API from your VPC without sending traffic over the public internet, use AWS PrivateLink. For more information, see Elastic Load Balancing and Interface VPC Endpoints (p. 22).

Controlling Network Traffic

Elastic Load Balancing supports three types of load balancers: Application Load Balancers, Network Load Balancers, and Classic Load Balancers. Application Load Balancers operate at the request level (layer 7) of the Open Systems Interconnection (OSI) model. Network Load Balancers operate at the connection level (layer 4) of the OSI model. Classic Load Balancers operate at both the request and connection levels.

Consider the following options for securing network traffic when you use a load balancer:

- Use secure listeners to support encrypted communication between clients and your load balancers. Application Load Balancers support HTTPS listeners. Network Load Balancers support TLS listeners. Classic Load Balancers support both HTTPS and TLS listeners. You can choose from predefined security policies for your load balancer to specify the cipher suites and protocol versions that are supported by your application. You can use AWS Certificate Manager (ACM) or AWS Identity and Access Management (IAM) to manage the server certificates installed on your load balancer. You can use the Server Name Indication (SNI) protocol to serve multiple secure websites using a single secure listener. SNI is automatically enabled for your load balancer when you associate more than one server certificate with a secure listener.
- Configure the security groups for your Application Load Balancers and Classic Load Balancers to accept traffic only from specific clients. These security groups must allow inbound traffic from clients on the listener ports and outbound traffic to the clients.
- Configure the security groups for your Amazon EC2 instances to accept traffic only from the load balancer. These security groups must allow inbound traffic from the load balancer on the listener ports and the health check ports.
- Configure your Application Load Balancer to securely authenticate users through an identity provider or using corporate identities. For more information, see Authenticate Users Using an Application Load Balancer.
- Use AWS WAF with your Application Load Balancers to allow or block requests based on the rules in a web access control list (web ACL).

Elastic Load Balancing and Interface VPC Endpoints

You can establish a private connection between your virtual private cloud (VPC) and the Elastic Load Balancing API by creating an interface VPC endpoint. You can use this connection to call the Elastic Load Balancing API from your VPC without sending traffic over the internet. The endpoint provides reliable, scalable connectivity to the Elastic Load Balancing API, versions 2015-12-01 and 2012-06-01. It does this without requiring an internet gateway, NAT instance, or VPN connection.

Interface VPC endpoints are powered by AWS PrivateLink, a feature that enables private communication between AWS services using private IP addresses. For more information, see AWS PrivateLink.

Limit

AWS PrivateLink does not support a Network Load Balancer with more than 50 listeners.

Create an Interface Endpoint for Elastic Load Balancing

Create an endpoint for Elastic Load Balancing using one of the following service names:

- **com.amazonaws.*region*.elasticloadbalancing** — Creates an endpoint for the Elastic Load Balancing API operations.
- **com.amazonaws.*region*.elasticloadbalancing-fips** — Creates an endpoint for the Elastic Load Balancing API that complies with the US government standard Federal Information Processing Standard (FIPS) 140-2.

For more information, see Creating an Interface Endpoint in the *Amazon VPC User Guide*.

Create a VPC Endpoint Policy for Elastic Load Balancing

You can attach a policy to your VPC endpoint to control access to the Elastic Load Balancing API. The policy specifies:

- The principal that can perform actions.
- The actions that can be performed.
- The resource on which the actions can be performed.

The following example shows a VPC endpoint policy that denies everyone permission to create a load balancer through the endpoint. The example policy also grants everyone permission to perform all other actions.

```
{
   "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
```

```
        {
            "Action": "elasticloadbalancing:CreateLoadBalancer",
            "Effect": "Deny",
            "Resource": "*",
            "Principal": "*"
        }
    ]
}
```

For more information, see Using VPC Endpoint Policies in the *Amazon VPC User Guide*.

Migrate Your Classic Load Balancer

If you have an existing Classic Load Balancer in a VPC, and you have determined that an Application Load Balancer or a Network Load Balancer would meet your needs, you can migrate your Classic Load Balancer. After you have completed the migration process, you can take advantage of the features of your new load balancer. For more information, see Comparison of Elastic Load Balancing Products.

Migration Process

- Step 1: Create a New Load Balancer (p. 24)
- Step 2: Gradually Redirect Traffic to Your New Load Balancer (p. 26)
- Step 3: Update References to Your Classic Load Balancer (p. 26)
- Step 4: Delete the Classic Load Balancer (p. 27)

Step 1: Create a New Load Balancer

Create an Application Load Balancer or a Network Load Balancer with a configuration that is equivalent to your Classic Load Balancer.

You can create the load balancer and target group using one of the following methods:

- Migration wizard in the console (p. 24)
- Load Balancer Copy Utility (p. 25)
- Manually (p. 25)

Option 1: Migrate Using the Migration Wizard

The migration wizard creates an Application Load Balancer or Network Load Balancer based on the configuration of your Classic Load Balancer. The type of load balancer that is created depends on the configuration of the Classic Load Balancer.

Migration Wizard Release Notes

- The Classic Load Balancer must be in a VPC.
- If the Classic Load Balancer has an HTTP or HTTPS listener, the wizard can create an Application Load Balancer. If the Classic Load Balancer has a TCP listener, the wizard can create a Network Load Balancer.
- If the name of the Classic Load Balancer matches the name of an existing Application Load Balancer or Network Load Balancer, the wizard requires that you specify a different name during migration.
- If the Classic Load Balancer has one subnet, the wizard requires that you specify a second subnet when creating an Application Load Balancer.
- If the Classic Load Balancer has registered instances in EC2-Classic, they are not registered with the target group for the new load balancer.
- If the Classic Load Balancer has registered instances of the following types, they are not registered with the target group for a Network Load Balancer: C1, CC1, CC2, CG1, CG2, CR1, CS1, G1, G2, HI1, HS1, M1, M2, M3, and T1.
- If the Classic Load Balancer has HTTP/HTTPS listeners but uses TCP health checks, the wizard changes to HTTP health checks. Then it sets the path to "/" by default when creating an Application Load Balancer.

- If the Classic Load Balancer is migrated to a Network Load Balancer, the health check settings are changed to meet the requirements for Network Load Balancers.
- If the Classic Load Balancer has multiple HTTPS listeners, the wizard chooses one and uses its certificate and policy. If there is an HTTPS listener on port 443, the wizard chooses this listener. If the listener that is chosen uses a custom policy or a policy not supported for Application Load Balancers, the wizard changes to the default security policy.
- If the Classic Load Balancer has a secure TCP listener, the Network Load Balancer uses a TCP listener. But it does not use the certificate or security policy.
- If the Classic Load Balancer has multiple listeners, the wizard uses the listener port with the lowest value as the target group port. Each instance registered with these listeners is registered with the target group on the listener ports for all the listeners.
- If the Classic Load Balancer has tags with the `aws` prefix in the tag name, these tags are not added to the new load balancer.

To migrate a Classic Load Balancer using the migration wizard

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 your Classic Load Balancer.
4. On the **Migration** tab, choose **Launch ALB Migration Wizard** or **Launch NLB Migration Wizard**. The button that is displayed depends on the load balancer type that was selected by the wizard after examining your Classic Load Balancer.
5. On the **Review** page, verify the configuration options selected by the wizard. To change an option, choose **Edit**.
6. When you are finished configuring the new load balancer, choose **Create**.

Option 2: Migrate Using the Load Balancer Copy Utility

This utility is available on GitHub. For more information, see Load Balancer Copy Utility.

Option 3: Migrate Manually

The following information provides general instructions for manually creating a new load balancer based on a Classic Load Balancer. You can migrate using the AWS Management Console, the AWS CLI, or an AWS SDK. For more information, see Getting Started with Elastic Load Balancing (p. 8).

- Create a new load balancer, with the same scheme (internet-facing or internal), subnets, and security groups as the Classic Load Balancer.
- Create one target group for your load balancer, with the same health check settings that you have for your Classic Load Balancer.
- Do one of the following:
  - If your Classic Load Balancer is attached to an Auto Scaling group, attach your target group to the Auto Scaling group. This also registers the Auto Scaling instances with the target group.
  - Register your EC2 instances with your target group.
- Create one or more listeners, each with a default rule that forwards requests to the target group. If you create an HTTPS listener, you can specify the same certificate that you specified for your Classic Load Balancer. We recommend that you use the default security policy.

- If your Classic Load Balancer has tags, review them and add the relevant tags to your new load balancer.

Step 2: Gradually Redirect Traffic to Your New Load Balancer

After your instances are registered with your new load balancer, you can begin the process of redirecting traffic to it. This allows you to test your new load balancer.

To redirect traffic gradually to your new load balancer

1. Paste the DNS name of your new load balancer into the address field of an internet-connected web browser. If everything is working, the browser displays the default page of your server.
2. Create a new DNS record that associates your domain name with your new load balancer. If your DNS service supports weighting, specify a weight of 1 in the new DNS record and a weight of 9 in the existing DNS record for your Classic Load Balancer. This directs 10% of the traffic to the new load balancer and 90% of the traffic to the Classic Load Balancer.
3. Monitor your new load balancer to verify that it is receiving traffic and routing requests to your instances.

   Important
   The time-to-live (TTL) in the DNS record is 60 seconds. This means that any DNS server that resolves your domain name keeps the record information in its cache for 60 seconds, while the changes propagate. Therefore, these DNS servers can still route traffic to your Classic Load Balancer for up to 60 seconds after you complete the previous step. During propagation, traffic could be directed to either load balancer.
4. Continue to update the weight of your DNS records until all traffic is directed to your new load balancer. When you are finished, you can delete the DNS record for your Classic Load Balancer.

Step 3: Update References to Your Classic Load Balancer

Now that you have migrated your Classic Load Balancer, be sure to update any references to it, such as the following:

- Scripts that use the AWS CLI **aws elb** commands (instead of the **aws elbv2** commands)
- Code that uses Elastic Load Balancing API version 2012-06-01 (instead of version 2015-12-01)
- IAM policies that use API version 2012-06-01 (instead of version 2015-12-01)
- Processes that use CloudWatch metrics
- AWS CloudFormation templates

Resources

- elbv2 in the *AWS CLI Command Reference*
- Elastic Load Balancing API Reference version 2015-12-01
- Identity and Access Management for Elastic Load Balancing (p. 10)
- Application Load Balancer Metrics in the *User Guide for Application Load Balancers*
- Network Load Balancer Metrics in the *User Guide for Network Load Balancers*

- AWS::ElasticLoadBalancingV2::LoadBalancer in the *AWS CloudFormation User Guide*

Step 4: Delete the Classic Load Balancer

You can delete the Classic Load Balancer after:

- You have redirected all traffic to the new load balancer.
- All existing requests that were routed to the Classic Load Balancer have completed.