Elastic Load Balancing

User Guide

Elastic Load Balancing: User Guide

Table of Contents

What Is Elastic Load Balancing?

Elastic Load Balancing distributes incoming application traffic across multiple EC2 instances, in multiple Availability Zones. This increases the fault tolerance of your applications.

The load balancer serves as a single point of contact for clients, which increases the availability of your application. You can add and remove instances from your load balancer as your needs change, without disrupting the overall flow of requests to your application. Elastic Load Balancing scales your load balancer as traffic to your application changes over time, and can scale to the vast majority of workloads automatically.

You can configure health checks, which are used to monitor the health of the registered instances so that the load balancer can send requests only to the healthy instances. You can also offload the work of encryption and decryption to your load balancer so that your instances can focus on their main work.

Features of Elastic Load Balancing

Elastic Load Balancing supports two types of load balancers: Application Load Balancers and Classic Load Balancers. Choose the load balancer type that meets your needs.

Feature	Classic Load Balancer	Application Load Balancer
Protocols	HTTP, HTTPS, TCP, SSL	HTTP, HTTPS
Platforms	EC2-Classic, EC2-VPC	EC2-VPC
Sticky sessions (cookies)	✔	load balancer generated
Back-end server authentication	✔	
Back-end server encryption	✔	✔
Idle connection timeout	✔	✔
Connection draining	✔	✔
Cross-zone load balancing †	✔	Always enabled
Health checks † †	✔	Improved
CloudWatch metrics	✔	Improved

Feature	Classic Load Balancer	Application Load Balancer
Access logs	✔	Improved
Path-based routing		✔
Route to multiple ports on a single instance		✔
HTTP/2 support		✔
Websockets support		✔
Load balancer deletion protection		✔

† Cross-zone load balancing is always enabled for an Application Load Balancer. For a Classic Load Balancer, it is disabled by default, but can be enabled and disabled as needed.

† † For an Application Load Balancer, you can specify the HTTP codes that indicate a successful health check response. An Application Load Balancer returns improved information about the cause of health check failures.

For more information about Application Load Balancers, see the Application Load Balancer Guide. For more information about Classic Load Balancers, see the Classic Load Balancer Guide.

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, and is supported on Windows, Mac, and Linux. For more information, see AWS Command Line Interface.
- **AWS SDKs** — Provides language-specific APIs and takes 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, but it 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 — API version 2015-12-01
  - Classic Load Balancers — API version 2012-06-01

Related Services

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

- **Amazon EC2** — Virtual servers that run your applications in the cloud. You can configure your load balancer to route traffic to your EC2 instances. For more information, see the Amazon EC2 User Guide for Linux Instances or the Amazon EC2 User Guide for Windows Instances.

- **Amazon 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 EC2 Container Service Developer Guide.
- **Auto Scaling** — Ensures that you are running your desired number of instances, even if an instance fails, and enables you to automatically increase or decrease the number of instances as the demand on your instances changes. If you enable Auto Scaling with Elastic Load Balancing, instances that are launched by Auto Scaling are automatically registered with the load balancer, and instances that are terminated by Auto Scaling are automatically de-registered from the load balancer. For more information, see the Auto Scaling User Guide.
- **Amazon CloudWatch** — Enables you to monitor your load balancer and take action as needed. For more information, see the Amazon CloudWatch User Guide.
- **Amazon Route 53** — Provides a reliable and cost-effective way to route visitors to websites by translating domain names (such as `www.example.com`) into the numeric IP addresses (such as `192.0.2.1`) that computers use to connect to each other. AWS assigns URLs to your resources, such as load balancers. However, you might want a URL that is easy for users to remember. For example, you can map your domain name to a load balancer. For more information, see the Amazon Route 53 Developer Guide.

Pricing

For more information, see the following pricing pages:

- Application Load Balancer Pricing
- Classic Load Balancer Pricing

How Elastic Load Balancing Works

A load balancer accepts incoming traffic from clients and routes requests to its registered EC2 instances in one or more Availability Zones. The load balancer also monitors the health of its registered instances and ensures that it routes traffic only to healthy instances. When the load balancer detects an unhealthy instance, it stops routing traffic to that instance, and then resumes routing traffic to that instance when it detects that the instance 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 and a protocol and port number for connections from the load balancer to the instances.

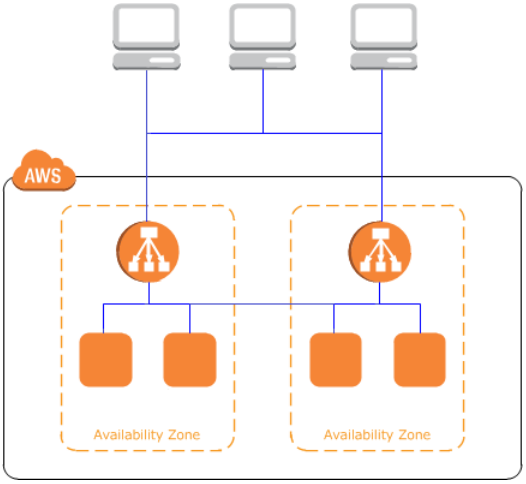
Elastic Load Balancing supports two types of load balancers: Application Load Balancers and Classic Load Balancers. There is a key difference between the way you configure these load balancers. With a Classic Load Balancer, you register instances with the load balancer. With an Application Load Balancer,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

Availability Zones and Instances

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 instances in an Availability Zone but do not enable the Availability Zone, these registered instances do not receive traffic.

With a Classic Load Balancer, we recommend that you enable multiple Availability Zones. With an Application Load Balancer, we require you to enable multiple Availability Zones. After you enable multiple Availability Zones, if one Availability Zone becomes unavailable or has no healthy instances, the load balancer can continue to route traffic to the healthy registered instances in another Availability Zone.

Cross-zone load balancing is always enabled for an Application Load Balancer and is disabled by default for a Classic Load Balancer. If cross-zone load balancing is enabled, the load balancer distributes traffic evenly across all registered instances in all enabled Availability Zones. If cross-zone load balancing is disabled, the load balancer distributes traffic evenly across all enabled Availability Zones. For example, suppose that you have 10 instances in Availability Zone us-west-2a and 2 instances in us-west-2b. If cross-zone load balancing is disabled, the requests are distributed evenly between us-west-2a and us-west-2b. As a result, the 2 instances in us-west-2b serve the same amount of traffic as the 10 instances in us-west-2a. However, if cross-zone load balancing is enabled, the load balancer distributes incoming requests evenly across all 12 instances.



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 instances are in the `amazonaws.com` domain. The Amazon DNS servers return one or more IP addresses to the client, which are the IP addresses of the load balancer nodes for your load balancer. As traffic to your application changes over time, Elastic Load Balancing scales your load balancer and updates the DNS entry. Note that the DNS entry also specifies the time-to-live (TTL) as 60 seconds, which ensures that the IP addresses can be remapped quickly in response to changing traffic.

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

Routing Algorithm

With a Classic Load Balancer, the load balancer node that receives the request selects a registered instance using the round robin routing algorithm for TCP listeners and the least outstanding requests routing algorithm for HTTP and HTTPS listeners.

With an Application Load Balancer, the load balancer node that receives the request selects a registered target from the target group using the round robin routing algorithm. Routing is performed independently for each target group, even when a target is registered with multiple target groups.

HTTP Connections

Classic Load Balancers use pre-open connections but Application Load Balancers do not.

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

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

Both Classic Load Balancers and Application Load Balancers use HTTP/1.1 on back-end connections (load balancer to registered instance). Keep-alive is supported on back-end 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 back-end connections. For Classic Load Balancer, the

host header contains the IP address of the load balancer node. For Application Load Balancer, the host header contains the DNS name of the load balancer.

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

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

HTTP Headers

Classic Load Balancers and Application Load Balancers support **X-Forwarded-For**, **X-Forwarded-Proto**, and **X-Forwarded-Port** headers.

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.

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

HTTP headers for Application Load Balancers have the following size limits:

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

Load Balancer Scheme

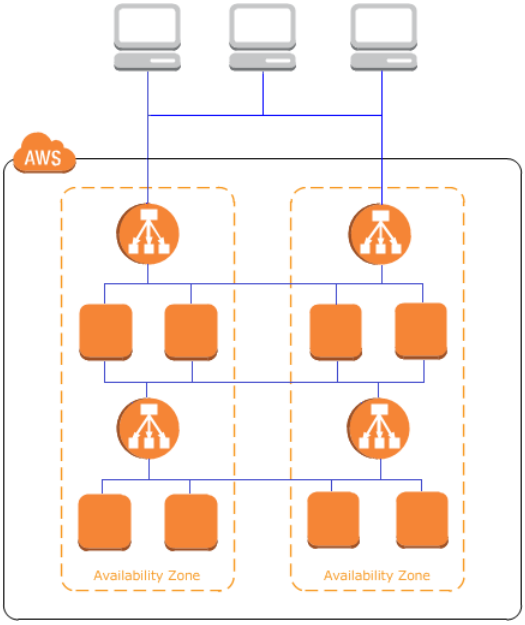
When you create a load balancer, you must choose whether to make it an internal load balancer or an Internet-facing load balancer. Note that when you create a Classic Load Balancer in EC2-Classic, it must be an Internet-facing load balancer.

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

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

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

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


AWS
Availability Zone
Availability Zone

Getting Started with Elastic Load Balancing

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

Tasks

Alternatively, to create a Classic Load Balancer, see Tutorial: Create a Classic Load Balancer in the *Classic Load Balancer Guide*.

Before You Begin

- Launch your EC2 instances in a virtual private cloud (VPC). Ensure that the security groups for these instances allow HTTP access on port 80.
- Install a web server, such as Apache or Internet Information Services (IIS), on each instance, enter its DNS name into the address field of an Internet-connected web browser, and verify that the browser displays the default page of the server.

Step 1: Select a Load Balancer Type

Elastic Load Balancing supports two types of load balancers: Application Load Balancers and Classic Load Balancers. For this tutorial, you create an Application Load Balancer.

To create an Application Load Balancer

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

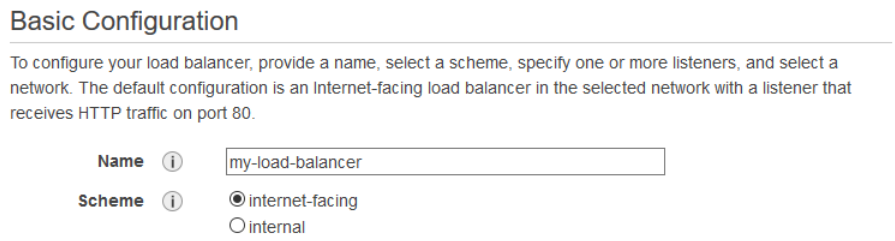
Step 2: Configure Your Load Balancer and Listener

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

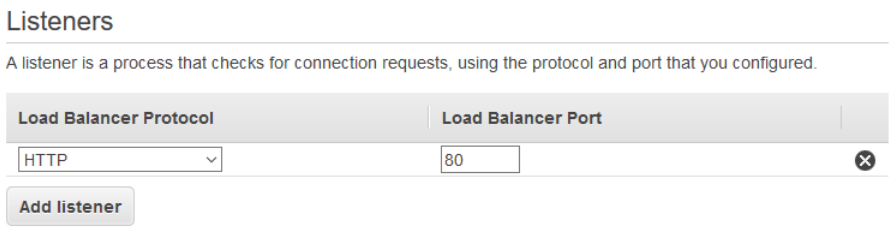
To configure your load balancer and listener

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

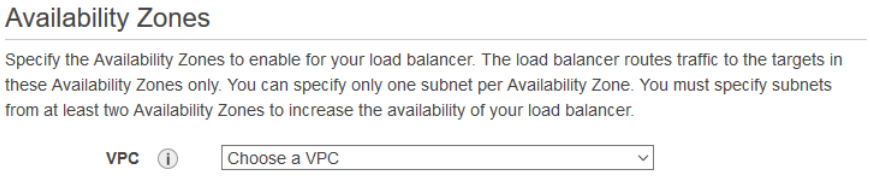
   The name of your Application Load Balancer must be unique within your set of Application Load Balancers for the region, can have a maximum of 32 characters, can contain only alphanumeric characters and hyphens, and must not begin or end with a hyphen.
2. For **Scheme**, keep the default value, **internet-facing**.



3. For **Listeners**, keep the default, which is a listener that accepts HTTP traffic on port 80.

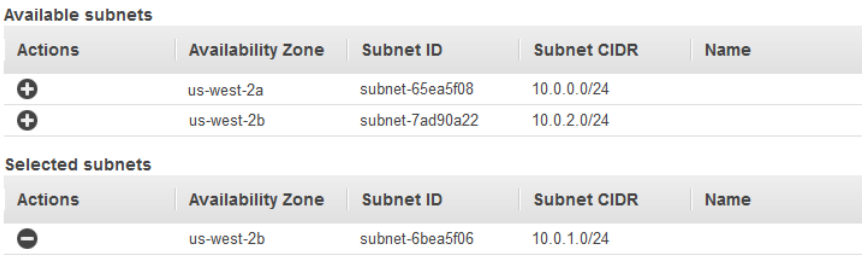


4. For **VPC**, select the same VPC that you used for your EC2 instances.



5. For **Available subnets**, select at least two public subnets using their add icons. The subnets are moved under **Selected subnets**. Note that you can select only one subnet per Availability Zone. If

you select a subnet from an Availability Zone where there is already a selected subnet, this subnet replaces the currently selected subnet for the Availability Zone.



6. Choose **Next: Configure Security Settings**.
7. For this tutorial, you are not using a secure listener. Choose **Next: Configure Security Groups**.

Step 3: Configure a Security Group for Your Load Balancer

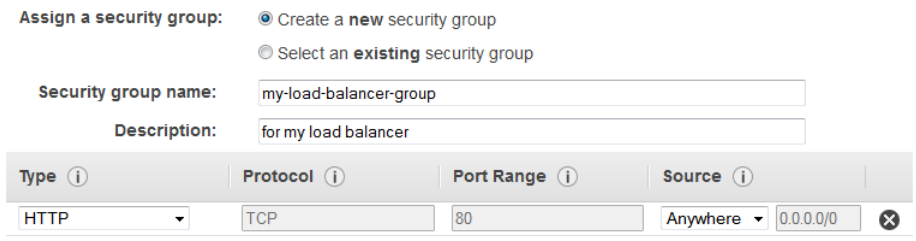
The security group for your load balancer must allow it to communicate with registered targets on both the listener port and the health check port. The console can create security groups for your load balancer on your behalf, with rules that specify the correct protocols and ports.

> **Note**
> If you prefer, you can create and select your own security group instead. For more information, see Recommended Rules in the *Application Load Balancer Guide*.

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

To configure a security group for your load balancer

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



3. Choose **Next: Configure Routing**.

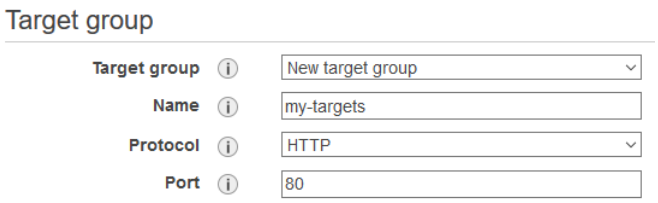
Step 4: Configure Your Target Group

Create a target group, which is used in request routing. The default rule for your listener routes requests to the to registered targets in this target group. The load balancer checks the health of targets

in this target group using the health check settings defined for the target group. On the **Configure Routing** page, complete the following procedure.

To configure your target group

1. For **Target group**, keep the default, **New target group**.
2. For **Name**, type a name for the new target group.
3. Keep **Protocol** as HTTP and **Port** as 80.



4. For **Health checks**, keep the default protocol and ping path.

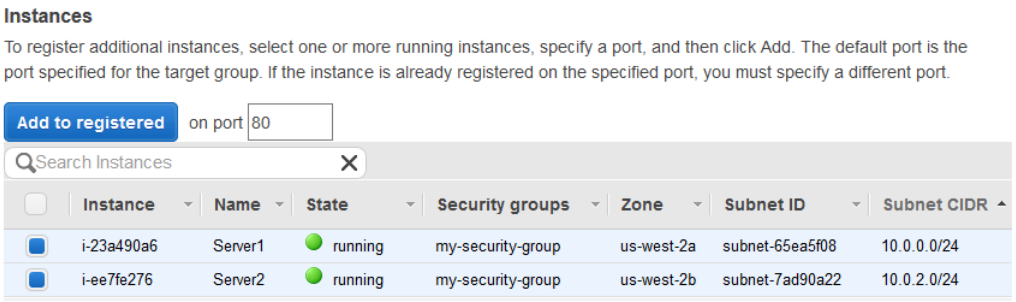
5. Choose **Next: Register Targets**.

Step 5: Register Targets with Your Target Group

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

To register targets with the target group

1. For **Instances**, select one or more instances.
2. Keep the default port, 80, and choose **Add to registered**.



3. If you need to remove an instance that you selected, for **Registered instances**, select the instance and then choose **Remove**.
4. When you have finished selecting instances, choose **Next: Review**.

Step 6: Create and Test Your Load Balancer

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

To create and test your load balancer

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

Step 7: Delete Your Load Balancer (Optional)

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

To delete your load balancer

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
3. Select the load balancer, and then choose **Actions**, **Delete**.
4. When prompted for confirmation, choose **Yes, Delete**.

Authentication and Access Control for Your Load Balancers

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

By default, IAM users don't have permission to create, view, or modify AWS resources. To allow an IAM user to access resources, such as a load balancer, and perform tasks, you must create an IAM policy that grants the IAM user permission to use the specific resources and API actions they'll need, then attach the policy to the IAM user or the group the IAM user belongs to. When you attach a policy to a user or group of users, it allows or denies the users permission to perform the specified tasks on the specified resources.

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

Grant Permissions Using IAM Policies

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

An IAM policy is a JSON document that consists of one or more statements. Each statement is structured as follows:

```
{
  "Version": "2012-10-17",
  "Statement":[{
    "Effect": "effect",
    "Action": "action",
    "Resource": "resource-arn",
```

```
    "Condition": {
      "condition": {
        "key":"value"
      }
    }
  }]
}
```

- **Effect**— The *effect* can be `Allow` or `Deny`. By default, IAM users don't have permission to use resources and API actions, so all requests are denied. An explicit allow overrides the default. An explicit deny overrides any allows.
- **Action**— The *action* is the specific API action for which you are granting or denying permission. For more information about specifying *action*, see Actions for Elastic Load Balancing (p. 14).
- **Resource**— The resource that's affected by the action. With many Elastic Load Balancing API actions, you can restrict the permissions granted or denied to a specific load balancer by specifying its Amazon Resource Name (ARN) in this statement. Otherwise, you can use the * wildcard to specify all of your load balancers. For more information, see Resource-Level Permissions for Elastic Load Balancing (p. 15).
- **Condition**— You can optionally use conditions to control when your policy is in effect. For more information, see Condition Keys for Elastic Load Balancing (p. 16).

For more information, see What is IAM?. For information about creating and managing users and groups, see IAM Users and Groups.

Actions for Elastic Load Balancing

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

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

To specify multiple actions in a single statement, enclose them in square brackets and separate them with a comma, as follows:

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

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

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

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

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

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

- Application Load Balancers — see the Elastic Load Balancing API Reference version 2015-12-01
- Classic Load Balancers — see the Elastic Load Balancing API Reference version 2012-06-01

Resource-Level Permissions for Elastic Load Balancing

Resource-level permissions refers to the ability to specify which the resources on which users are allowed to perform actions. Elastic Load Balancing has partial support for resource-level permissions.

Important
There are two API sets for Elastic Load Balancing: one for Application Load Balancers and one for Classic Load Balancers. Currently, the API for Application Load Balancers (API version 2015-12-01) does not support resource-level permissions. The API for Classic Load Balancers (API version 2012-06-01) supports resource-level permissions; however, not every API action supports resource-level permissions.

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

ARN Syntax

The ARN for a Classic Load Balancer has the following syntax:

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

region
The region for the load balancer. For more information about the regions supported by Elastic Load Balancing, see Regions and Endpoints in the *Amazon Web Services General Reference*.

account-id
Your AWS account ID, with no hyphens (for example, `0123456789012`).

load-balancer-name
The name of your load balancer.

ARN Example

The following is an example ARN for a Classic Load Balancer:

```
"Resource": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/my-load-balancer"
```

API Actions with No Support for Resource-Level Permissions

You can't specify the ARN for a specific load balancer using the following API actions:

- All API actions for Application Load Balancers (API version 2015-12-01)
- The following API actions for Classic Load Balancers (API version 2012-06-01):
  - `DescribeInstanceHealth`
  - `DescribeLoadBalancerAttributes`
  - `DescribeLoadBalancerPolicyTypes`
  - `DescribeLoadBalancers`
  - `DescribeLoadBalancerPolicies`
  - `DescribeTags`

For API actions that don't support resource-level permissions, you must specify the following resource statement:

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

Condition Keys for Elastic Load Balancing

In an IAM policy statement, you have the option to specify conditions that control when it is in effect. Each condition contains one or more key-value pairs. AWS has defined the following keys that you can use to specify the conditions under which your IAM users and groups can use the load balancers.

Tip
Key names are case-sensitive.

- `aws:CurrentTime`— Use with date/time conditions to restrict access based on request time (see Date Conditions).
- `aws:EpochTime`— Use with date/time conditions to specify a date in epoch or UNIX time (see Date Conditions).
- `aws:MultiFactorAuthPresent`— Use to check whether the IAM user making the API request was authenticated using a multi-factor authentication (MFA) device.
- `aws:MultiFactorAuthAge`— Use to check how long ago (in seconds) the MFA-validated security credentials making the request were issued. Unlike other keys, this key is not present if MFA is not used (see Existence of Condition Keys, Numeric Conditions, and Using Multi-Factor Authentication (MFA) Devices with AWS).
- `aws:SecureTransport`— Use to check whether the request was sent using SSL (see Boolean Conditions).
- `aws:SourceIp`— Use to check the requester's IP address (see IP Address). Note that if you use `aws:SourceIp`, and the request comes from an EC2 instance, the public IP address of the instance is evaluated.
- `aws:Referer`— Use to check the user making the HTTP request.
- `aws:UserAgent`— Use with string conditions to check the client application that made the request (see String Conditions).
- `aws:userid`— Use to check the user ID of the requester (see String Conditions).
- `aws:username`—Use to check the user name of the requester, if available (see String Conditions).

After you have decided how to control access to your load balancers, open the IAM console and follow the directions in Managing IAM Policies to create an IAM policy for Elastic Load Balancing. For more information, see Testing IAM Policies.

Migrate from a Classic Load Balancer to an Application Load Balancer

If you have an existing Classic Load Balancer and you have determined that an Application Load Balancer would meet your needs, you can migrate from a Classic Load Balancer to an Application Load Balancer. After you have completed the migration process, you can take advantage of the features of your Application Load Balancer.

Migration Process

Step 1: Create an Application Load Balancer

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

- Create a load balancer, with the same scheme (Internet-facing or internal), subnets, and security groups as the Classic Load Balancer.
- Create one target group for your load balancer, with the same health check settings that you have for your Classic Load Balancer.
- Do one of the following:
  - If your Classic Load Balancer is attached to an Auto Scaling group, attach your target group to the Auto Scaling group. This also registers the Auto Scaling instances with the target group.
  - Register your EC2 instances with your target group.
- Create one or more listeners, each with a default rule that forwards requests to the target group. If you create an HTTPS listener, you can specify the same certificate that you specified for your Classic Load Balancer, but you must use the default security policy, **ELBSecurityPolicy-2015-05**.

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

You can create the Application Load Balancer and target group step-by-step using the AWS Management Console, AWS CLI, or an AWS SDK. Alternatively, you can create them using the Classic Load Balancer to Application Load Balancer copy utility, which you can find on GitHub.

Resources

- Load Balancer Copy Utility (GitHub)
- Getting Started with Elastic Load Balancing (p. 8) (*Elastic Load Balancing User Guide*)
- Tutorial: Create an Application Load Balancer Using the AWS CLI (*Application Load Balancer Guide*)

Step 2: Gradually Redirect Traffic to Your Application Load Balancer

After your instances are registered with your Application Load Balancer, you can begin the process of testing your Application Load Balancer as you gradually redirect traffic.

To redirect traffic to your Application Load Balancer gradually

1. Paste the DNS name of your Application Load Balancer into the address field of an Internet-connected web browser. If everything is working, the browser displays the default page of your server.
2. Create a new DNS record that associates your domain name with your Application Load Balancer. If your DNS service supports weighting, specify a weight of 1 in the new DNS record and a weight of 9 in the existing DNS record for your Classic Load Balancer. This directs 10% of the traffic to the Application Load Balancer and 90% of the traffic to the Classic Load Balancer.
3. Monitor your Application 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, which means that any DNS server that resolves your domain name keeps the record information in its cache for 60 seconds. Therefore, these DNS servers can still route traffic to your Classic Load Balancer for up to 60 seconds after you complete the previous step and the changes start to propagate to DNS servers around the world. 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 Application Load Balancer. When you are finished, you can delete the DNS record for your Classic Load Balancer.

Step 3: Update References to Your Classic Load Balancer

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

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

- IAM policies that grant users access to your Classic Load Balancer (for Application Load Balancers, action names are different and resource-level permissions are not supported)
- Processes that use the CloudWatch metrics for Classic Load Balancers (instead of Application Load Balancers)
- AWS CloudFormation templates
- AWS OpsWorks recipes

Resources

- Authentication and Access Control for Your Load Balancers (p. 13)
- elbv2 in the *AWS Command Line Interface Reference*
- The Elastic Load Balancing API Reference version 2015-12-01
- Application Load Balancer Metrics in the *Application Load Balancer Guide*

Step 4: Delete the Classic Load Balancer

After you've redirected all traffic to the Application Load Balancer and all existing requests that were routed to the Classic Load Balancer have completed, you can delete the Classic Load Balancer.