aws

User Guide

Elastic Load Balancing

Elastic Load Balancing: User Guide

Table of Contents

What is Elastic Load Balancing?

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

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 the following load balancers: Application Load Balancers, Network Load Balancers, Gateway Load Balancers, and Classic Load Balancers. You can select the type of load balancer that best suits your needs. For more information, see Product comparisons.

For more information about using each load balancer, see the following documentation:

- User Guide for Application Load Balancers
- User Guide for Network Load Balancers
- User Guide for Gateway Load Balancers
- 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.
- **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 the following types of load balancers:

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

There is a key difference in how the load balancer types are configured. With Application Load Balancers, Network Load Balancers, and Gateway 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 enabling multiple Availability Zones for all load balancers. With an Application Load Balancer however, it is a requirement that you enable at least two or more 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.

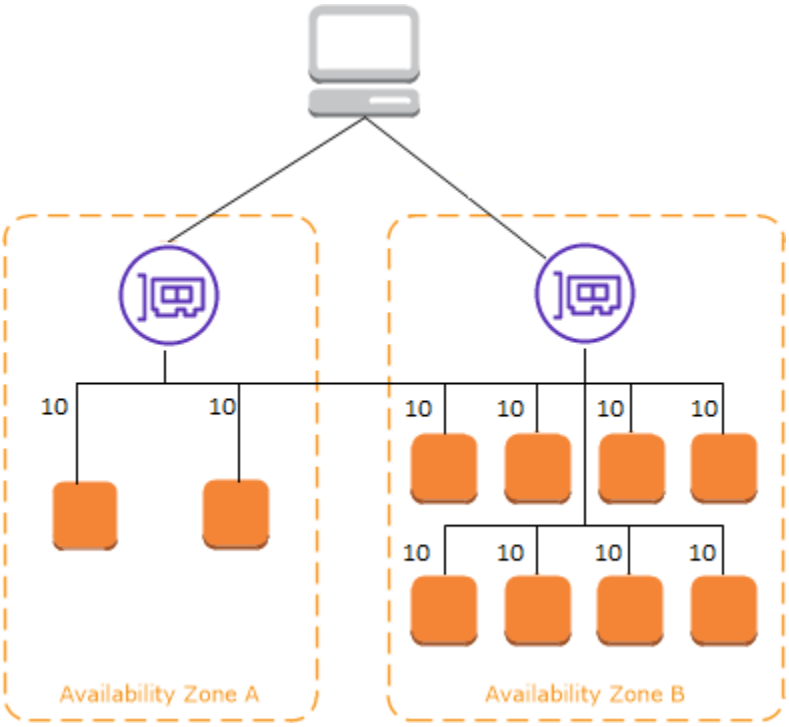
After you disable an Availability Zone, the targets in that Availability Zone remain registered with the load balancer. However, even though they remain registered, the load balancer does not route traffic to them.

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 with round robin as the default routing algorithm. 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. Based on the round robin routing algorithm, traffic is distributed 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.

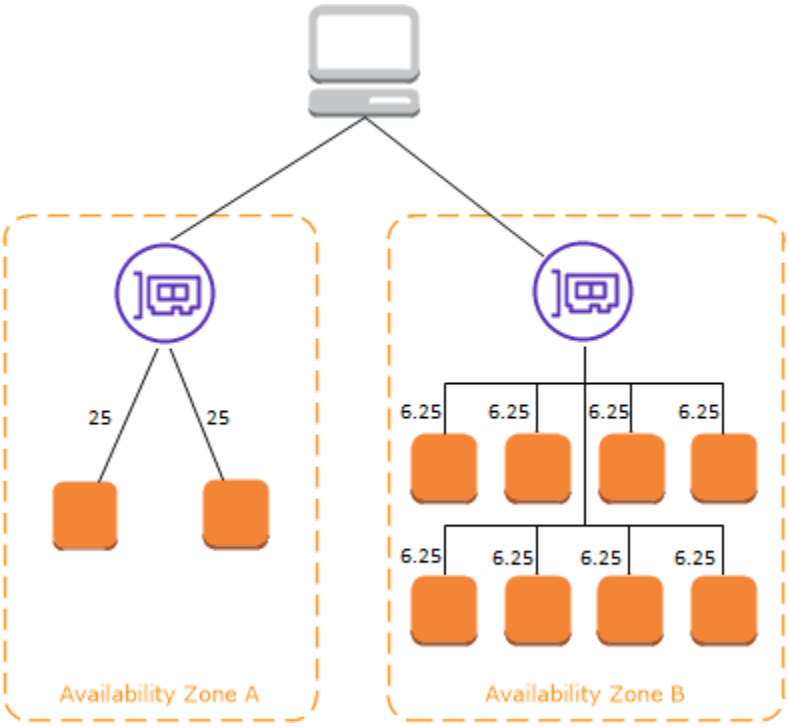
If cross-zone load balancing is enabled, each of the 10 targets receives 10% of the traffic. This is because each load balancer node can route its 50% of the client traffic to all 10 targets.



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 at the load balancer level. At the target group level, cross-zone load balancing can be disabled. For more information, see Turn off cross-zone load balancing in the *User Guide for Application Load Balancers*.

With Network Load Balancers and Gateway Load Balancers, cross-zone load balancing is disabled by default. After you create the load balancer, you can enable or disable cross-zone load balancing at any time.

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*.

Zonal shift

Zonal shift is a capability in Amazon Route 53 Application Recovery Controller (Route 53 ARC). With zonal shift, you can shift a load balancer resource away from an impaired Availability Zone with a single action. This way, you can continue operating from other healthy Availability Zones in an AWS Region.

When you start a zonal shift, your load balancer stops sending traffic for the resource to the affected Availability Zone. Route 53 ARC creates the zonal shift immediately. However, it can take a short time, typically up to a few minutes, to complete existing, in-progress connections in the affected Availability Zone. For more information, see How a zonal shift works: health checks and zonal IP addresses in the *Amazon Route 53 Application Recovery Controller Developer Guide*.

Zonal shifts are only supported on Application Load Balancers and Network Load Balancers with cross-zone load balancing turned off. If you turn on cross-zone load balancing, you can't start a zonal shift. For more information, see Resources supported for zonal shifts in the *Amazon Route 53 Application Recovery Controller Developer Guide*.

Before you use a zonal shift, review the following:

- Cross-zone load balancing isn't supported with zonal shifts. You must turn off cross-zone load balancing to use this capability.
- Zonal shift isn't supported when you use an Application Load Balancer as an accelerator endpoint in AWS Global Accelerator.
- You can start a zonal shift for a specific load balancer only for a single Availability Zone. You can't start a zonal shift for multiple Availability Zones.
- AWS proactively removes zonal load balancer IP addresses from DNS when multiple infrastructure issues impact services. Always check current Availability Zone capacity before you start a zonal shift. If your load balancers have cross-zone load balancing turned off and you use a zonal shift to remove a zonal load balancer IP address, the Availability Zone affected by the zonal shift also loses target capacity.
- When an Application Load Balancer is a target of a Network Load Balancer, always start the zonal shift from the Network Load Balancer. If you start a zonal shift from the Application Load Balancer, the Network Load Balancer doesn't recognize the shift and continues to send traffic to the Application Load Balancer.

For more guidance and information, see Best practices with Route 53 ARC zonal shifts in the *Amazon Route 53 Application Recovery Controller Developer Guide*.

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, and uses it to get a static IP address. You can optionally associate one Elastic IP address with each network interface when you create the Network 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.

For more information, see Routing traffic to an ELB load balancer in the *Amazon Route 53 Developer Guide*.

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-alive` headers by setting the `Connection: close` header in your HTTP responses.

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

Application Load Balancers support the following HTTP request methods: GET, HEAD, POST, PUT, DELETE, OPTIONS, and PATCH.

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. However, if there is a connection upgrade, Application Load Balancer listener routing rules and AWS WAF integrations no longer apply.

Application Load Balancers use HTTP/1.1 on backend connections (load balancer to registered target) by default. However, you can use the protocol version to send the request to the targets using HTTP/2 or gRPC. For more information, see Protocol versions. The `keep-alive` header 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. The host header contains the DNS name of the load balancer.

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. They use HTTP/1.1 on backend connections (load balancer to registered target). The `keep-alive` header 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. The host header contains the IP address of the load balancer node.

HTTP headers

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

Application Load Balancers convert the hostnames in HTTP host headers to lower case before sending them to targets.

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.

When Application Load Balancers and Classic Load Balancers using HTTP/1.1 receive an **Expect: 100-Continue** header, they immediately respond with **HTTP/1.1 100 Continue** without testing the content length header. The **Expect: 100-Continue** request header is not forwarded to it's targets.

When using HTTP/2, Application Load Balancers do not support the **Expect: 100-Continue** header from client requests. The Application Load Balancer will not respond with **HTTP/2 100 Continue** or forward this header to its targets.

HTTP header limits

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

- Request line: 16 K
- Single header: 16 K
- Entire response header: 32 K
- Entire request 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.

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 application 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 application servers with it. The web servers receive requests from the internet-facing load balancer and send requests for the application servers to the internal load balancer. The application servers receive requests from the internal load balancer.

Network MTU for your load balancer

The maximum transmission unit (MTU) determines the size, in bytes, for the largest packet that can be sent over the network. The larger the MTU of a connection, the more data that can be passed in a single packet. Ethernet frames consist of the packet, or the actual data you are sending, and the network overhead information information that surrounds it. Traffic sent over an internet gateway has an MTU of 1500. This means that if a packet is over 1500 bytes, it is fragmented to be sent using multiple frames, or it is dropped if the `Don't Fragment` is set in the IP header.

The MTU size on load balancer nodes is not configurable. Jumbo frames (9001 MTU) are standard across load balancer nodes for Application Load Balancers, Network Load Balancers, and Classic Load Balancers. Gateway Load Balancers support 8500 MTU. For more information, see Maximum transmission unit (MTU) in the *User Guide for Gateway Load Balancers*.

The path MTU is the maximum packet size that is supported on the path between the originating host and the receiving host. Path MTU Discovery (PMTUD) is used to determine the path MTU between two devices. Path MTU Discovery is especially important if the client or target does not support jumbo frames.

When a host sends a packet that is larger than the MTU of the receiving host or larger than the MTU of a device along the path, the receiving host or device drops the packet, and then returns the following ICMP message: `Destination Unreachable: Fragmentation Needed and Don't Fragment was Set (Type 3, Code 4)`. This instructs the transmitting host to split the payload into multiple smaller packets, and retransmit them.

If packets larger than the MTU size of the client or target interface continue to be dropped, it is likely that Path MTU Discovery (PMTUD) is not working. To avoid this, ensure that Path MTU Discovery is working end to end, and that you have enabled jumbo frames on your clients and targets. For more information about Path MTU Discovery and enabling jumbo frames, see Path MTU Discovery in the *Amazon EC2 User Guide*.

Getting started with Elastic Load Balancing

Elastic Load Balancing supports the following load balancers: Application Load Balancers, Network Load Balancers, Gateway Load Balancers, and Classic Load Balancers. You can select the type of load balancer that best suits your needs. For more information, see Product comparisons.

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.

Contents

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 Gateway Load Balancer

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

To create a Gateway Load Balancer using the AWS CLI, see Getting started with Gateway Load Balancers using the AWS CLI in the *User Guide for Gateway 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.

With a Gateway Load Balancer, you are responsible for choosing and qualifying software from appliance vendors. You must trust the appliance software to inspect or modify traffic from the load balancer, which operates at the layer 3 of the Open Systems Interconnection (OSI) model, the network layer. The appliance vendors listed as Elastic Load Balancing Partners have integrated and qualified their appliance software with AWS. You can place a higher degree of trust in the appliance software from vendors in this list. However, AWS does not guarantee the security or reliability of software from these vendors.

Contents

- Infrastructure security in Elastic Load Balancing
- Access Elastic Load Balancing using an interface endpoint (AWS PrivateLink)

Data protection in Elastic Load Balancing

The AWS shared responsibility model applies to data protection in Elastic Load Balancing. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, 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 SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- 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 sensitive data that is stored in Amazon S3.
- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with Elastic Load Balancing or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

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 KMS 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 Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Elastic Load Balancing resources. IAM is an AWS service that you can use with no additional charge.

Contents

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in Elastic Load Balancing.

Service user – If you use the Elastic Load Balancing service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more Elastic Load Balancing features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator.

Service administrator – If you're in charge of Elastic Load Balancing resources at your company, you probably have full access to Elastic Load Balancing. It's your job to determine which Elastic Load Balancing features and resources your service users should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM.

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to Elastic Load Balancing.

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be *authenticated* (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (IAM Identity Center) users, your company's single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see Signing AWS API requests in the *IAM User Guide*.

Regardless of the authentication method that you use, you might be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Multi-factor authentication in the *AWS IAM Identity Center User Guide* and Using multi-factor authentication (MFA) in AWS in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

As a best practice, require human users, including users that require administrator access, to use federation with an identity provider to access AWS services by using temporary credentials.

A *federated identity* is a user from your enterprise user directory, a web identity provider, the AWS Directory Service, the Identity Center directory, or any user that accesses AWS services by using credentials provided through an identity source. When federated identities access AWS accounts, they assume roles, and the roles provide temporary credentials.

For centralized access management, we recommend that you use AWS IAM Identity Center. You can create users and groups in IAM Identity Center, or you can connect and synchronize to a set of users and groups in your own identity source for use across all your AWS accounts and applications. For information about IAM Identity Center, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see Rotate access keys regularly for use cases that require long-term credentials in the *IAM User Guide*.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see When to create an IAM user (instead of a role) in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by switching roles. You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Using IAM roles in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Creating a role for a third-party Identity Provider in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see Permission sets in the *AWS IAM Identity Center User Guide*.
- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or

store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.

- **Forward access sessions (FAS)** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.
- **Service role** – A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.
- **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

To learn whether to use IAM roles or IAM users, see When to create an IAM role (instead of a user) in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most

policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choosing between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access control list (ACL) overview in the *Amazon Simple Storage Service Developer Guide*.

Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of an entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see How SCPs work in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How Elastic Load Balancing works with IAM

Before you use IAM to manage access to Elastic Load Balancing, learn what IAM features are available to use with Elastic Load Balancing.

IAM features you can use with Elastic Load Balancing

IAM feature	Elastic Load Balancing support
Identity-based policies	Yes
Resource-based policies	No
Policy actions	Yes
Policy resources	Yes
Policy condition keys (service-specific)	Yes
ACLs	No
ABAC (tags in policies)	Yes
Temporary credentials	Yes
Principal permissions	Yes
Service roles	No
Service-linked roles	Yes

Identity-based policies for Elastic Load Balancing

Supports identity-based policies	Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. You can't specify the principal in an identity-based policy because it applies to the user or role to which it is attached. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Resource-based policies within Elastic Load Balancing

Supports resource-based policies	No

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship. When the principal and the resource are in different AWS accounts, an IAM administrator in the trusted account must also grant the principal entity (user or role) permission to access the resource. They grant permission by attaching an identity-based policy to the entity. However, if a resource-based policy grants access to a principal in the same account, no additional identity-based policy is required. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Policy actions for Elastic Load Balancing

Supports policy actions	Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

To see a list of Elastic Load Balancing actions, see Actions defined by Elastic Load Balancing in the *Service Authorization Reference*.

Policy actions in Elastic Load Balancing use the following prefix before the action:

```
elasticloadbalancing
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [
    "elasticloadbalancing:action1",
    "elasticloadbalancing:action2"
]
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word `Describe`, include the following action:

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

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

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

For more information about the permissions required by each Elastic Load Balancing action, see Elastic Load Balancing API permissions.

Policy resources for Elastic Load Balancing

Supports policy resources	Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. As a best practice, specify a resource using its Amazon Resource Name (ARN). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

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

Some Elastic Load Balancing API actions support multiple resources. To specify multiple resources in a single statement, separate the ARNs with commas.

```
"Resource": [
    "resource1",
    "resource2"
]
```

To see a list of Elastic Load Balancing resource types and their ARNs, see Resources defined by Elastic Load Balancing in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see Actions defined by Elastic Load Balancing.

Policy condition keys for Elastic Load Balancing

Supports service-specific policy condition keys	Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element (or `Condition` *block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical `AND` operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical `OR` operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM policy elements: variables and tags in the *IAM User Guide*.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

To see a list of Elastic Load Balancing condition keys, see Condition keys for Elastic Load Balancing in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see Actions defined by Elastic Load Balancing.

`elasticloadbalancing:ResourceTag` condition key

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

`elasticloadbalancing:ListenerProtocol` condition key

The `elasticloadbalancing:ListenerProtocol` condition key can be used for conditions that define the types of listeners that can be created and used. The following actions support this condition key:

API version 2015-12-01

- `CreateListener`
- `ModifyListener`

API version 2012-06-01

- `CreateLoadBalancer`
- `CreateLoadBalancerListeners`

The policy is available for Application Load Balancers, Network Load Balancers, and Classic Load Balancers. The following is an example policy that only allows users to select one of the specified protocols for their listener.

Supported protocols:

- HTTPS
- HTTP
- TCP
- SSL
- TLS
- UDP
- TCP_UDP

```
"Version": "2015-12-01",
    "Statement": {"Effect": "Allow",
        "Action": [
            "elasticloadbalancing:CreateListener",
            "elasticloadbalancing:ModifyListener"
        ],
        "Resource": "*",
```

```
        "Condition": {
            "StringEquals":{
                    "elasticloadbalancing:ListenerProtocol": [
                    "HTTPS",
                    "TLS"
                    ]
            },
        }
```

`elasticloadbalancing:SecurityPolicy` condition key

The `elasticloadbalancing:SecurityPolicy` condition key can be used for conditions that define and enforce specific security policies on the load balancers. The following actions support this condition key:

API version 2015-12-01

- `CreateListener`
- `ModifyListener`

API version 2012-06-01

- `CreateLoadBalancerPolicy`
- `SetLoadBalancerPoliciesOfListener`

The policy is available for Application Load Balancers, Network Load Balancers and Classic Load Balancers. The following is an example policy that only allows users to select one of the specified security policies for their load balancer.

```
"Resource": [
"Version": "2015-12-01",
    "Statement": {"Effect": "Allow",
        "Action": [
            "elasticloadbalancing:CreateListener",
            "elasticloadbalancing:ModifyListener"
        ],
        "Resource": "*",
        "Condition": {
            "ForAnyValue:StringEquals":{
                "elasticloadbalancing:SecurityPolicy": [
```

```
                    "ELBSecurityPolicy-TLS13-1-2-2021-06",
                    "ELBSecurityPolicy-TLS13-1-2-Res-2021-06",
                    "ELBSEcurityPolicy-TLS13-1-1-2021-06"
                ]
            },
        }
]
```

`elasticloadbalancing:Scheme` condition key

The `elasticloadbalancing:Scheme` condition key can be used for conditions that define which scheme can be selected during load balancer creation. The following actions support this condition key:

API version 2015-12-01

- `CreateLoadBalancer`

API version 2012-06-01

- `CreateLoadBalancer`

The policy is available for Application Load Balancers, Network Load Balancers, and Classic Load Balancers. The following is an example policy that only allows users to select one of the specified scheme for their load balancer.

```
"Version": "2015-12-01",
    "Statement": {"Effect": "Allow",
        "Action":"elasticloadbalancing:CreateLoadBalancer",
        "Resource": "*",
        "Condition": {
            "StringEquals":{
                "elasticloadbalancing:Scheme": "internal"
            },
        }
```

`elasticloadbalancing:Subnet` condition key

> **Important**
>
> Elastic Load Balancing accepts all capitalizations of Subnet IDs. However, make sure to use the appropriate case insensitive condition operators, for example `StringEqualsIgnoreCase`.

The `elasticloadbalancing:Subnet` condition key can be used for conditions that define which subnets can be created and attached to load balancers. The following actions support this condition key:

API version 2015-12-01

- `CreateLoadBalancer`
- `SetSubnets`

API version 2012-06-01

- `CreateLoadBalancer`
- `AttachLoadBalancerToSubnets`

The policy is available for Application Load Balancers, Network Load Balancers, Gateway Load Balancers and Classic Load Balancers. The following is an example policy that only allows users to select one of the specified subnets for their load balancer.

```
"Version": "2015-12-01",
    "Statement": {"Effect": "Allow",
        "Action": [
            "elasticloadbalancing:CreateLoadBalancer",
            "elasticloadbalancing:SetSubnets"
        ],
        "Resource": "*",
        "Condition": {
            "ForAnyValue:StringEqualsIgnoreCase":{
                "elasticloadbalancing:Subnet": [
                    "subnet-01234567890abcdef",
                    "subnet-01234567890abcdeg "
```

```
                ]
            },
        }
```

`elasticloadbalancing:SecurityGroup` condition key

> **Important**
>
> Elastic Load Balancing accepts all capitalizations of SecurityGroup IDs. However, make sure to use the appropriate case insensitive condition operators, for example `StringEqualsIgnoreCase`.

The `elasticloadbalancing:SecurityGroup` condition key can be used for conditions that define which security groups can be applied to the load balancers. The following actions support this condition key:

API version 2015-12-01

- `CreateLoadBalancer`
- `SetSecurityGroups`

API version 2012-06-01

- `CreateLoadBalancer`
- `ApplySecurityGroupsToLoadBalancer`

The policy is available for Application Load Balancers, Network Load Balancers and Classic Load Balancers. The following is an example policy that only allows users to select one of the specified security groups for their load balancer.

```
"Version": "2015-12-01",
    "Statement": {"Effect": "Allow",
        "Action": [
            "elasticloadbalancing:CreateLoadBalancer",
            "elasticloadbalancing:SetSecurityGroup"
        ],
        "Resource": "*",
        "Condition": {
```

```
            "ForAnyValue:StringEqualsIgnoreCase":{
                "elasticloadbalancing:SecurityGroup": [
                    "sg-51530134",
                    "sg-51530144",
                    "sg-51530139"
                ]
            },
        }
```

ACLs in Elastic Load Balancing

Supports ACLs	No

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

ABAC with Elastic Load Balancing

Supports ABAC (tags in policies)	Yes

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see What is ABAC? in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

Using temporary credentials with Elastic Load Balancing

Supports temporary credentials	Yes

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see AWS services that work with IAM in the *IAM User Guide*.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see Switching to a role (console) in the *IAM User Guide*.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM.

Cross-service principal permissions for Elastic Load Balancing

Supports forward access sessions (FAS)	Yes

When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.

Service roles for Elastic Load Balancing

Supports service roles	No

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.

Service-linked roles for Elastic Load Balancing

Supports service-linked roles	Yes

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing Elastic Load Balancing service-linked roles, see Elastic Load Balancing service-linked role.

Elastic Load Balancing API permissions

You must grant users permission to call the Elastic Load Balancing API actions they need. In addition, for some Elastic Load Balancing actions, you must grant 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 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 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 API permissions to tag resources during creation

For users to tag resources during creation, they must have permissions to use the action that creates the resource, such as elasticloadbalancing:CreateLoadBalancer or elasticloadbalancing:CreateTargetGroup. If tags are specified in the resource-creating action, additional authorization is required on the elasticloadbalancing:AddTags action to verify if users have permissions to apply tags to the resources being created. Therefore, users must also have explicit permissions to use the elasticloadbalancing:AddTags action.

In the IAM policy definition for the elasticloadbalancing:AddTags action, you can use the Condition element with the elasticloadbalancing:CreateAction condition key to give tagging permissions to the action that creates the resource.

The following example demonstrates a policy that allows users to create target groups and apply any tags to them during creation. Users are not permitted to tag any existing resources (they can't call the elasticloadbalancing:AddTags action directly).

```
{
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Action": [
         "elasticloadbalancing:CreateTargetGroup"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
         "elasticloadbalancing:AddTags"
      ],
      "Resource": "*",
      "Condition": {
         "StringEquals": {
             "elasticloadbalancing:CreateAction" : "CreateTargetGroup"
          }
       }
    }
  ]
}
```

Similarly, the following policy allows users to create a load balancer and apply tags during creation. Users are not permitted to tag any existing resources (they can't call the `elasticloadbalancing:AddTags` action directly).

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
         "elasticloadbalancing:CreateLoadBalancer"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
         "elasticloadbalancing:AddTags"
      ],
      "Resource": "*",
      "Condition": {
         "StringEquals": {
```

```
            "elasticloadbalancing:CreateAction" : "CreateLoadBalancer"
          }
        }
      }
    ]
}
```

The `elasticloadbalancing:AddTags` action is only evaluated if tags are applied during the resource-creating action. Therefore, a user that has permissions to create a resource (assuming there are no tagging conditions) does not require permissions to use the `elasticloadbalancing:AddTags` action if no tags are specified in the request. However, if the user attempts to create a resource with tags, the request fails if the user does not have permissions to use the `elasticloadbalancing:AddTags` action.

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:AssignIpv6Addresses`
- `ec2:AssignPrivateIpAddresses`
- `ec2:AssociateAddress`
- `ec2:AttachNetworkInterface`
- `ec2:AuthorizeSecurityGroupIngress`
- `ec2:CreateNetworkInterface`
- `ec2:CreateSecurityGroup`
- `ec2:DeleteNetworkInterface`
- `ec2:DescribeAccountAttributes`
- `ec2:DescribeAddresses`

- ec2:DescribeClassicLinkInstances
- ec2:DescribeCoipPools
- ec2:DescribeInstances
- ec2:DescribeInternetGateways
- ec2:DescribeNetworkInterfaces
- ec2:DescribeSecurityGroups
- ec2:DescribeSubnets
- ec2:DescribeVpcClassicLink
- ec2:DescribeVpcPeeringConnections
- ec2:DescribeVpcs
- ec2:DetachNetworkInterface
- ec2:DisassociateAddress
- ec2:GetCoipPoolUsage
- ec2:ModifyNetworkInterfaceAttribute
- ec2:ReleaseAddress
- ec2:UnassignIpv6Addresses
- logs:CreateLogDelivery
- logs:DeleteLogDelivery
- logs:GetLogDelivery
- logs:ListLogDeliveries
- logs:UpdateLogDelivery
- outposts:GetOutpostInstanceTypes

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 or a target group.

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.

AWS managed policies for Elastic Load Balancing

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the *IAM User Guide*.

AWS managed policy: AWSElasticLoadBalancingClassicServiceRolePolicy

This policy includes all the permissions that Elastic Load Balancing (Classic Load Balancer) requires to call other AWS services on your behalf. Service-linked roles are predefined. With predefined roles you don't have to manually add the necessary permissions for Elastic Load Balancing to complete actions on your behalf. You cannot attach, detach, modify, or delete this policy.

To view the permissions for this policy, see AWSElasticLoadBalancingClassicServiceRolePolicy in the *AWS Managed Policy Reference*.

AWS managed policy: AWSElasticLoadBalancingServiceRolePolicy

This policy includes all the permissions that Elastic Load Balancing requires to call other AWS services on your behalf. Service-linked roles are predefined. With predefined roles you don't have to manually add the necessary permissions for Elastic Load Balancing to complete actions on your behalf. You cannot attach, detach, modify, or delete this policy.

To view the permissions for this policy, see AWSElasticLoadBalancingServiceRolePolicy in the *AWS Managed Policy Reference*.

AWS managed policy: ElasticLoadBalancingFullAccess

This policy gives full access to the Elastic Load Balancing service and limited access to other services via the AWS Management Console.

To view the permissions for this policy, see ElasticLoadBalancingFullAccess in the *AWS Managed Policy Reference*.

AWS managed policy: ElasticLoadBalancingReadOnly

This policy provides read-only access to Elastic Load Balancing and dependent services.

To view the permissions for this policy, see ElasticLoadBalancingReadOnly in the *AWS Managed Policy Reference*.

Elastic Load Balancing updates to AWS managed policies

View details about updates to AWS managed policies for Elastic Load Balancing since this service began tracking these changes.

Change	Description	Date
AWS managed policy: ElasticLoadBalancingFullAccess - Update to an existing policy.	Elastic Load Balancing added a new action to grant permissions to use zonal shift. This action was added to the Elastic Load Balancing full access policy. It is associated with the `arc-zonal-shift:*` API operations.	November 28, 2022
AWS managed policy: ElasticLoadBalancingReadOnly - Update to an existing policy.	Elastic Load Balancing added a new action to grant permissions to use zonal shift. This action was added to the Elastic Load Balancing read only policy. It is associated with the `arc-zonal-shift:GetManagedResource`, `arc-zonal-shift:ListManagedResources` and `arc-zonal-shift:ListZonalShifts` API operations.	November 28, 2022
AWS managed policy: AWSElasticLoadBalancingServiceRolePolicy - Update to an existing policy.	Elastic Load Balancing added a new action to grant permissions to use peering connections. This action was added to the service-linked role policy, for Elastic Load Balancing control plane. It is associated with the `ec2:DescribeVpcPeeringConnections` API operation.	October 11, 2021
AWS managed policy: ElasticLoadBalancingFullAccess - Update to an existing policy.	Elastic Load Balancing added a new action to grant permissions to use peering connections. This action was added to the Elastic Load Balancing full access policy. It is	October 11, 2021

Change	Description	Date
	associated with the `ec2:DescribeVpcPeeringConnections` API operation.	
AWS managed policy: AWSElasticLoadBalancingClassicServiceRolePolicy - Update to an existing policy.	Elastic Load Balancing added a service-linked role policy (for the control plane) for the Classic Load Balancer. This update is for version 2 (default).	October 7, 2019
AWS managed policy: ElasticLoadBalancingReadOnly	Provides read-only access to Elastic Load Balancing and dependent services. This is the version 1 (default).	September 20, 2018
Elastic Load Balancing started tracking changes	Elastic Load Balancing started tracking changes for its AWS managed policies.	July 23, 2021

Compliance validation for Elastic Load Balancing

To learn whether an AWS service is within the scope of specific compliance programs, see AWS services in Scope by Compliance Program and choose the compliance program that you are interested in. 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 AWS services 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 baseline environments on AWS that are security and compliance focused.
- Architecting for HIPAA Security and Compliance on Amazon Web Services – This whitepaper describes how companies can use AWS to create HIPAA-eligible applications.

> **Note**
>
> Not all AWS services are HIPAA eligible. For more information, see the HIPAA Eligible Services Reference.

- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- AWS Customer Compliance Guides – Understand the shared responsibility model through the lens of compliance. The guides summarize the best practices for securing AWS services and map the guidance to security controls across multiple frameworks (including National Institute of Standards and Technology (NIST), Payment Card Industry Security Standards Council (PCI), and International Organization for Standardization (ISO)).
- Evaluating Resources with Rules in the *AWS Config Developer Guide* – The AWS Config service 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. Security Hub uses security controls to evaluate your AWS resources and to check your compliance against security industry standards and best practices. For a list of supported services and controls, see Security Hub controls reference.
- Amazon GuardDuty – This AWS service detects potential threats to your AWS accounts, workloads, containers, and data by monitoring your environment for suspicious and malicious activities. GuardDuty can help you address various compliance requirements, like PCI DSS, by meeting intrusion detection requirements mandated by certain compliance frameworks.
- AWS Audit Manager – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

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.

In addition to the AWS Global Infrastructure, Elastic Load Balancing provides the following features to support your data resiliency:

- Distributes incoming traffic across multiple instances in a single Availability Zone or multiple Availability Zones.
- You can use AWS Global Accelerator with your Application Load Balancers to distribute incoming traffic across multiple load balancers in one or more AWS Regions. For more information, see the AWS Global Accelerator Developer Guide.
- Amazon ECS enables you to run, stop, and manage Docker containers on a cluster of EC2 instances. You can configure your Amazon ECS service to use a load balancer to distribute incoming traffic across the services in a cluster. For more information, see the Amazon Elastic Container Service Developer Guide.

Infrastructure security in Elastic Load Balancing

As a managed service, Elastic Load Balancing is protected by AWS global network security. For information about AWS security services and how AWS protects infrastructure, see AWS Cloud Security. To design your AWS environment using the best practices for infrastructure security, see Infrastructure Protection in *Security Pillar AWS Well-Architected Framework*.

You use AWS published API calls to access Elastic Load Balancing through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by 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 using private IP addresses, use AWS PrivateLink. For more information, see Access Elastic Load Balancing using an interface endpoint (AWS PrivateLink).

Controlling network traffic

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).

Access Elastic Load Balancing using an interface endpoint (AWS PrivateLink)

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 requiring that you attach an internet gateway, NAT instance, or VPN connection to your VPC. The endpoint provides reliable, scalable connectivity to the Elastic Load Balancing API, versions 2015-12-01 and 2012-06-01, which you use to create and manage your load balancers.

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

Limit

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

Create an interface endpoint for Elastic Load Balancing

Create an endpoint for Elastic Load Balancing using the following service name:

```
com.amazonaws.region.elasticloadbalancing
```

For more information, see Create an interface endpoint in the *AWS PrivateLink 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 Control access to services using endpoint policies in the *AWS PrivateLink Guide*.

Migrate your Classic Load Balancer

Elastic Load Balancing supports the following types of load balancers: Application Load Balancers, Network Load Balancers, Gateway Load Balancers, and Classic Load Balancers. For information about the different features of each load balancer type, see Comparison of Elastic Load Balancing products.

You can also choose to migrate an existing Classic Load Balancer in a VPC, to an Application Load Balancer or a Network Load Balancer.

Benefits of migrating from a Classic Load Balancer

Each type of load balancer has its own unique features, functions, and configurations. Review the benefits of each load balancer to help decide which one is best for you.

Application Load Balancer

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

Support for:

- Path conditions, Host conditions, and HTTP header conditions.
- Redirecting requests from one URL to another, and routing requests to multiple applications on a single EC2 instance.
- Returning custom HTTP responses.
- Registering targets by IP address, and registering Lambda functions as targets. Including targets outside the VPC for the load balancer.
- Authenticating users through corporate or social identities.
- Amazon Elastic Container Service (Amazon ECS) containerized applications.
- Independently monitoring the health of each service.

Access logs contain additional information and are stored in a compressed format.

Improved load balancer performance overall.

Network Load Balancer

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

Support for:

- Static IP addresses, which allow assigning one Elastic IP address per subnet enabled for the load balancer.
- Registering targets by IP address, including targets outside the VPC for the load balancer.
- Routing requests to multiple applications on a single EC2 instance.
- Amazon Elastic Container Service (Amazon ECS) containerized applications.
- Independently monitoring the health of each service.

Ability to handle volatile workloads and scale to millions of requests per second.

Migrate using migration wizard

Migration wizard uses the configuration of your Classic Load Balancer to create an equivalent Application Load Balancer or Network Load Balancer. It reduces the time and effort required to migrate a Classic Load Balancer compared to other methods.

> **Note**
>
> The wizard creates a new load balancer. The wizard doesn't convert the existing Classic Load Balancer to an Application Load Balancer or Network Load Balancer. You must manually redirect the traffic to the newly created load balancer.

Limitations

- The name of the new load balancer can't be the same as an existing load balancer of the same type, in the same region.
- If the Classic Load Balancer has any tags containing the `aws:` prefix in their key, those tags are not migrated.

When migrating to an Application Load Balancer

- If the Classic Load Balancer has only one subnet, you must specify a second subnet.
- If the Classic Load Balancer has HTTP/HTTPS listeners that use TCP health checks, the health check protocol is updated to HTTP and the path is set to "/".
- If the Classic Load Balancer has HTTPS listeners using a custom or unsupported security policy, migration wizard uses the default security policy for the new load balancer type.

When migrating to a Network Load Balancer

- The following instance types will not be registered with the new target group: C1, CC1, CC2, CG1, CG2, CR1, CS1, G1, G2, HI1, HS1, M1, M2, M3, T1
- Certain health check settings from your Classic Load Balancer may not be transferrable to the new target group. These cases will be indicated as a change in the summary section of the migration wizard.
- If the Classic Load Balancer has SSL listeners, migration wizard creates a TLS listener using the certificate and security policy from the SSL listener.

Migration wizard process

To migrate a Classic Load Balancer using 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 the Classic Load Balancer you want to migrate.
4. In the load balancers **Details** section, choose **Launch migration wizard**.
5. Choose **Migrate to Application Load Balancer**, or **Migrate to Network Load Balancer**, to open migration wizard.
6. Under **Name new load balancer**, for **Load balancer name** enter a name for your new load balancer.
7. Under **Name new target group and review targets**, for **Target group name** enter a name for your new target group.
8. (Optional) Under **Targets**, you can review the target instances that will be registered with the new target group.

9. (Optional) Under **Review tags**, you can review the tags that will be applied to your new load balancer
10. Under **Summary for Application Load Balancer**, or **Summary for Network Load Balancer**, review and verify the configuration options assigned by migration wizard.
11. After you're satisfied with the configuration summary, choose **Create Application Load Balancer**, or **Create Network Load Balancer**, to start the migration.

Migrate using the load balancer copy utility

The load balancer copy utilities are available within the Elastic Load Balancing Tools repository, on the AWS GitHub page.

Resources

- Elastic Load Balancing Tools
- Classic Load Balancer to Application Load Balancer copy utility
- Classic Load Balancer to Network Load Balancer copy utility

Migrate your load balancer manually

The following information provides general instructions for manually creating a new Application Load Balancer or Network Load Balancer based on an existing Classic Load Balancer in a VPC. 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.

After you have completed the migration process, you can take advantage of the features of your new load balancer.

Manual migration process

Step 1: Create a new load balancer

Create a load balancer with a configuration that is equivalent to the Classic Load Balancer to migrate.

1. Create a new load balancer, with the same scheme (internet-facing or internal), subnets, and security groups as the Classic Load Balancer.

2. Create one target group for your load balancer, with the same health check settings that you have for your Classic Load Balancer.
3. 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.
4. 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.
5. 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 from the old load balancer to the new load balancer. This enables you to test your new load balancer while minimizing risk to the availability of your application.

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 application.
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 old load balancer. This directs 10% of the traffic to the new load balancer and 90% of the traffic to the old 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 old 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 old load balancer.

Step 3: Update policies, scripts, and code

If you migrated your Classic Load Balancer to an Application Load Balancer or Network Load Balancer, be sure to do the following:

- Update IAM policies that use API version 2012-06-01 to use version 2015-12-01.
- Update processes that use CloudWatch metrics in the `AWS/ELB` namespace to use metrics from the `AWS/ApplicationELB` or `AWS/NetworkELB` namespace.
- Update scripts that use **aws elb** AWS CLI commands to use **aws elbv2** AWS CLI commands.
- Update AWS CloudFormation templates that use the `AWS::ElasticLoadBalancing::LoadBalancer` resource to use the `AWS::ElasticLoadBalancingV2` resources.
- Update code that uses Elastic Load Balancing API version 2012-06-01 to use version 2015-12-01.

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
- 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 old load balancer

You can delete the old Classic Load Balancer after:

- You have redirected all traffic from the old load balancer to the new load balancer.
- All existing requests that were routed to the old load balancer have completed.