aws

Classic Load Balancers

# Elastic Load Balancing

# Elastic Load Balancing: Classic Load Balancers

# Table of Contents

# What is a Classic Load Balancer?

> **Note**
>
> Classic Load Balancers are the previous generation of load balancers from Elastic Load Balancing. We recommend that you migrate to a current generation load balancer. For more information, see Migrate your Classic Load Balancer.

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 as your incoming traffic changes over time. It can automatically scale to the vast majority of workloads.

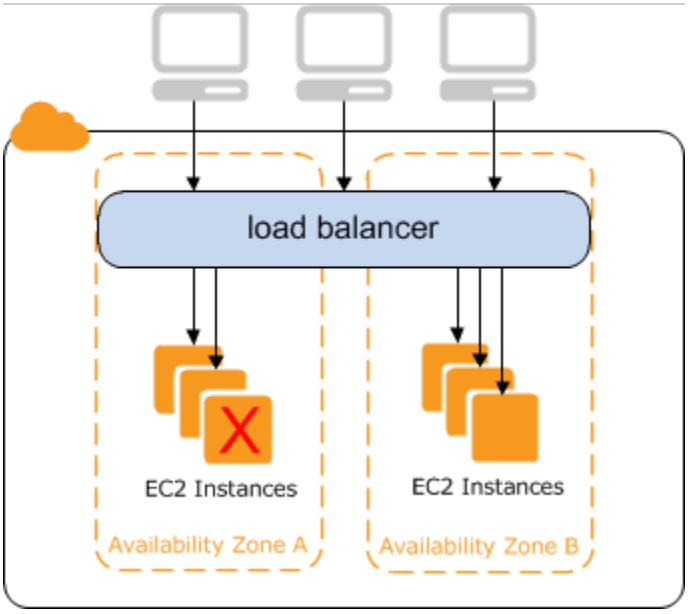
## Classic Load Balancer overview

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

Your load balancer serves as a single point of contact for clients. This 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. Elastic Load Balancing can scale to the vast majority of workloads automatically.

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

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




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

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

For more information, see How Elastic Load Balancing works in the *Elastic Load Balancing User Guide*.

## Benefits

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

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

For more information about the features supported by each load balancer type, see Product comparisons for Elastic Load Balancing.

## How to get started

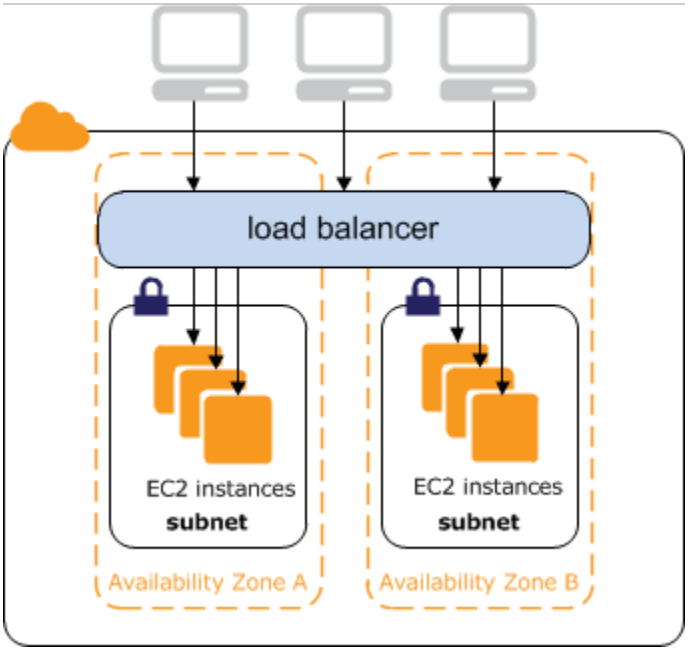
- To learn how to create a Classic Load Balancer and register EC2 instances with it, see Create an internet-facing Classic Load Balancer.
- To learn how to create an HTTPS load balancer and register EC2 instances with it, see Create a Classic Load Balancer with an HTTPS listener.
- To learn how to use the various features supported by Classic Load Balancers, see Configure your Classic Load Balancer.

## Pricing

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

# Internet-facing Classic Load Balancers

When you create a Classic Load Balancer, you can make it an internal load balancer or an internet-facing load balancer. An internet-facing load balancer has a publicly resolvable DNS name, so it can route requests from clients over the internet to the EC2 instances that are registered with the load balancer.




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. For more information, see *Internal load balancers*.

**Contents**



# Public DNS names for your load balancer

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

The console displays a public DNS name with the following form:

```
name-1234567890.region.elb.amazonaws.com
```

# Create an internet-facing Classic Load Balancer

When you create a load balancer, you configure listeners, configure health checks, and register back-end instances. You configure a listener by specifying a protocol and a port for front-end (client to load balancer) connections, and a protocol and a port for back-end (load balancer to back-end instances) connections. You can configure multiple listeners for your load balancer.

This tutorial provides a hands-on introduction to Classic Load Balancers through the AWS Management Console, a web-based interface. You'll create a load balancer that receives public HTTP traffic and sends it to your EC2 instances.

To create a load balancer with an HTTPS listener, see Create a Classic Load Balancer with an HTTPS listener.

**Tasks**

- Before you begin
- Create a Classic Load Balancer using the AWS Management Console

## Before you begin

- Create a virtual private cloud (VPC). For more information, see Recommendations for your VPC.
- Launch the EC2 instances that you plan to register with your load balancer. 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.

## Create a Classic Load Balancer using the AWS Management Console

Use the following procedure to create your Classic Load Balancer. Provide basic configuration information for your load balancer, such as a name and scheme. Then provide information about your network, and the listener that routes traffic to your instances.

**To create a Classic Load Balancer using the console**

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. Expand the **Classic Load Balancer** section, then choose **Create**.
6. **Basic configuration**

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

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

   b. For **Scheme**, select **Internet-facing**.
7. **Network mapping**

   a. For **VPC**, select the same VPC that you selected for your instances.

   b. For **Mappings**, first select an Availability Zone, then choose a public subnet from its available subnets. You can only select one subnet per Availability Zone. To improve the availability of your load balancer, select more than one Availability Zone and subnet.
8. **Security groups**

   - For **Security groups**, select an existing security group that is configured to allow the required HTTP traffic on port 80.
9. **Listeners and routing**

   a. For **Listener**, ensure the protocol is HTTP and the port is 80.

   b. For **Instance**, ensure the protocol is HTTP and the port is 80.
10. **Health checks**

    a. For **Ping Protocol**, ensure the protocol is HTTP.

    b. For **Ping Port**, ensure the port is 80.

    c. For **Ping Path**, ensure the path is /.

    d. For **Advanced health check settings**, use the default values.

11. **Instances**

    a. Select **Add instances**, to bring up the instance selection screen.

    b. Under **Available instances**, you can select from the current instances that are available to the load balancer, based on the current network settings.

    c. After you're satisfied with your selections, select **Confirm** to add the instances to be registered to the load balancer.

12. **Attributes**

    - For **Enable cross-zone load balancing**, **Enable connection draining**, and **Timeout (draining interval)** keep the default values.

13. **Load balancer tags (optional)**

    a. The **Key** field is required.

    b. The **Value** field is optional.

    c. To add another tag, select **Add new tag** then input your values into the **Key** field, and optionally the **Value** field.

    d. To remove an existing tag, select **Remove** next to the tag you want to remove.

14. **Summary and creation**

    a. If you need to change any settings, select **Edit** next to the setting needing to be changed.

    b. After you're satisfied with all the settings shown in the summary, select **Create load balancer** to begin creation of your load balancer.

    c. On the final creation page, select **View load balancer** to view your load balancer in the Amazon EC2 console.

15. **Verify**

    a. Select your new load balancer.

    b. On the **Target instances** tab, check the **Health status** column. After at least one of your EC2 instances is **In-service**, you can test your load balancer.

    c. In the **Details** section, copy the load balancers **DNS name**, which would look similar to `my-load-balancer-1234567890.us-east-1.elb.amazonaws.com`.

    d. Paste your load balancers **DNS name** into the address field of a public internet connected web browser. If your load balancer is functioning correctly, you will see the default page of your server.

16. **Delete (optional)**

    a. If you have a CNAME record for your domain that points to your load balancer, point it to a new location and wait for the DNS change to take effect before deleting your load balancer.

    b. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

    c. Select the load balancer.

    d. Choose **Actions**, **Delete load balancer**.

    e. When prompted for confirmation, type `confirm` then select **Delete**.

    f. After you delete a load balancer, the EC2 instances that were registered with the load balancer continue to run. You will be billed for each partial or full hour that they continue running. When you no longer need an EC2 instance, you can stop or terminate it to prevent incurring additional charges.

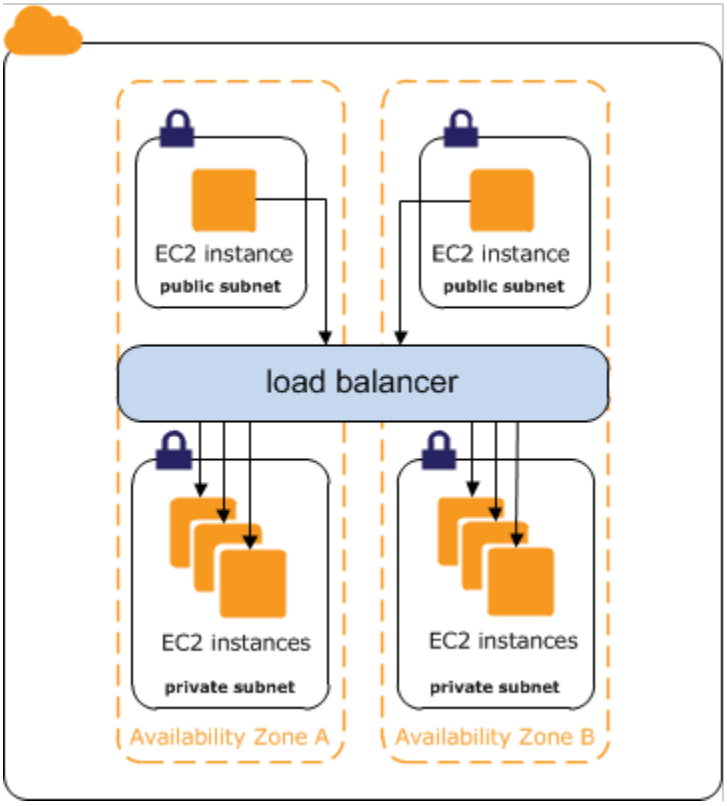
# Internal Classic Load Balancers

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. For more information, see Internet-facing Classic Load Balancers.

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.

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.




**Contents**



# Public DNS name for your load balancer

When an internal load balancer is created, it receives a public DNS name with the following form:

```
internal-name-123456789.region.elb.amazonaws.com
```

The DNS servers resolve the DNS name of your load balancer to the private IP addresses of the load balancer nodes for your internal load balancer. Each load balancer node is connected to the private IP addresses of the back-end instances using elastic network interfaces. If cross-zone load balancing is enabled, each node is connected to each back-end instance, regardless of Availability Zone. Otherwise, each node is connected only to the instances that are in its Availability Zone.

# Create an internal Classic Load Balancer

You can create an internal load balancer to distribute traffic to your EC2 instances from clients with access to the VPC for the load balancer.

**Contents**



## Prerequisites

- If you have not yet created a VPC for your load balancer, you must create it before you get started. For more information, see Recommendations for your VPC.
- Launch the EC2 instances that you plan to register with your internal load balancer. Ensure that you launch them in private subnets in the VPC intended for the load balancer.

## Create an internal load balancer using the console

Use the following procedure to create your internal Classic Load Balancer. Provide basic configuration information for your load balancer, such as a name and scheme. Then provide information about your network, and the listener that routes traffic to your instances..

**To create an internal Classic Load Balancer using the console**

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. Expand the **Classic Load Balancer** section, then choose **Create**.
6. **Basic configuration**

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

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

b. For **Scheme**, select **Internal**.

7. **Network mapping**

   a. For **VPC**, select the same VPC that you selected for your instances.

   b. For **Mappings**, first select an Availability Zone, then choose a subnet from its available subnets. You can only select one subnet per Availability Zone. To improve the availability of your load balancer, select more than one Availability Zone and subnet.

8. For **Security groups**, select an existing security group that is configured to allow the required HTTP traffic on port 80. Or you can create a new security group if your application uses different protocols and ports.

9. **Listeners and routing**

   a. For **Listener**, ensure the protocol is HTTP and the port is 80.

   b. For **Instance**, ensure the protocol is HTTP and the port is 80.

10. **Health checks**

    a. For **Ping Protocol**, the default is HTTP.

    b. For **Ping Port**, the default is 80.

    c. For **Ping Path**, the default is /.

    d. For **Advanced health check settings**, use the default values or enter values specific to your application.

11. **Instances**

    a. Select **Add instances**, to bring up the instance selection screen.

    b. Under **Available instances**, you can select from the current instances that are available to the load balancer, based on the network settings selected before.

    c. After you're satisfied with your selections, select **Confirm** to add the instances to be registered to the load balancer.

12. **Attributes**

    - For **Enable cross-zone load balancing**, **Enable connection draining**, and **Timeout (draining interval)** keep the default values.

13. **Load balancer tags (optional)**

    a. The **Key** field is required.

    b. The **Value** field is optional.

    c. To add another tag, select **Add new tag** then input your values into the **Key** field, and optionally the **Value** field.

    d. To remove an existing tag, select **Remove** next to the tag you want to remove.

14. **Summary and creation**

    a. If you need to change any settings, select **Edit** next to the setting needing to be changed.

    b. After you're satisfied with all the settings shown in the summary, select **Create load balancer** to begin creation of your load balancer.

    c. On the final creation page, select **View load balancer** to view your load balancer in the Amazon EC2 console.

15. **Verify**

    a. Select your new load balancer.

    b. On the **Target instances** tab, check the **Health status** column. After at least one of your EC2 instances is **In-service**, you can test your load balancer.

    c. In the **Details** section, copy the load balancers **DNS name**, which would look similar to `my-load-balancer-1234567890.us-east-1.elb.amazonaws.com`.

    d. Paste your load balancers **DNS name** into the address field of a public internet connected web browser. If your load balancer is functioning correctly, you will see the default page of your server.

16. **Delete (optional)**

    a. If you have a CNAME record for your domain that points to your load balancer, point it to a new location and wait for the DNS change to take effect before deleting your load balancer.

    b. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

    c. Select the load balancer.

    d. Choose **Actions**, **Delete load balancer**.

    e. When prompted for confirmation, type `confirm` then select **Delete**.

    f. After you delete a load balancer, the EC2 instances that were registered with the load balancer continue to run. You will be billed for each partial or full hour that they continue

running. When you no longer need an EC2 instance, you can stop or terminate it to prevent incurring additional charges.

## Create an internal load balancer using the AWS CLI

By default, Elastic Load Balancing creates an internet-facing load balancer. Use the following procedure to create an internal load balancer and register your EC2 instances with the newly created internal load balancer.

**To create an internal load balancer**

1. Use the create-load-balancer command with the `--scheme` option set to `internal`, as follows:

```
aws elb create-load-balancer --load-balancer-name my-internal-loadbalancer --
listeners Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,InstancePort=80
 --subnets subnet-4e05f721 --scheme internal --security-groups sg-b9ffedd5
```

   The following is an example response. Note that the name indicates that this is an internal load balancer.

```
{
    "DNSName": "internal-my-internal-loadbalancer-786501203.us-
west-2.elb.amazonaws.com"
}
```

2. Use the following register-instances-with-load-balancer command to add instances:

```
aws elb register-instances-with-load-balancer --load-balancer-name my-internal-
loadbalancer --instances i-4f8cf126 i-0bb7ca62
```

   The following is an example response:

```
{
    "Instances": [
        {
            "InstanceId": "i-4f8cf126"
        },
        {
            "InstanceId": "i-0bb7ca62"
```

```
        }
    ]
}
```

3. (Optional) Use the following describe-load-balancers command to verify the internal load balancer:

```
aws elb describe-load-balancers --load-balancer-name my-internal-loadbalancer
```

The response includes the DNSName and Scheme fields, which indicate that this is an internal load balancer.

```
{
    "LoadBalancerDescriptions": [
        {
            ...
            "DNSName": "internal-my-internal-loadbalancer-1234567890.us-
west-2.elb.amazonaws.com",
            "SecurityGroups": [
                "sg-b9ffedd5"
            ],
            "Policies": {
                "LBCookieStickinessPolicies": [],
                "AppCookieStickinessPolicies": [],
                "OtherPolicies": []
            },
            "LoadBalancerName": "my-internal-loadbalancer",
            "CreatedTime": "2014-05-22T20:32:19.920Z",
            "AvailabilityZones": [
                "us-west-2a"
            ],
            "Scheme": "internal",
            ...
        }
    ]
}
```

# Configure your Classic Load Balancer

After you create a Classic Load Balancer, you can change its configuration. For example, you can update the load balancer attributes, subnets, and security groups.

**Load balancer attributes**

Connection draining

If enabled, the load balancer allows existing requests to complete before the load balancer shifts traffic away from a deregistered or unhealthy instance.

Cross-zone load balancing

If enabled, the load balancer routes the request traffic evenly across all instances regardless of the Availability Zones.

Desync migitation mode

Determines how the load balancer handles requests that might pose a security risk to your application. The possible values are `monitor`, `defensive`, and `strictest`. The default is `defensive`.

Idle timeout

If enabled, the load balancer allows the connections to remain idle (no data is sent over the connection) for the specified duration. The default is 60 seconds.

Sticky sessions

Classic Load Balancers support both duration-based and application-based session stickiness.

**Load balancer details**

Security groups

The security groups for your load balancer must allow traffic on the listener and health check ports.

Subnets

You can expand the ability of your load balancer to additional subnets.

Proxy protocol

If enabled, we add a header with connection information that is sent to the instance.

Tags

You can add tags to categorize your load balancres.

# Configure the idle connection timeout for your Classic Load Balancer

For each request that a client makes through a Classic Load Balancer, the load balancer maintains two connections. The front-end connection is between the client and the load balancer. The back-end connection is between the load balancer and a registered EC2 instance. The load balancer has a configured idle timeout period that applies to its connections. If no data has been sent or received by the time that the idle timeout period elapses, the load balancer closes the connection. To ensure that lengthy operations such as file uploads have time to complete, send at least 1 byte of data before each idle timeout period elapses, and increase the length of the idle timeout period as needed.

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

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

**Contents**



## Configure the idle timeout using the console

By default, Elastic Load Balancing sets the idle timeout for your load balancer to 60 seconds. Use the following procedure to set a different value for the idle timeout.

**To configure the idle timeout setting for your load balancer using the console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Load Balancing**, choose **Load Balancers**.
3. Choose the name of the load balancer to open its detail page.
4. On the **Attributes** tab, choose **Edit**.
5. On the **Edit load balancer attributes** page, in the **Traffic configuration** section, type a value for **Idle timeout**. The range for the idle timeout is from 1 to 4,000 seconds.
6. Choose **Save changes**.

## Configure the idle timeout using the AWS CLI

Use the following modify-load-balancer-attributes command to set the idle timeout for your load balancer:

```
aws elb modify-load-balancer-attributes --load-balancer-name my-loadbalancer --load-balancer-attributes "{\"ConnectionSettings\":{\"IdleTimeout\":30}}"
```

The following is an example response:

```
{
    "LoadBalancerAttributes": {
        "ConnectionSettings": {
            "IdleTimeout": 30
        }
    },
    "LoadBalancerName": "my-loadbalancer"
}
```

# Configure cross-zone load balancing for your Classic Load Balancer

With *cross-zone load balancing*, each load balancer node for your Classic Load Balancer distributes requests evenly across the registered instances in all enabled Availability Zones. If cross-zone load balancing is disabled, each load balancer node distributes requests evenly across the registered instances in its Availability Zone only. For more information, see Cross-zone load balancing in the *Elastic Load Balancing User Guide*.

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

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

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

**Contents**



## Enable cross-zone load balancing

You can enable cross-zone load balancing for your Classic Load Balancer at any time.

**To enable cross-zone load balancing using the console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Load Balancing**, choose **Load Balancers**.
3. Choose the name of the load balancer to open its detail page.
4. On the **Attributes** tab, choose **Edit**.
5. On the **Edit load balancer attributes** page, in the **Availability Zone routing configuration** section, enable **Cross-zone load balancing**.
6. Choose **Save changes**.

**To enable cross-zone load balancing using the AWS CLI**

1. Use the following modify-load-balancer-attributes command to set the `CrossZoneLoadBalancing` attribute of your load balancer to `true`:

```
aws elb modify-load-balancer-attributes --load-balancer-name my-loadbalancer --
load-balancer-attributes "{\"CrossZoneLoadBalancing\":{\"Enabled\":true}}"
```

The following is an example response:

```
{
   "LoadBalancerAttributes": {
     "CrossZoneLoadBalancing": {
         "Enabled": true
       }
   },
   "LoadBalancerName": "my-loadbalancer"
 }
```

2. (Optional) Use the following describe-load-balancer-attributes command to verify that cross-zone load balancing is enabled for your load balancer:

```
aws elb describe-load-balancer-attributes --load-balancer-name my-loadbalancer
```

The following is an example response:

```
{
    "LoadBalancerAttributes": {
        "ConnectionDraining": {
            "Enabled": false,
            "Timeout": 300
        },
        "CrossZoneLoadBalancing": {
            "Enabled": true
        },
        "ConnectionSettings": {
            "IdleTimeout": 60
        },
        "AccessLog": {
            "Enabled": false
        }
    }
}
```

# Disable cross-zone load balancing

You can disable the cross-zone load balancing option for your load balancer at any time.

**To disable cross-zone load balancing using the console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Load Balancing**, choose **Load Balancers**.
3. Choose the name of the load balancer to open its detail page.
4. On the **Attributes** tab, choose **Edit**.
5. On the **Edit load balancer attributes** page, in the **Availability Zone routing configuration** section, disable **Cross-zone load balancing**.
6. Choose **Save changes**.

To disable cross-zone load balancing, set the `CrossZoneLoadBalancing` attribute of your load balancer to `false`.

**To disable cross-zone load balancing using the AWS CLI**

1. Use the following modify-load-balancer-attributes command:

```
aws elb modify-load-balancer-attributes --load-balancer-name my-loadbalancer --
load-balancer-attributes "{\"CrossZoneLoadBalancing\":{\"Enabled\":false}}"
```

   The following is an example response:

```
{
   "LoadBalancerAttributes": {
     "CrossZoneLoadBalancing": {
         "Enabled": false
       }
   },
   "LoadBalancerName": "my-loadbalancer"
 }
```

2. (Optional) Use the following describe-load-balancer-attributes command to verify that cross-zone load balancing is disabled for your load balancer:

```
aws elb describe-load-balancer-attributes --load-balancer-name my-loadbalancer
```

The following is an example response:

```
{
    "LoadBalancerAttributes": {
        "ConnectionDraining": {
            "Enabled": false,
            "Timeout": 300
        },
        "CrossZoneLoadBalancing": {
            "Enabled": false
        },
        "ConnectionSettings": {
            "IdleTimeout": 60
        },
        "AccessLog": {
            "Enabled": false
        }
    }
}
```

# Configure connection draining for your Classic Load Balancer

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

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

If a de-registering instance has no in-flight requests and no active connections, Elastic Load Balancing immediately completes the deregistration process.

While in-flight requests are being served, the load balancer reports the state of a de-registering instance as `InService: Instance deregistration currently in progress`. When the

de-registering instance is finished serving all in-flight requests, or when the maximum timeout limit is reached, the load balancer reports the instance state as `OutOfService: Instance is not currently registered with the LoadBalancer`.

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

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

You can disable connection draining if you want your load balancer to immediately close connections to the instances that are de-registering or have become unhealthy. When connection draining is disabled, any in-flight requests made to instances that are de-registering or unhealthy are not completed.

**Contents**



## Enable connection draining

You can enable connection draining for your load balancer at any time.

**To enable connection draining using the console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Load Balancing**, choose **Load Balancers**.
3. Choose the name of the load balancer to open its detail page.
4. On the **Attributes** tab, choose **Edit**.
5. On the **Edit load balancer attributes** page, in the **Traffic configuration** section, select **Enable connection draining**.
6. (Optional) For **Timeout (draining interval)**, enter a value between 1 and 3,600 seconds. Otherwise the default of 300 seconds is used.
7. Choose **Save changes**.

### To enable connection draining using the AWS CLI

Use the following modify-load-balancer-attributes command:

```
aws elb modify-load-balancer-attributes --load-balancer-name my-loadbalancer --load-balancer-attributes "{\"ConnectionDraining\":{\"Enabled\":true,\"Timeout\":300}}"
```

The following is an example response:

```
{
    "LoadBalancerAttributes": {
        "ConnectionDraining": {
            "Enabled": true,
            "Timeout": 300
        }
    },
    "LoadBalancerName": "my-loadbalancer"
}
```

## Disable connection draining

You can disable connection draining for your load balancer at any time.

### To disable connection draining using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Load Balancing**, choose **Load Balancers**.
3. Choose the name of the load balancer to open its detail page.
4. On the **Attributes** tab, choose **Edit**.
5. On the **Edit load balancer attributes** page, in the **Traffic configuration** section, deselect **Enable connection draining**.
6. Choose **Save changes**.

### To disable connection draining using the AWS CLI

Use the following modify-load-balancer-attributes command:

```
aws elb modify-load-balancer-attributes --load-balancer-name my-loadbalancer --load-balancer-attributes "{\"ConnectionDraining\":{\"Enabled\":false}}"
```

The following is an example response:

```
{
    "LoadBalancerAttributes": {
        "ConnectionDraining": {
            "Enabled": false,
            "Timeout": 300
        }
    },
    "LoadBalancerName": "my-loadbalancer"
}
```

# Configure sticky sessions for your Classic Load Balancer

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

The key to managing sticky sessions is to determine how long your load balancer should consistently route the user's request to the same instance. If your application has its own session cookie, then you can configure Elastic Load Balancing so that the session cookie follows the duration specified by the application's session cookie. If your application does not have its own session cookie, then you can configure Elastic Load Balancing to create a session cookie by specifying your own stickiness duration.

Elastic Load Balancing creates a cookie, named AWSELB, that is used to map the session to the instance.

**Requirements**

- An HTTP/HTTPS load balancer.
- At least one healthy instance in each Availability Zone.

**Compatibility**

- The RFC for the path property of a cookie allows underscores. However, Elastic Load Balancing URI encodes underscore characters as %5F because some browsers, such as Internet Explorer 7, expect underscores to be URI encoded as %5F. Because of the potential to impact browsers that

are currently working, Elastic Load Balancing continues to URI encode underscore characters. For example, if the cookie has the property `path=/my_path`, Elastic Load Balancing changes this property in the forwarded request to `path=/my%5Fpath`.

- You can't set the `secure` flag or `HttpOnly` flag on your duration-based session stickiness cookies. However, these cookies contain no sensitive data. Note that if you set the `secure` flag or `HttpOnly` flag on an application-controlled session stickiness cookie, it is also set on the AWSELB cookie.
- If you have a trailing semicolon in the `Set-Cookie` field of an application cookie, the load balancer ignores the cookie.

**Contents**



## Duration-based session stickiness

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

With CORS (cross-origin resource sharing) requests, some browsers require `SameSite=None; Secure` to enable stickiness. In this case, Elastic Load Balancing creates a second stickiness cookie, AWSELBCORS, which includes the same information as the original stickiness cookie plus this `SameSite` attribute. Clients receive both cookies.

If an instance fails or becomes unhealthy, the load balancer stops routing requests to that instance, and chooses a new healthy instance based on the existing load balancing algorithm. The request is routed to the new instance as if there is no cookie and the session is no longer sticky.

If a client switches to a listener with a different backend port, stickiness is lost.

**To enable duration-based sticky sessions for a load balancer using the console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Load Balancing**, choose **Load Balancers**.
3. Choose the name of the load balancer to open its detail page.
4. On the **Listeners** tab, choose **Manage listeners**.
5. On the **Manage listeners** page, locate the listener to be updated and choose **Edit** under **Cookie stickiness**.
6. On the **Edit cookie stickiness setting** pop-up, select **Generated by load balancer**.
7. (Optional) For **Expiration period**, type the cookie expiration period, in seconds. If you do not specify an expiration period, the sticky session lasts for the duration of the browser session.
8. Choose **Save changes** to close the pop-up window.
9. Choose **Save changes** to return to the load balancer details page.

**To enable duration-based sticky sessions for a load balancer using the AWS CLI**

1. Use the following create-lb-cookie-stickiness-policy command to create a load balancer-generated cookie stickiness policy with a cookie expiration period of 60 seconds:

```
aws elb create-lb-cookie-stickiness-policy --load-balancer-name my-loadbalancer --
policy-name my-duration-cookie-policy --cookie-expiration-period 60
```

2. Use the following set-load-balancer-policies-of-listener command to enable session stickiness for the specified load balancer:

```
aws elb set-load-balancer-policies-of-listener --load-balancer-name my-loadbalancer
 --load-balancer-port 443 --policy-names my-duration-cookie-policy
```

> **Note**
>
> The `set-load-balancer-policies-of-listener` command replaces the current set of policies associated with the specified load balancer port. Every time you use this command, specify the `--policy-names` option to list all policies to enable.

3. (Optional) Use the following describe-load-balancers command to verify that the policy is enabled:

```
aws elb describe-load-balancers --load-balancer-name my-loadbalancer
```

The response includes the following information, which shows that the policy is enabled for the listener on the specified port:

```
{
    "LoadBalancerDescriptions": [
        {
            ...
            "ListenerDescriptions": [
                {
                    "Listener": {
                        "InstancePort": 443,
                        "SSLCertificateId": "arn:aws:iam::123456789012:server-
certificate/my-server-certificate",
                        "LoadBalancerPort": 443,
                        "Protocol": "HTTPS",
                        "InstanceProtocol": "HTTPS"
                    },
                    "PolicyNames": [
                        "my-duration-cookie-policy",
                        "ELBSecurityPolicy-TLS-1-2-2017-01"
                    ]
                },
                ...
            ],
            ...
            "Policies": {
                "LBCookieStickinessPolicies": [
                 {
                        "PolicyName": "my-duration-cookie-policy",
                        "CookieExpirationPeriod": 60
                    }

                ],
                "AppCookieStickinessPolicies": [],
                "OtherPolicies": [
                    "ELBSecurityPolicy-TLS-1-2-2017-01"
                ]
            },
            ...
        }
```

```
    ]
}
```

## Application-controlled session stickiness

The load balancer uses a special cookie to associate the session with the instance that handled the initial request, but follows the lifetime of the application cookie specified in the policy configuration. The load balancer only inserts a new stickiness cookie if the application response includes a new application cookie. The load balancer stickiness cookie does not update with each request. If the application cookie is explicitly removed or expires, the session stops being sticky until a new application cookie is issued.

The following attributes set by back-end instances are sent to clients in the cookie: `path`, `port`, `domain`, `secure`, `httponly`, `discard`, `max-age`, `expires`, `version`, `comment`, `commenturl`, and `samesite`.

If an instance fails or becomes unhealthy, the load balancer stops routing requests to that instance, and chooses a new healthy instance based on the existing load balancing algorithm. The load balancer treats the session as now "stuck" to the new healthy instance, and continues routing requests to that instance even if the failed instance comes back.

**To enable application-controlled session stickiness using the console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Load Balancing**, choose **Load Balancers**.
3. Choose the name of the load balancer to open its detail page.
4. On the **Listeners** tab, choose **Manage listeners**.
5. On the **Manage listeners** page, locate the listener to be updated and choose **Edit** under **Cookie stickiness**.
6. Select **Generated by application**.
7. For **Cookie Name**, type the name of your application cookie.
8. Choose **Save changes**.

**To enable application-controlled session stickiness using the AWS CLI**

1. Use the following create-app-cookie-stickiness-policy command to create an application-generated cookie stickiness policy:

```
aws elb create-app-cookie-stickiness-policy --load-balancer-name my-loadbalancer --
policy-name my-app-cookie-policy --cookie-name my-app-cookie
```

2. Use the following set-load-balancer-policies-of-listener command to enable session stickiness for a load balancer:

```
aws elb set-load-balancer-policies-of-listener --load-balancer-name my-loadbalancer
 --load-balancer-port 443 --policy-names my-app-cookie-policy
```

> **Note**
>
> The `set-load-balancer-policies-of-listener` command replaces the current set of policies associated with the specified load balancer port. Every time you use this command, specify the `--policy-names` option to list all policies to enable.

3. (Optional) Use the following describe-load-balancers command to verify that the sticky policy is enabled:

```
aws elb describe-load-balancers --load-balancer-name my-loadbalancer
```

4. The response includes the following information, which shows that the policy is enabled for the listener on the specified port:

```
{
    "LoadBalancerDescriptions": [
        {
            ...
            "ListenerDescriptions": [
                {
                    "Listener": {
                        "InstancePort": 443,
                        "SSLCertificateId": "arn:aws:iam::123456789012:server-
certificate/my-server-certificate",
                        "LoadBalancerPort": 443,
                        "Protocol": "HTTPS",
```

```
                        "InstanceProtocol": "HTTPS"
                    },
                    "PolicyNames": [
                        "my-app-cookie-policy",
                        "ELBSecurityPolicy-TLS-1-2-2017-01"
                    ]
                },
                {
                    "Listener": {
                        "InstancePort": 80,
                        "LoadBalancerPort": 80,
                        "Protocol": "TCP",
                        "InstanceProtocol": "TCP"
                    },
                    "PolicyNames": []
                }
            ],
            ...
            "Policies": {
                "LBCookieStickinessPolicies": [],
                "AppCookieStickinessPolicies": [
                {
                        "PolicyName": "my-app-cookie-policy",
                        "CookieName": "my-app-cookie"
                    }

                ],
                "OtherPolicies": [
                    "ELBSecurityPolicy-TLS-1-2-2017-01"
                ]
            },
            ...
        }
    ]
}
```

# Configure desync mitigation mode for your Classic Load Balancer

Desync mitigation mode protects your application from issues due to HTTP Desync. The load balancer classifies each request based on its threat level, allows safe requests, and then mitigates

risk as specified by the mitigation mode that you specify. The desync mitigation modes are monitor, defensive, and strictest. The default is the defensive mode, which provides durable mitigation against HTTP desync while maintaining the availability of your application. You can switch to strictest mode to ensure that your application receives only requests that comply with RFC 7230.

The http_desync_guardian library analyzes HTTP requests to prevent HTTP Desync attacks. For more information, see HTTP Desync Guardian on github.

**Contents**



> **Tip**
>
> This configuration applies only to Classic Load Balancers. For information that applies to Application Load Balancers, see Desync mitigation mode for Application Load Balancers.

## Classifications

The classifications are as follows.

- Compliant — Request complies with RFC 7230 and poses no known security threats.
- Acceptable — Request does not comply with RFC 7230 but poses no known security threats.
- Ambiguous — Request does not comply with RFC 7230 but poses a risk, as various web servers and proxies could handle it differently.
- Severe — Request poses a high security risk. The load balancer blocks the request, serves a 400 response to the client, and closes the client connection.

The following lists describe the issues for each classification.

**Acceptable**

- A header contains a non-ASCII or control character.

- The request version contains a bad value.
- There is a Content-Length header with a value of 0 for a GET or HEAD request.
- The request URI contains a space that is not URL encoded.

**Ambiguous**

- The request URI contains control characters.
- The request contains both a Transfer-Encoding header and a Content-Length header.
- There are multiple Content-Length headers with the same value.
- A header is empty or there is a line with only spaces.
- There is a header that can be normalized to Transfer-Encoding or Content-Length using common text normalization techniques.
- There is a Content-Length header for a GET or HEAD request.
- There is a Transfer-Encoding header for a GET or HEAD request.

**Severe**

- The request URI contains a null character or carriage return.
- The Content-Length header contains a value that cannot be parsed or is not a valid number.
- A header contains a null character or carriage return.
- The Transfer-Encoding header contains a bad value.
- The request method is malformed.
- The request version is malformed.
- There are multiple Content-Length headers with different values.
- There are multiple Transfer-Encoding: chunked headers.

If a request does not comply with RFC 7230, the load balancer increments the `DesyncMitigationMode_NonCompliant_Request_Count` metric. For more information, see Classic Load Balancer metrics.

## Modes

The following table describes how Classic Load Balancers treat requests based on mode and classification.

| Classification | Monitor mode | Defensive mode | Strictest mode |
|---|---|---|---|
| Compliant | Allowed | Allowed | Allowed |
| Acceptable | Allowed | Allowed | Blocked |
| Ambiguous | Allowed | Allowed[1] | Blocked |
| Severe | Allowed | Blocked | Blocked |

[1] Routes the requests but closes the client and target connections.

## Modify desync mitigation mode

### To update desync mitigation mode using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Load Balancing**, choose **Load Balancers**.
3. Choose the name of the load balancer to open its detail page.
4. On the **Attributes** tab, choose **Edit**.
5. On the **Edit load balancer attributes** page, under **Traffic configuration**, choose **Defensive - recommended**, **Strictest**, or **Monitor**.
6. Choose **Save changes**.

### To update desync mitigation mode using the AWS CLI

Use the modify-load-balancer-attributes command with the `elb.http.desyncmitigationmode` attribute set to `monitor`, `defensive`, or `strictest`.

```
aws elb modify-load-balancer-attributes --load-balancer-name my-load-balancer --load-
balancer-attributes file://attribute.json
```

The following is the contents of `attribute.json`.

```
{
    "AdditionalAttributes": [
        {
```

```
            "Key": "elb.http.desyncmitigationmode",
            "Value": "strictest"
        }
    ]
}
```

# Configure proxy protocol for your Classic Load Balancer

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

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

> **Note**
>
> The AWS Management Console does not support enabling proxy protocol.

**Contents**



## Proxy protocol header

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

The address of the proxy in the header for IPv6 is the public IPv6 address of your load balancer. This IPv6 address matches the IP address that is resolved from your load balancer's DNS name, which begins with either `ipv6` or `dualstack`. If the client connects with IPv4, the address of the proxy in the header is the private IPv4 address of the load balancer, which is not resolvable through a DNS lookup.

The proxy protocol line is a single line that ends with a carriage return and line feed (`"\r\n"`), and has the following form:

```
PROXY_STRING + single space + INET_PROTOCOL + single space + CLIENT_IP + single space +
 PROXY_IP + single space + CLIENT_PORT + single space + PROXY_PORT + "\r\n"
```

**Example: IPv4**

The following is an example of the proxy protocol line for IPv4.

```
PROXY TCP4 198.51.100.22 203.0.113.7 35646 80\r\n
```

## Prerequisites for enabling proxy protocol

Before you begin, do the following:

- Confirm that your load balancer is not behind a proxy server with proxy protocol enabled. If proxy protocol is enabled on both the proxy server and the load balancer, the load balancer adds another header to the request, which already has a header from the proxy server. Depending on how your instance is configured, this duplication might result in errors.
- Confirm that your instances can process the proxy protocol information.
- Confirm that your listener settings support proxy protocol. For more information, see Listener configurations for Classic Load Balancers.

## Enable proxy protocol using the AWS CLI

To enable proxy protocol, you must create a policy of type `ProxyProtocolPolicyType` and then enable the policy on the instance port.

Use the following procedure to create a new policy for your load balancer of type `ProxyProtocolPolicyType`, set the newly created policy to the instance on port `80`, and verify that the policy is enabled.

**To enable proxy protocol for your load balancer**

1. (Optional) Use the following describe-load-balancer-policy-types command to list the policies supported by Elastic Load Balancing:

```
aws elb describe-load-balancer-policy-types
```

   The response includes the names and descriptions of the supported policy types. The following shows the output for the `ProxyProtocolPolicyType` type:

```
{
    "PolicyTypeDescriptions": [
        ...
        {
            "PolicyAttributeTypeDescriptions": [
                {
                    "Cardinality": "ONE",
                    "AttributeName": "ProxyProtocol",
                    "AttributeType": "Boolean"
                }
            ],
            "PolicyTypeName": "ProxyProtocolPolicyType",
            "Description": "Policy that controls whether to include the IP address
 and port of the originating
request for TCP messages. This policy operates on TCP/SSL listeners only"
        },
        ...
    ]
}
```

2. Use the following create-load-balancer-policy command to create a policy that enables proxy protocol:

```
aws elb create-load-balancer-policy --load-balancer-name my-loadbalancer --policy-
name my-ProxyProtocol-policy --policy-type-name ProxyProtocolPolicyType --policy-
attributes AttributeName=ProxyProtocol,AttributeValue=true
```

3. Use the following set-load-balancer-policies-for-backend-server command to enable the newly created policy on the specified port. Note that this command replaces the current set of enabled policies. Therefore, the `--policy-names` option must specify both the policy that

you are adding to the list (for example, `my-ProxyProtocol-policy`) and any policies that are currently enabled (for example, `my-existing-policy`).

```
aws elb set-load-balancer-policies-for-backend-server --load-balancer-name my-loadbalancer --instance-port 80 --policy-names my-ProxyProtocol-policy my-existing-policy
```

4. (Optional) Use the following describe-load-balancers command to verify that proxy protocol is enabled:

```
aws elb describe-load-balancers --load-balancer-name my-loadbalancer
```

The response includes the following information, which shows that the `my-ProxyProtocol-policy` policy is associated with port `80`.

```
{
    "LoadBalancerDescriptions": [
        {
            ...
            "BackendServerDescriptions": [
                {
                    "InstancePort": 80,
                    "PolicyNames": [
                        "my-ProxyProtocol-policy"
                    ]
                }
            ],
            ...
        }
    ]
}
```

## Disable proxy protocol using the AWS CLI

You can disable the policies associated with your instance and then enable them at a later time.

**To disable the proxy protocol policy**

1. Use the following set-load-balancer-policies-for-backend-server command to disable the proxy protocol policy by omitting it from the `--policy-names` option, but including the other policies that should remain enabled (for example, `my-existing-policy`).

```
aws elb set-load-balancer-policies-for-backend-server --load-balancer-name my-loadbalancer --instance-port 80 --policy-names my-existing-policy
```

   If there are no other policies to enable, specify an empty string with `--policy-names` option as follows:

```
aws elb set-load-balancer-policies-for-backend-server --load-balancer-name my-loadbalancer --instance-port 80 --policy-names "[]"
```

2. (Optional) Use the following describe-load-balancers command to verify that the policy is disabled:

```
aws elb describe-load-balancers --load-balancer-name my-loadbalancer
```

   The response includes the following information, which shows that no ports are associated with a policy.

```
{
    "LoadBalancerDescriptions": [
        {
            ...
            "BackendServerDescriptions": [],
            ...
        }
    ]
}
```

# Tag your Classic Load Balancer

Tags help you to categorize your load balancers in different ways, for example, by purpose, owner, or environment.

You can add multiple tags to each Classic Load Balancer. Tag keys must be unique for each load balancer. If you add a tag with a key that is already associated with the load balancer, it updates the value of that tag.

When you are finished with a tag, you can remove it from your load balancer.

**Contents**



## Tag restrictions

The following basic restrictions apply to tags:

- Maximum number of tags per resource—50
- Maximum key length—127 Unicode characters
- Maximum value length—255 Unicode characters
- Tag keys and values are case sensitive. Allowed characters are letters, spaces, and numbers representable in UTF-8, plus the following special characters: + - = . _ : / @. Do not use leading or trailing spaces.
- Do not use the `aws:` prefix in your tag names or values because it is reserved for AWS use. You can't edit or delete tag names or values with this prefix. Tags with this prefix do not count against your tags per resource limit.

## Add a tag

You can add tags to your load balancer at any time.

**To add a tag using the console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Load Balancing**, choose **Load Balancers**.
3. Choose the name of the load balancer to open its detail page.
4. On the **Tags** tab, choose **Manage tags**.

5. On the **Manage tags** page, for each tag, choose **Add new tag** and then specify a key and a value.
6. After you have finished adding tags, choose **Save changes**.

#### To add a tag using the AWS CLI

Use the following add-tags command to add the specified tag:

```
aws elb add-tags --load-balancer-name my-loadbalancer --tag "Key=project,Value=lima"
```

### Remove a tag

You can remove tags from your load balancer whenever you are finished with them.

#### To remove a tag using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Load Balancing**, choose **Load Balancers**.
3. Choose the name of the load balancer to open its detail page.
4. On the **Tags** tab, choose **Manage tags**.
5. On the **Manage tags** page, choose **Remove** next to each tag you want to remove.
6. After you have finished removing tags, choose **Save changes**.

#### To remove a tag using the AWS CLI

Use the following remove-tags command to remove the tag with the specified key:

```
aws elb remove-tags --load-balancer-name my-loadbalancer --tag project
```

## Configure subnets for your Classic Load Balancer

When you add a subnet to your load balancer, Elastic Load Balancing creates a load balancer node in the Availability Zone. Load balancer nodes accept traffic from clients and forward requests to the healthy registered instances in one or more Availability Zones. We recommend that you add one subnet per Availability Zone for at least two Availability Zones. This improves the availability of your load balancer. Note that you can modify the subnets for your load balancer at any time.

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

To add a subnet, register the instances in the Availability Zone with the load balancer, then attach a subnet from that Availability Zone to the load balancer. For more information, see Register instances with your Classic Load Balancer.

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

You might want to remove a subnet from your load balancer temporarily when its Availability Zone has no healthy registered instances, or when you want to troubleshoot or update the registered instances. After you've removed a subnet, the load balancer stops routing requests to the registered instances in its Availability Zone, but continues to route requests to the registered instances in the Availability Zones for the remaining subnets. Note that after you remove a subnet, the instances in that subnet remain registered with the load balancer, but you can deregister them if you choose. For more information, see Register instances with your Classic Load Balancer.

**Contents**



## Requirements

When you update the subnets for your load balancer, you must meet the following requirements:

- The load balancer must have at least one subnet at all times.
- You can add at most one subnet per Availability Zone.
- You cannot add a Local Zone subnet.

Because there are separate APIs to add and remove subnets from a load balancer, you must consider the order of operations carefully when swapping the current subnets for new subnets in order to meet these requirements. Also, you must temporarily add a subnet from another Availability Zone if you need to swap all subnets for your load balancer. For example, if your load balancer has a single Availability Zone and you need to swap its subnet for another subnet, you must first add a subnet from a second Availability Zone. Then you can remove the subnet from the original Availability Zone (without going below one subnet), add a new subnet from the original Availability Zone (without exceeding one subnet per Availability Zone), and then remove the subnet from the second Availability Zone (if it is only needed to perform the swap).

## Configure subnets using the console

Use the following procedure to add or remove subnets using the console.

**To configure subnets using the console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Load Balancing**, choose **Load Balancers**.
3. Choose the name of the load balancer to open its detail page.
4. On the **Network mapping** tab, choose **Edit subnets**.
5. On the **Edit subnets** page, in the **Network mapping** section, add and remove subnets as needed..
6. When you are finished, choose **Save changes**.

## Configure subnets using the CLI

Use the following examples to add or remove subnets using the AWS CLI.

**To add a subnet to your load balancer using the CLI**

Use the following attach-load-balancer-to-subnets command to add two subnets to your load balancer:

```
aws elb attach-load-balancer-to-subnets --load-balancer-name my-load-balancer --
subnets subnet-dea770a9 subnet-fb14f6a2
```

The response lists all subnets for the load balancer. For example:

```
{
    "Subnets": [
        "subnet-5c11033e",
        "subnet-dea770a9",
        "subnet-fb14f6a2"
    ]
}
```

**To remove a subnet using the AWS CLI**

Use the following detach-load-balancer-from-subnets command to remove the specified subnets from the specified load balancer:

```
aws elb detach-load-balancer-from-subnets --load-balancer-name my-loadbalancer --
subnets subnet-450f5127
```

The response lists the remaining subnets for the load balancer. For example:

```
{
    "Subnets": [
        "subnet-15aaab61"
    ]
}
```

# Configure security groups for your Classic Load Balancer

When you use the AWS Management Console to create a load balancer, you can choose an existing security group or create a new one. If you choose an existing security group, it must allow traffic in both directions to the listener and health check ports for the load balancer. If you choose to create a security group, the console automatically adds rules to allow all traffic on these ports.

[Nondefault VPC] If you use the AWS CLI or API create a load balancer in a nondefault VPC, but you don't specify a security group, your load balancer is automatically associated with the default security group for the VPC.

[Default VPC] If you use the AWS CLI or API to create a load balancer in your default VPC, you can't choose an existing security group for your load balancer. Instead, Elastic Load Balancing provides a security group with rules to allow all traffic on the ports specified for the load balancer. Elastic

Load Balancing creates only one such security group per AWS account, with a name of the form default_elb_*id* (for example, `default_elb_fc5fbed3-0405-3b7d-a328-ea290EXAMPLE`). Subsequent load balancers that you create in the default VPC also use this security group. Be sure to review the security group rules to ensure that they allow traffic on the listener and health check ports for the new load balancer. When you delete your load balancer, this security group is not deleted automatically.

If you add a listener to an existing load balancer, you must review your security groups to ensure they allow traffic on the new listener port in both directions.

**Contents**



## Recommended rules for load balancer security groups

The security groups for your load balancers must allow them to communicate with your instances. The recommended rules depend on the type of load balancer, internet-facing or internal.

### Internet-facing load balancer

The following table shows the recommended inbound rules for an internet-facing load balancer.

| Source | Protocol | Port Range | Comment |
| --- | --- | --- | --- |
| 0.0.0.0/0 | TCP | *listener* | Allow all inbound traffic on the load balancer listener port |

The following table shows the recommended outbound rules for an internet-facing load balancer.

| Destination | Protocol | Port Range | Comment |
| --- | --- | --- | --- |
| *instance security group* | TCP | *instance listener* | Allow outbound traffic to instances on the instance listener port |

| Destination | Protocol | Port Range | Comment |
| --- | --- | --- | --- |
| *instance security group* | TCP | *health check* | Allow outbound traffic to instances on the health check port |

### Internal load balancers

The following table shows the recommended inbound rules for an internal load balancer.

| Source | Protocol | Port Range | Comment |
| --- | --- | --- | --- |
| *VPC CIDR* | TCP | *listener* | Allow inbound traffic from the VPC CIDR on the load balancer listener port |

The following table shows the recommended outbound rules for an internal load balancer.

| Destination | Protocol | Port Range | Comment |
| --- | --- | --- | --- |
| *instance security group* | TCP | *instance listener* | Allow outbound traffic to instances on the instance listener port |
| *instance security group* | TCP | *health check* | Allow outbound traffic to instances on the health check port |

## Assign security groups using the console

Use the following procedure to change the security groups associated with your load balancer.

**To update a security group assigned to your load balancer using the console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Load Balancing**, choose **Load Balancers**.

3. Choose the name of the load balancer to open its detail page.
4. On the **Security** tab, choose **Edit**.
5. On the **Edit security groups** page, Under **Security groups**, add or remove security groups as needed.

   You can add up to five security groups.
6. When you are finished, choose **Save changes**.

## Assign security groups using the AWS CLI

Use the following apply-security-groups-to-load-balancer command to associate a security group with a load balancer. The specified security groups override the previously associated security groups.

```
aws elb apply-security-groups-to-load-balancer --load-balancer-name my-loadbalancer --
security-groups sg-53fae93f
```

The following is an example response:

```
{
  "SecurityGroups": [
     "sg-53fae93f"
  ]
}
```

# Configure network ACLs for your Classic Load Balancer

The default network access control list (ACL) for a VPC allows all inbound and outbound traffic. If you create custom network ACLs, you must add rules that allow the load balancer and instances to communicate.

The recommended rules for the subnet for your load balancer depend on the type of load balancer, internet-facing or internal.

### Internet-facing load balancer

The following are the recommended inbound rules for an internet-facing load balancer.

| Source | Protocol | Port Range | Comment |
| --- | --- | --- | --- |
| 0.0.0.0/0 | TCP | *listener* | Allow all inbound traffic on the load balancer listener port |
| *VPC CIDR* | TCP | 1024-65535 | Allow inbound traffic from the VPC CIDR on the ephemeral ports |

The following are the recommended outbound rules for an internet-facing load balancer.

| Destination | Protocol | Port Range | Comment |
| --- | --- | --- | --- |
| *VPC CIDR* | TCP | *instance listener* | Allow all outbound traffic on the instance listener port |
| *VPC CIDR* | TCP | *health check* | Allow all outbound traffic on the health check port |
| 0.0.0.0/0 | TCP | 1024-65535 | Allow all outbound traffic on the ephemeral ports |

**Internal load balancer**

The following are the recommended inbound rules for an internal load balancer.

| Source | Protocol | Port Range | Comment |
| --- | --- | --- | --- |
| *VPC CIDR* | TCP | *listener* | Allow inbound traffic from the VPC CIDR on the load balancer listener port |
| *VPC CIDR* | TCP | 1024-65535 | Allow inbound traffic from the VPC CIDR on the ephemeral ports |

The following are the recommended outbound rules for an internal load balancer.

| Destination | Protocol | Port Range | Comment |
| --- | --- | --- | --- |
| *VPC CIDR* | TCP | *instance listener* | Allow outbound traffic to the VPC CIDR on the instance listener port |
| *VPC CIDR* | TCP | *health check* | Allow outbound traffic to the VPC CIDR on the health check port |
| *VPC CIDR* | TCP | 1024-65535 | Allow outbound traffic to the VPC CIDR on the ephemeral ports |

# Configure a custom domain name for your Classic Load Balancer

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

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

**Contents**

## Associating your custom domain name with your load balancer name

First, if you haven't already done so, register your domain name. The Internet Corporation for Assigned Names and Numbers (ICANN) manages domain names on the internet. You register a domain name using a *domain name registrar*, an ICANN-accredited organization that manages the registry of domain names. The website for your registrar will provide detailed instructions and pricing information for registering your domain name. For more information, see the following resources:

- To use Amazon Route 53 to register a domain name, see Registering domain names using Route 53 in the *Amazon Route 53 Developer Guide*.
- For a list of accredited registrars, see the List of Accredited Registrars.

Next, use your DNS service, such as your domain registrar, to create a CNAME record to route queries to your load balancer. For more information, see the documentation for your DNS service.

Alternatively, you can use Route 53 as your DNS service. You create a *hosted zone*, which contains information about how to route traffic on the internet for your domain, and an *alias resource record set*, which routes queries for your domain name to your load balancer. Route 53 doesn't charge for DNS queries for alias record sets, and you can use alias record sets to route DNS queries to your load balancer for the zone apex of your domain (for example, `example.com`). For information about transferring DNS services for existing domains to Route 53, see Configuring Route 53 as your DNS service in the *Amazon Route 53 Developer Guide*.

Finally, create a hosted zone and an alias record set for your domain using Route 53. For more information, see Routing traffic to a load balancer in the *Amazon Route 53 Developer Guide*.

## Using Route 53 DNS failover for your load balancer

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

For example, suppose that you have a web application for `www.example.com`, and you want redundant instances running behind two load balancers residing in different Regions. You want

the traffic to be primarily routed to the load balancer in one Region, and you want to use the load balancer in the other Region as a backup during failures. If you configure DNS failover, you can specify your primary and secondary (backup) load balancers. Route 53 directs traffic to the primary load balancer if it is available, or to the secondary load balancer otherwise.

**Using evaluate target health**

- When evaluate target health is set to `Yes` on an alias record for a Classic Load Balancer, Route 53 evaluates the health of the resource specified by the `alias target` value. For a Classic Load Balancer, Route 53 uses the instance health checks associated with the load balancer.
- When at least one of the registered instances in a Classic Load Balancer is healthy, Route 53 marks the alias record as healthy. Route 53 then returns records according to your routing policy. If the failover routing policy is used, Route 53 returns the primary record.
- When all the registered instances for a Classic Load Balancer are unhealthy, Route 53 marks the alias record as unhealthy. Route 53 then returns records according to your routing policy. If the failover routing policy is used, then Route 53 returns the secondary record.

For more information, see Configuring DNS failover in the *Amazon Route 53 Developer Guide*.

## Disassociating your custom domain name from your load balancer

You can disassociate your custom domain name from a load balancer instance by first deleting the resource record sets in your hosted zone and then deleting the hosted zone. For more information, see Editing records and Deleting a public hosted zone in the *Amazon Route 53 Developer Guide*.

# Listeners for your Classic Load Balancer

Before you start using Elastic Load Balancing, you must configure one or more *listeners* for your Classic Load Balancer. A listener is a process that checks for connection requests. It is configured with a protocol and a port for front-end (client to load balancer) connections, and a protocol and a port for back-end (load balancer to back-end instance) connections.

Elastic Load Balancing supports the following protocols:

- HTTP
- HTTPS (secure HTTP)
- TCP
- SSL (secure TCP)

The HTTPS protocol uses the SSL protocol to establish secure connections over the HTTP layer. You can also use the SSL protocol to establish secure connections over the TCP layer.

If the front-end connection uses TCP or SSL, then your back-end connections can use either TCP or SSL. If the front-end connection uses HTTP or HTTPS, then your back-end connections can use either HTTP or HTTPS.

Back-end instances can listen on ports 1-65535.

Load balancers can listen on the following ports: 1-65535

**Contents**



# Protocols

Communication for a typical web application goes through layers of hardware and software. Each layer provides a specific communication function. The control over the communication function is passed from one layer to the next, in sequence. The Open System Interconnection (OSI) defines

a model framework for implementing a standard format for communication, called a *protocol*, in these layers. For more information, see OSI model in Wikipedia.

When you use Elastic Load Balancing, you need a basic understanding of layer 4 and layer 7. Layer 4 is the transport layer that describes the Transmission Control Protocol (TCP) connection between the client and your back-end instance, through the load balancer. Layer 4 is the lowest level that is configurable for your load balancer. Layer 7 is the application layer that describes the use of Hypertext Transfer Protocol (HTTP) and HTTPS (secure HTTP) connections from clients to the load balancer and from the load balancer to your back-end instance.

The Secure Sockets Layer (SSL) protocol is primarily used to encrypt confidential data over insecure networks such as the internet. The SSL protocol establishes a secure connection between a client and the back-end server, and ensures that all the data passed between your client and your server is private and integral.

## TCP/SSL protocol

When you use TCP (layer 4) for both front-end and back-end connections, your load balancer forwards the request to the back-end instances without modifying the headers. After your load balancer receives the request, it attempts to open a TCP connection to the back-end instance on the port specified in the listener configuration.

Because load balancers intercept traffic between clients and your back-end instances, the access logs for your back-end instance contain the IP address of the load balancer instead of the originating client. You can enable proxy protocol, which adds a header with the connection information of the client, such as the source IP address, destination IP address, and port numbers. The header is then sent to the back-end instance as a part of the request. You can parse the first line in the request to retrieve the connection information. For more information, see Configure proxy protocol for your Classic Load Balancer.

Using this configuration, you do not receive cookies for session stickiness or X-Forwarded headers.

## HTTP/HTTPS protocol

When you use HTTP (layer 7) for both front-end and back-end connections, your load balancer parses the headers in the request before sending the request to the back-end instances.

For every registered and healthy instance behind an HTTP/HTTPS load balancer, Elastic Load Balancing opens and maintains one or more TCP connections. These connections ensure that there is always an established connection ready to receive HTTP/HTTPS requests.

The HTTP requests and HTTP responses use header fields to send information about HTTP messages. Elastic Load Balancing supports `X-Forwarded-For` headers. Because load balancers intercept traffic between clients and servers, your server access logs contain only the IP address of the load balancer. To see the IP address of the client, use the `X-Forwarded-For` request header. For more information, see X-Forwarded-For.

When you use HTTP/HTTPS, you can enable sticky sessions on your load balancer. A sticky session binds a user's session to a specific back-end instance. This ensures that all requests coming from the user during the session are sent to the same back-end instance. For more information, see Configure sticky sessions for your Classic Load Balancer.

Not all HTTP extensions are supported in the load balancer. You may need to use a TCP listener if the load balancer is not able to terminate the request due to unexpected methods, response codes, or other non-standard HTTP 1.0/1.1 implementations.

# HTTPS/SSL listeners

You can create a load balancer with the following security features.

## SSL server certificates

If you use HTTPS or SSL for your front-end connections, you must deploy an X.509 certificate (SSL server certificate) on your load balancer. The load balancer decrypts requests from clients before sending them to the back-end instances (known as *SSL termination*). For more information, see SSL/TLS certificates for Classic Load Balancers.

If you don't want the load balancer to handle the SSL termination (known as *SSL offloading*), you can use TCP for both the front-end and back-end connections, and deploy certificates on the registered instances handling requests.

## SSL negotiation

Elastic Load Balancing provides predefined SSL negotiation configurations that are used for SSL negotiation when a connection is established between a client and your load balancer. The SSL negotiation configurations provide compatibility with a broad range of clients and use high-strength cryptographic algorithms called *ciphers*. However, some use cases might require all data on the network to be encrypted and allow only specific ciphers. Some security compliance standards (such as PCI, SOX, and so on) might require a specific set of protocols and ciphers from

clients to ensure that the security standards are met. In such cases, you can create a custom SSL negotiation configuration, based on your specific requirements. Your ciphers and protocols should take effect within 30 seconds. For more information, see SSL negotiation configurations for Classic Load Balancers.

### Back-end server authentication

If you use HTTPS or SSL for your back-end connections, you can enable authentication of your registered instances. You can then use the authentication process to ensure that the instances accept only encrypted communication, and to ensure that each registered instance has the correct public key.

For more information, see Configure Back-end Server Authentication.

## Listener configurations for Classic Load Balancers

The following table describes possible configurations for HTTP and HTTPS listeners for a Classic Load Balancer.

| Use case | Front-end protocol | Front-end options | Back-end protocol | Back-end options | Notes |
|---|---|---|---|---|---|
| Basic HTTP load balancer | HTTP | NA | HTTP | NA | • Supports the X-Forwarded headers |
| Secure website or application using Elastic Load Balancing to offload SSL decryption | HTTPS | SSL negotiation | HTTP | NA | • Supports the X-Forwarded headers<br>• Requires an SSL certificate deployed on the load balancer |

| Use case | Front-end protocol | Front-end options | Back-end protocol | Back-end options | Notes |
|---|---|---|---|---|---|
| Secure website or application using end-to-end encryption | HTTPS | SSL negotiation | HTTPS | Back-end authentication | • Supports the X-Forwarded headers<br>• Requires SSL certificates deployed on the load balancer and the registered instances |

The following table describes possible configurations for TCP and SSL listeners for a Classic Load Balancer.

| Use case | Front-end protocol | Front-end options | Back-end protocol | Back-end options | Notes |
|---|---|---|---|---|---|
| Basic TCP load balancer | TCP | NA | TCP | NA | • Supports the proxy protocol header |
| Secure website or application using Elastic Load Balancing to offload SSL decryption | SSL | SSL negotiation | TCP | NA | • Requires an SSL certificate deployed on the load balancer<br>• Supports the proxy |

| Use case | Front-end protocol | Front-end options | Back-end protocol | Back-end options | Notes |
|---|---|---|---|---|---|
| | | | | | protocol header |
| Secure website or application using end-to-end encryption with Elastic Load Balancing | SSL | SSL negotiation | SSL | Back-end authentication | • Requires SSL certificates deployed on the load balancer and the registered instances<br>• Does not insert SNI headers on back-end SSL connections<br>• Does not support the proxy protocol header |

## HTTP headers and Classic Load Balancers

HTTP requests and HTTP responses use header fields to send information about the HTTP messages. Header fields are colon-separated name-value pairs that are separated by a carriage return (CR) and a line feed (LF). A standard set of HTTP header fields is defined in RFC 2616, Message Headers. There are also non-standard HTTP headers available (and automatically added) that are widely used by the applications. Some of the non-standard HTTP headers have an `X-Forwarded` prefix. Classic Load Balancers support the following `X-Forwarded` headers.

For more information about HTTP connections, see Request routing in the *Elastic Load Balancing User Guide*.

**Prerequisites**

- Confirm that your listener settings support the X-Forwarded headers. For more information, see Listener configurations for Classic Load Balancers.
- Configure your web server to log client IP addresses.

**X-Forwarded headers**

- X-Forwarded-For
- X-Forwarded-Proto
- X-Forwarded-Port

## X-Forwarded-For

The `X-Forwarded-For` request header is automatically added and helps you identify the IP address of a client when you use an HTTP or HTTPS load balancer. Because load balancers intercept traffic between clients and servers, your server access logs contain only the IP address of the load balancer. To see the IP address of the client, use the `X-Forwarded-For` request header. Elastic Load Balancing stores the IP address of the client in the `X-Forwarded-For` request header and passes the header to your server. If the `X-Forwarded-For` request header is not included in the request, the load balancer creates one with the client IP address as the request value. Otherwise, the load balancer appends the client IP address to the existing header and passes the header to your server. The `X-Forwarded-For` request header may contain multiple IP addresses that are comma separated. The left-most address is the client IP where the request was first made. This is followed by any subsequent proxy identifiers, in a chain.

The `X-Forwarded-For` request header takes the following form:

```
X-Forwarded-For: client-ip-address
```

The following is an example `X-Forwarded-For` request header for a client with an IP address of `203.0.113.7`.

```
X-Forwarded-For: 203.0.113.7
```

The following is an example `X-Forwarded-For` request header for a client with an IPv6 address of `2001:DB8::21f:5bff:febf:ce22:8a2e`.

```
X-Forwarded-For: 2001:DB8::21f:5bff:febf:ce22:8a2e
```

## X-Forwarded-Proto

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

Your application or website can use the protocol stored in the `X-Forwarded-Proto` request header to render a response that redirects to the appropriate URL.

The `X-Forwarded-Proto` request header takes the following form:

```
X-Forwarded-Proto: originatingProtocol
```

The following example contains an `X-Forwarded-Proto` request header for a request that originated from the client as an HTTPS request:

```
X-Forwarded-Proto: https
```

## X-Forwarded-Port

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

# HTTPS listeners for your Classic Load Balancer

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

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

Classic Load Balancers do not support Server Name Indication (SNI). You can use one of the following alternatives instead:

- Deploy one certificate on the load balancer, and add a Subject Alternative Name (SAN) for each additional website. SANs enable you to protect multiple host names using a single certificate. Check with your certificate provider for more information about the number of SANs they support per certificate and how to add and remove SANs.
- Use TCP listeners on port 443 for the front-end and back-end connections. The load balancer passes the request through as is, so you can handle HTTPS termination on the EC2 instance.

Classic Load Balancers do not support mutual TLS authentication (mTLS). For mTLS support, create a TCP listener. The load balancer passes the request through as is, so you can implement mTLS on the EC2 instance.

**Contents**

- Update the SSL negotiation configuration of your Classic Load Balancer

# SSL/TLS certificates for Classic Load Balancers

If you use HTTPS (SSL or TLS) for your front-end listener, you must deploy an SSL/TLS certificate on your load balancer. The load balancer uses the certificate to terminate the connection and then decrypt requests from clients before sending them to the instances.

The SSL and TLS protocols use an X.509 certificate (SSL/TLS server certificate) to authenticate both the client and the back-end application. An X.509 certificate is a digital form of identification issued by a certificate authority (CA) and contains identification information, a validity period, a public key, a serial number, and the digital signature of the issuer.

You can create a certificate using AWS Certificate Manager or a tool that supports the SSL and TLS protocols, such as OpenSSL. You will specify this certificate when you create or update an HTTPS listener for your load balancer. When you create a certificate for use with your load balancer, you must specify a domain name.

When you create a certificate for use with your load balancer, you must specify a domain name. The domain name on the certificate must match the custom domain name record. If they do not match, the traffic will not be encrypted as the TLS connection cannot be verified.

You must specify a fully qualified domain name (FQDN) for your certificate, such as `www.example.com` or an apex domain name such as `example.com`. You can also use an asterisk (*) as a wild card to protect several site names in the same domain. When you request a wild-card certificate, the asterisk (*) must be in the leftmost position of the domain name and can protect only one subdomain level. For instance, `*.example.com` protects `corp.example.com`, and `images.example.com`, but it cannot protect `test.login.example.com`. Also note that `*.example.com` protects only the subdomains of `example.com`, it does not protect the bare or apex domain (`example.com`). The wild-card name will appear in the **Subject** field and in the **Subject Alternative Name** extension of the certificate. For more information about public certificates, see Requesting a public certificate in the *AWS Certificate Manager User Guide*.

## Create or import an SSL/TLS certificate using AWS Certificate Manager

We recommend that you use AWS Certificate Manager (ACM) to create or import certificates for your load balancer. ACM integrates with Elastic Load Balancing so that you can deploy the certificate on your load balancer. To deploy a certificate on your load balancer, the certificate must

be in the same Region as the load balancer. For more information, see Request a public certificate or Importing certificates in the *AWS Certificate Manager User Guide*.

To allow a user to deploy the certificate on your load balancer using the AWS Management Console, you must allow access to the ACM `ListCertificates` API action. For more information, see Listing certificates in the *AWS Certificate Manager User Guide*.

> **Important**
>
> You cannot install certificates with 4096-bit RSA keys or EC keys on your load balancer through integration with ACM. You must upload certificates with 4096-bit RSA keys or EC keys to IAM in order to use them with your load balancer.

### Import an SSL/TLS certificate using IAM

If you are not using ACM, you can use SSL/TLS tools, such as OpenSSL, to create a certificate signing request (CSR), get the CSR signed by a CA to produce a certificate, and upload the certificate to IAM. For more information, see Working with server certificates in the *IAM User Guide*.

## SSL negotiation configurations for Classic Load Balancers

Elastic Load Balancing uses a Secure Socket Layer (SSL) negotiation configuration, known as a *security policy*, to negotiate SSL connections between a client and the load balancer. A security policy is a combination of SSL protocols, SSL ciphers, and the Server Order Preference option. For more information about configuring an SSL connection for your load balancer, see Listeners for your Classic Load Balancer.

**Contents**

# Security policies

A security policy determines which ciphers and protocols are supported during SSL negotiations between a client and a load balancer. You can configure your Classic Load Balancers to use either predefined or custom security policies.

Note that a certificate provided by AWS Certificate Manager (ACM) contains an RSA public key. Therefore, you must include a cipher suite that uses RSA in your security policy if you use a certificate provided by ACM; otherwise, the TLS connection fails.

### Predefined security policies

The names of the most recent predefined security policies includes version information based on the year and month that they were released. For example, the default predefined security policy is `ELBSecurityPolicy-2016-08`. Whenever a new predefined security policy is released, you can update your configuration to use it.

For information about the protocols and ciphers enabled for the predefined security policies, see Predefined SSL security policies for Classic Load Balancers.

### Custom security policies

You can create a custom negotiation configuration with the ciphers and protocols that you need. For example, some security compliance standards (such as PCI and SOC) might require a specific set of protocols and ciphers to ensure that the security standards are met. In such cases, you can create a custom security policy to meet those standards.

For information about creating a custom security policy, see Update the SSL negotiation configuration of your Classic Load Balancer.

# SSL protocols

The *SSL protocol* establishes a secure connection between a client and a server, and ensures that all the data passed between the client and your load balancer is private.

Secure Sockets Layer (SSL) and Transport Layer Security (TLS) are cryptographic protocols that are used to encrypt confidential data over insecure networks such as the internet. The TLS protocol is a newer version of the SSL protocol. In the Elastic Load Balancing documentation, we refer to both SSL and TLS protocols as the SSL protocol.

### Recommended protocol

We recommend TLS 1.2, which is used in the ELBSecurityPolicy-TLS-1-2-2017-01 predefined security policy. You can also use TLS 1.2 in your custom security policies. The default security policy supports both TLS 1.2 and earlier versions of TLS, so it is less secure than ELBSecurityPolicy-TLS-1-2-2017-01.

**Deprecated protocol**

If you previously enabled the SSL 2.0 protocol in a custom policy, we recommend that you update your security policy to one of the predefined security policies.

## Server Order Preference

Elastic Load Balancing supports the *Server Order Preference* option for negotiating connections between a client and a load balancer. During the SSL connection negotiation process, the client and the load balancer present a list of ciphers and protocols that they each support, in order of preference. By default, the first cipher on the client's list that matches any one of the load balancer's ciphers is selected for the SSL connection. If the load balancer is configured to support Server Order Preference, then the load balancer selects the first cipher in its list that is in the client's list of ciphers. This ensures that the load balancer determines which cipher is used for SSL connection. If you do not enable Server Order Preference, the order of ciphers presented by the client is used to negotiate connections between the client and the load balancer.

## SSL ciphers

An *SSL cipher* is an encryption algorithm that uses encryption keys to create a coded message. SSL protocols use several SSL ciphers to encrypt data over the internet.

Note that a certificate provided by AWS Certificate Manager (ACM) contains an RSA public key. Therefore, you must include a cipher suite that uses RSA in your security policy if you use a certificate provided by ACM; otherwise, the TLS connection fails.

Elastic Load Balancing supports the following ciphers for use with Classic Load Balancers. A subset of these ciphers are used by the predefined SSL policies. All of these ciphers are available for use in a custom policy. We recommend that you use only the ciphers included in the default security policy (those with an asterisk). Many of the other ciphers are not secure and should be used at your own risk.

**Ciphers**

- ECDHE-ECDSA-AES128-GCM-SHA256 *

- ECDHE-RSA-AES128-GCM-SHA256 *
- ECDHE-ECDSA-AES128-SHA256 *
- ECDHE-RSA-AES128-SHA256 *
- ECDHE-ECDSA-AES128-SHA *
- ECDHE-RSA-AES128-SHA *
- DHE-RSA-AES128-SHA
- ECDHE-ECDSA-AES256-GCM-SHA384 *
- ECDHE-RSA-AES256-GCM-SHA384 *
- ECDHE-ECDSA-AES256-SHA384 *
- ECDHE-RSA-AES256-SHA384 *
- ECDHE-RSA-AES256-SHA *
- ECDHE-ECDSA-AES256-SHA *
- AES128-GCM-SHA256 *
- AES128-SHA256 *
- AES128-SHA *
- AES256-GCM-SHA384 *
- AES256-SHA256 *
- AES256-SHA *
- DHE-DSS-AES128-SHA
- CAMELLIA128-SHA
- EDH-RSA-DES-CBC3-SHA
- DES-CBC3-SHA
- ECDHE-RSA-RC4-SHA
- RC4-SHA
- ECDHE-ECDSA-RC4-SHA
- DHE-DSS-AES256-GCM-SHA384
- DHE-RSA-AES256-GCM-SHA384
- DHE-RSA-AES256-SHA256
- DHE-DSS-AES256-SHA256

- DHE-RSA-AES256-SHA
- DHE-DSS-AES256-SHA
- DHE-RSA-CAMELLIA256-SHA
- DHE-DSS-CAMELLIA256-SHA
- CAMELLIA256-SHA
- EDH-DSS-DES-CBC3-SHA
- DHE-DSS-AES128-GCM-SHA256
- DHE-RSA-AES128-GCM-SHA256
- DHE-RSA-AES128-SHA256
- DHE-DSS-AES128-SHA256
- DHE-RSA-CAMELLIA128-SHA
- DHE-DSS-CAMELLIA128-SHA
- ADH-AES128-GCM-SHA256
- ADH-AES128-SHA
- ADH-AES128-SHA256
- ADH-AES256-GCM-SHA384
- ADH-AES256-SHA
- ADH-AES256-SHA256
- ADH-CAMELLIA128-SHA
- ADH-CAMELLIA256-SHA
- ADH-DES-CBC3-SHA
- ADH-DES-CBC-SHA
- ADH-RC4-MD5
- ADH-SEED-SHA
- DES-CBC-SHA
- DHE-DSS-SEED-SHA
- DHE-RSA-SEED-SHA
- EDH-DSS-DES-CBC-SHA
- EDH-RSA-DES-CBC-SHA
- IDEA-CBC-SHA

- RC4-MD5
- SEED-SHA
- DES-CBC3-MD5
- DES-CBC-MD5
- RC2-CBC-MD5
- PSK-AES256-CBC-SHA
- PSK-3DES-EDE-CBC-SHA
- KRB5-DES-CBC3-SHA
- KRB5-DES-CBC3-MD5
- PSK-AES128-CBC-SHA
- PSK-RC4-SHA
- KRB5-RC4-SHA
- KRB5-RC4-MD5
- KRB5-DES-CBC-SHA
- KRB5-DES-CBC-MD5
- EXP-EDH-RSA-DES-CBC-SHA
- EXP-EDH-DSS-DES-CBC-SHA
- EXP-ADH-DES-CBC-SHA
- EXP-DES-CBC-SHA
- EXP-RC2-CBC-MD5
- EXP-KRB5-RC2-CBC-SHA
- EXP-KRB5-DES-CBC-SHA
- EXP-KRB5-RC2-CBC-MD5
- EXP-KRB5-DES-CBC-MD5
- EXP-ADH-RC4-MD5
- EXP-RC4-MD5
- EXP-KRB5-RC4-SHA
- EXP-KRB5-RC4-MD5

* These are the ciphers included in the default security policy, ELBSecurityPolicy-2016-08.

# Cipher suite for back-end connections

Classic Load Balancers uses a static cipher suite for back-end connections. If your Classic Load Balancer and registered instances can't negotiate a connection, include one of the following ciphers.

- AES256-GCM-SHA384
- AES256-SHA256
- AES256-SHA
- CAMELLIA256-SHA
- AES128-GCM-SHA256
- AES128-SHA256
- AES128-SHA
- CAMELLIA128-SHA
- RC4-SHA
- DES-CBC3-SHA
- DES-CBC-SHA
- DHE-DSS-AES256-GCM-SHA384
- DHE-RSA-AES256-GCM-SHA384
- DHE-RSA-AES256-SHA256
- DHE-DSS-AES256-SHA256
- DHE-RSA-AES256-SHA
- DHE-DSS-AES256-SHA
- DHE-RSA-CAMELLIA256-SHA
- DHE-DSS-CAMELLIA256-SHA
- DHE-DSS-AES128-GCM-SHA256
- DHE-RSA-AES128-GCM-SHA256
- DHE-RSA-AES128-SHA256
- DHE-DSS-AES128-SHA256
- DHE-RSA-AES128-SHA
- DHE-DSS-AES128-SHA
- DHE-RSA-CAMELLIA128-SHA

- DHE-DSS-CAMELLIA128-SHA
- EDH-RSA-DES-CBC3-SHA
- EDH-DSS-DES-CBC3-SHA
- EDH-RSA-DES-CBC-SHA
- EDH-DSS-DES-CBC-SHA

# Predefined SSL security policies for Classic Load Balancers

You can choose one of the predefined security policies for your HTTPS/SSL listeners. You can use one of the `ELBSecurityPolicy-TLS` policies to meet compliance and security standards that require disabling certain TLS protocol versions. Alternatively, you can create a custom security policy. For more information, see Update the SSL negotiation configuration.

The RSA- and DSA-based ciphers are specific to the signing algorithm used to create SSL certificate. Make sure to create an SSL certificate using the signing algorithm that is based on the ciphers that are enabled for your security policy.

If you select a policy that is enabled for Server Order Preference, the load balancer uses the ciphers in the order that they are specified here to negotiate connections between the client and load balancer. Otherwise, the load balancer uses the ciphers in the order that they are presented by the client.

The following sections describe the most recent predefined security policies for Classic Load Balancers, including their enabled SSL protocols and SSL ciphers. You can also describe the predefined policies using the describe-load-balancer-policies command.

> **Tip**
>
> This information applies only to Classic Load Balancers. For information that applies to other load balancers, see Security policies for your Application Load Balancer and Security policies for your Network Load Balancer.

**Contents**

- Policies by cipher

## Protocols by policy

The following table describes the TLS protocols that each security policy supports.

| Security policies | TLS 1.2 | TLS 1.1 | TLS 1.0 |
|---|---|---|---|
| ELBSecurityPolicy-TLS-1-2-2017-01 | Yes | No | No |
| ELBSecurityPolicy-TLS-1-1-2017-01 | Yes | Yes | No |
| ELBSecurityPolicy-2016-08 | Yes | Yes | Yes |
| ELBSecurityPolicy-2015-05 | Yes | Yes | Yes |
| ELBSecurityPolicy-2015-03 | Yes | Yes | Yes |
| ELBSecurityPolicy-2015-02 | Yes | Yes | Yes |

## Ciphers by policy

The following table describes the ciphers that each security policy supports.

| Security policy | Ciphers |
|---|---|
| ELBSecurityPolicy-TLS-1-2-2017-01 | • ECDHE-ECDSA-AES128-GCM-SHA256<br>• ECDHE-RSA-AES128-GCM-SHA256<br>• ECDHE-ECDSA-AES128-SHA256<br>• ECDHE-RSA-AES128-SHA256<br>• ECDHE-ECDSA-AES256-GCM-SHA384<br>• ECDHE-RSA-AES256-GCM-SHA384<br>• ECDHE-ECDSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA384<br>• AES128-GCM-SHA256<br>• AES128-SHA256 |

| Security policy | Ciphers |
| --- | --- |
| | • AES256-GCM-SHA384<br>• AES256-SHA256 |
| ELBSecurityPolicy-TLS-1-1-2017-01 | • ECDHE-ECDSA-AES128-GCM-SHA256<br>• ECDHE-RSA-AES128-GCM-SHA256<br>• ECDHE-ECDSA-AES128-SHA256<br>• ECDHE-RSA-AES128-SHA256<br>• ECDHE-ECDSA-AES128-SHA<br>• ECDHE-RSA-AES128-SHA<br>• ECDHE-ECDSA-AES256-GCM-SHA384<br>• ECDHE-RSA-AES256-GCM-SHA384<br>• ECDHE-ECDSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA384<br>• ECDHE-ECDSA-AES256-SHA<br>• ECDHE-RSA-AES256-SHA<br>• AES128-GCM-SHA256<br>• AES128-SHA256<br>• AES128-SHA<br>• AES256-GCM-SHA384<br>• AES256-SHA256<br>• AES256-SHA |

| Security policy | Ciphers |
| --- | --- |
| ELBSecurityPolicy-2016-08 | • ECDHE-ECDSA-AES128-GCM-SHA256<br>• ECDHE-RSA-AES128-GCM-SHA256<br>• ECDHE-ECDSA-AES128-SHA256<br>• ECDHE-RSA-AES128-SHA256<br>• ECDHE-ECDSA-AES128-SHA<br>• ECDHE-RSA-AES128-SHA<br>• ECDHE-ECDSA-AES256-GCM-SHA384<br>• ECDHE-RSA-AES256-GCM-SHA384<br>• ECDHE-ECDSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA384<br>• ECDHE-ECDSA-AES256-SHA<br>• ECDHE-RSA-AES256-SHA<br>• AES128-GCM-SHA256<br>• AES128-SHA256<br>• AES128-SHA<br>• AES256-GCM-SHA384<br>• AES256-SHA256<br>• AES256-SHA |

| Security policy | Ciphers |
| --- | --- |
| ELBSecurityPolicy-2015-05 | • ECDHE-ECDSA-AES128-GCM-SHA256<br>• ECDHE-RSA-AES128-GCM-SHA256<br>• ECDHE-ECDSA-AES128-SHA256<br>• ECDHE-RSA-AES128-SHA256<br>• ECDHE-ECDSA-AES128-SHA<br>• ECDHE-RSA-AES128-SHA<br>• ECDHE-ECDSA-AES256-GCM-SHA384<br>• ECDHE-RSA-AES256-GCM-SHA384<br>• ECDHE-ECDSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA384<br>• ECDHE-ECDSA-AES256-SHA<br>• ECDHE-RSA-AES256-SHA<br>• AES128-GCM-SHA256<br>• AES128-SHA256<br>• AES128-SHA<br>• AES256-GCM-SHA384<br>• AES256-SHA256<br>• AES256-SHA<br>• DES-CBC3-SHA |

| Security policy | Ciphers |
| --- | --- |
| ELBSecurityPolicy-2015-03 | • ECDHE-ECDSA-AES128-GCM-SHA256<br>• ECDHE-RSA-AES128-GCM-SHA256<br>• ECDHE-ECDSA-AES128-SHA256<br>• ECDHE-RSA-AES128-SHA256<br>• ECDHE-ECDSA-AES128-SHA<br>• ECDHE-RSA-AES128-SHA<br>• ECDHE-ECDSA-AES256-GCM-SHA384<br>• ECDHE-RSA-AES256-GCM-SHA384<br>• ECDHE-ECDSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA384<br>• ECDHE-ECDSA-AES256-SHA<br>• ECDHE-RSA-AES256-SHA<br>• AES128-GCM-SHA256<br>• AES128-SHA256<br>• AES128-SHA<br>• AES256-GCM-SHA384<br>• AES256-SHA256<br>• AES256-SHA<br>• DHE-RSA-AES128-SHA<br>• DHE-DSS-AES128-SHA<br>• DES-CBC3-SHA |

| Security policy | Ciphers |
| --- | --- |
| ELBSecurityPolicy-2015-02 | • ECDHE-ECDSA-AES128-GCM-SHA256<br>• ECDHE-RSA-AES128-GCM-SHA256<br>• ECDHE-ECDSA-AES128-SHA256<br>• ECDHE-RSA-AES128-SHA256<br>• ECDHE-ECDSA-AES128-SHA<br>• ECDHE-RSA-AES128-SHA<br>• ECDHE-ECDSA-AES256-GCM-SHA384<br>• ECDHE-RSA-AES256-GCM-SHA384<br>• ECDHE-ECDSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA384<br>• ECDHE-ECDSA-AES256-SHA<br>• ECDHE-RSA-AES256-SHA<br>• AES128-GCM-SHA256<br>• AES128-SHA256<br>• AES128-SHA<br>• AES256-GCM-SHA384<br>• AES256-SHA256<br>• AES256-SHA<br>• DHE-RSA-AES128-SHA<br>• DHE-DSS-AES128-SHA |

## Policies by cipher

The following table describes the security policies that support each cipher.

| Cipher name | Security policies | Cipher suite |
| --- | --- | --- |
| **OpenSSL** – ECDHE-ECDSA-AES128-GCM-SHA256 | • ELBSecurityPolicy-TLS-1-2-2017-01<br>• ELBSecurityPolicy-TLS-1-1-2017-01 | c02b |

| Cipher name | Security policies | Cipher suite |
|---|---|---|
| **IANA** – TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 | • ELBSecurityPolicy-2016-08<br>• ELBSecurityPolicy-2015-05<br>• ELBSecurityPolicy-2015-03<br>• ELBSecurityPolicy-2015-02 | |
| **OpenSSL** – ECDHE-RSA-AES128-GCM-SHA256<br>**IANA** – TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 | • ELBSecurityPolicy-TLS-1-2-2017-01<br>• ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08<br>• ELBSecurityPolicy-2015-05<br>• ELBSecurityPolicy-2015-03<br>• ELBSecurityPolicy-2015-02 | c02f |
| **OpenSSL** – ECDHE-ECDSA-AES128-SHA256<br>**IANA** – TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 | • ELBSecurityPolicy-TLS-1-2-2017-01<br>• ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08<br>• ELBSecurityPolicy-2015-05<br>• ELBSecurityPolicy-2015-03<br>• ELBSecurityPolicy-2015-02 | c023 |
| **OpenSSL** – ECDHE-RSA-AES128-SHA256<br>**IANA** – TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 | • ELBSecurityPolicy-TLS-1-2-2017-01<br>• ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08<br>• ELBSecurityPolicy-2015-05<br>• ELBSecurityPolicy-2015-03<br>• ELBSecurityPolicy-2015-02 | c027 |

| Cipher name | Security policies | Cipher suite |
|---|---|---|
| **OpenSSL** – ECDHE-ECDSA-AES128-SHA<br><br>**IANA** – TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA | • ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08<br>• ELBSecurityPolicy-2015-05<br>• ELBSecurityPolicy-2015-03<br>• ELBSecurityPolicy-2015-02 | c009 |
| **OpenSSL** – ECDHE-RSA-AES128-SHA<br><br>**IANA** – TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA | • ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08<br>• ELBSecurityPolicy-2015-05<br>• ELBSecurityPolicy-2015-03<br>• ELBSecurityPolicy-2015-02 | c013 |
| **OpenSSL** – ECDHE-ECDSA-AES256-GCM-SHA384<br><br>**IANA** – TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 | • ELBSecurityPolicy-TLS-1-2-2017-01<br>• ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08<br>• ELBSecurityPolicy-2015-05<br>• ELBSecurityPolicy-2015-03<br>• ELBSecurityPolicy-2015-02 | c02c |
| **OpenSSL** – ECDHE-RSA-AES256-GCM-SHA384<br><br>**IANA** – TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 | • ELBSecurityPolicy-TLS-1-2-2017-01<br>• ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08<br>• ELBSecurityPolicy-2015-05<br>• ELBSecurityPolicy-2015-03<br>• ELBSecurityPolicy-2015-02 | c030 |

| Cipher name | Security policies | Cipher suite |
| --- | --- | --- |
| **OpenSSL** – ECDHE-ECDSA-AES256-SHA384<br>**IANA** – TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 | • ELBSecurityPolicy-TLS-1-2-2017-01<br>• ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08<br>• ELBSecurityPolicy-2015-05<br>• ELBSecurityPolicy-2015-03<br>• ELBSecurityPolicy-2015-02 | c024 |
| **OpenSSL** – ECDHE-RSA-AES256-SHA384<br>**IANA** – TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 | • ELBSecurityPolicy-TLS-1-2-2017-01<br>• ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08<br>• ELBSecurityPolicy-2015-05<br>• ELBSecurityPolicy-2015-03<br>• ELBSecurityPolicy-2015-02 | c028 |
| **OpenSSL** – ECDHE-ECDSA-AES256-SHA<br>**IANA** – TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA | • ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08<br>• ELBSecurityPolicy-2015-05<br>• ELBSecurityPolicy-2015-03<br>• ELBSecurityPolicy-2015-02 | c014 |
| **OpenSSL** – ECDHE-RSA-AES256-SHA<br>**IANA** – TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA | • ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08<br>• ELBSecurityPolicy-2015-05<br>• ELBSecurityPolicy-2015-03<br>• ELBSecurityPolicy-2015-02 | c00a |

| Cipher name | Security policies | Cipher suite |
| --- | --- | --- |
| **OpenSSL** – AES128-GCM-SHA256<br><br>**IANA** – TLS_RSA_WITH_AES_128_GCM_SHA256 | • ELBSecurityPolicy-TLS-1-2-2017-01<br>• ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08<br>• ELBSecurityPolicy-2015-05<br>• ELBSecurityPolicy-2015-03<br>• ELBSecurityPolicy-2015-02 | 9c |
| **OpenSSL** – AES128-SHA256<br><br>**IANA** – TLS_RSA_WITH_AES_128_CBC_SHA256 | • ELBSecurityPolicy-TLS-1-2-2017-01<br>• ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08<br>• ELBSecurityPolicy-2015-05<br>• ELBSecurityPolicy-2015-03<br>• ELBSecurityPolicy-2015-02 | 3c |
| **OpenSSL** – AES128-SHA<br><br>**IANA** – TLS_RSA_WITH_AES_128_CBC_SHA | • ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08<br>• ELBSecurityPolicy-2015-05<br>• ELBSecurityPolicy-2015-03<br>• ELBSecurityPolicy-2015-02 | 2f |
| **OpenSSL** – AES256-GCM-SHA384<br><br>**IANA** – TLS_RSA_WITH_AES_256_GCM_SHA384 | • ELBSecurityPolicy-TLS-1-2-2017-01<br>• ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08<br>• ELBSecurityPolicy-2015-05<br>• ELBSecurityPolicy-2015-03<br>• ELBSecurityPolicy-2015-02 | 9d |

| Cipher name | Security policies | Cipher suite |
|---|---|---|
| **OpenSSL** – AES256-SHA256<br>**IANA** – TLS_RSA_WITH_AES_256_CBC_SHA256 | • ELBSecurityPolicy-TLS-1-2-2017-01<br>• ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08<br>• ELBSecurityPolicy-2015-05<br>• ELBSecurityPolicy-2015-03<br>• ELBSecurityPolicy-2015-02 | 3d |
| **OpenSSL** – AES256-SHA<br>**IANA** – TLS_RSA_WITH_AES_256_CBC_SHA | • ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08<br>• ELBSecurityPolicy-2015-05<br>• ELBSecurityPolicy-2015-03<br>• ELBSecurityPolicy-2015-02 | 35 |
| **OpenSSL** – DHE-RSA-AES128-SHA<br>**IANA** – TLS_DHE_RSA_WITH_AES_128_CBC_SHA | • ELBSecurityPolicy-2015-03<br>• ELBSecurityPolicy-2015-02 | 33 |
| **OpenSSL** – DHE-DSS-AES128-SHA<br>**IANA** – TLS_DHE_DSS_WITH_AES_128_CBC_SHA | • ELBSecurityPolicy-2015-03<br>• ELBSecurityPolicy-2015-02 | 32 |
| **OpenSSL** – DES-CBC3-SHA<br>**IANA** – TLS_RSA_WITH_3DES_EDE_CBC_SHA | • ELBSecurityPolicy-2015-05<br>• ELBSecurityPolicy-2015-03 | 0a |

# Create a Classic Load Balancer with an HTTPS listener

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

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

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

For information about adding an HTTPS listener to an existing load balancer, see Configure an HTTPS listener for your Classic Load Balancer.

**Contents**



## Prerequisites

Before you get started, be sure that you've met the following prerequisites:

- Complete the steps in Recommendations for your VPC.
- Launch the EC2 instances that you plan to register with your load balancer. The security groups for these instances must allow traffic from the load balancer.
- The EC2 instances must respond to the target of the health check with an HTTP status code 200. For more information, see Health checks for the instances for your Classic Load Balancer.
- If you plan to enable the keep-alive option on your EC2 instances, we recommend that you set the keep-alive settings to at least the idle timeout settings of your load balancer. If you want to ensure that the load balancer is responsible for closing the connections to your instance, make sure that the value set on your instance for the keep-alive time is greater than the idle timeout setting on your load balancer. For more information, see Configure the idle connection timeout for your Classic Load Balancer.
- If you create a secure listener, you must deploy an SSL server certificate on your load balancer. The load balancer uses the certificate to terminate and then decrypt requests before sending

them to the instances. If you don't have an SSL certificate, you can create one. For more information, see SSL/TLS certificates for Classic Load Balancers.

## Create an HTTPS load balancer using the console

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

A *listener* is a process that checks for connection requests. It is configured with a protocol and a port for front-end (client to load balancer) connections and a protocol and a port for back-end (load balancer to instance) connections. For information about the ports, protocols, and listener configurations supported by Elastic Load Balancing, see Listeners for your Classic Load Balancer.

**To create your secure Classic Load Balancer using the console**

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. Expand the **Classic Load Balancer** section, then choose **Create**.
6. **Basic configuration**

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

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

   b. For **Scheme**, select **Internet-facing**.
7. **Network mapping**

   a. For **VPC**, select the same VPC that you selected for your instances.

   b. For **Mappings**, first select an Availability Zone, then choose a public subnet from its available subnets. You can only select one subnet per Availability Zone. To improve the availability of your load balancer, select more than one Availability Zone and subnet.

8. **Security groups**

   - For **Security groups**, select an existing security group that is configured to allow the required HTTP traffic on port 80 and HTTPS traffic on port 443.

     If one doesn't exist, you can create a new security group with the necessary rules.

9. **Listeners and routing**

   a. Leave the default listener with the default settings, and select **Add listener**.

   b. For **Listener** on the new listener, select HTTPS as the protocol and the port will update to 443. By default, **Instance** uses the HTTP protocol on port 80.

   c. If back end authentication is needed, change the **Instance** protocol to HTTPS. This will also update the **Instance** port to 443

10. **Secure listener settings**

    When you use HTTPS or SSL for your front-end listener, you must deploy an SSL certificate on your load balancer. The load balancer uses the certificate to terminate the connection and then decrypt requests from clients before sending them to the instances. You must also specify a security policy. Elastic Load Balancing provides security policies that have predefined SSL negotiation configurations, or you can create your own custom security policy. If you configured HTTPS/SSL on the back-end connection, you can enable authentication of your instances.

    a. For **Security policy**, we recommend that you always use the latest predefined security policy, or create a custom policy. See Update the SSL Negotiation Configuration.

    b. For **Default SSL/TLS certificate**, the following options are available:

       - If you created or imported a certificate using AWS Certificate Manager, select **From ACM**, then select the certificate from **Select a certificate**.
       - If you imported a certificate using IAM, select **From IAM**, and then select your certificate from **Select a certificate**.
       - If you have a certificate to import but ACM is not available in your Region, select **Import**, then select **To IAM**. Type the name of the certificate in the **Certificate name** field. In **Certificate private key**, copy and paste the contents of the private key file (PEM-encoded). In **Certificate body**, copy and paste the contents of the public key certificate file (PEM-encoded). In **Certificate Chain**, copy and paste the contents of the

certificate chain file (PEM-encoded), unless you are using a self-signed certificate and it's not important that browsers implicitly accept the certificate.

c. (Optional) If you configured the HTTPS listener to communicate with the instances using an encrypted connection, you can optionally set up authentication of the instances in **Backend authentication certificate**.

> **Note**
>
> If you do not see the **Backend authentication certificate** section, go back to **Listeners and routing** and select HTTPS as the protocol for **Instance**.

i. For **Certificate name**, type the name of the public key certificate.

ii. For **Certificate Body (PEM encoded)**, copy and paste the contents of the certificate. The load balancer communicates with an instance only if its public key matches this key.

iii. To add another certificate, choose **Add new backend certificate**. The limit is five.

11. **Health checks**

a. In the **Ping target** section, select a **Ping Protocol** and **Ping Port**. Your EC2 instances must accept traffic on the specified ping port.

b. For **Ping Port**, ensure the port is `80`.

c. For **Ping Path**, replace the default alue with a single forward slash, (/). This tells Elastic Load Balancing to send health check requests to the default home page for your web server, such as `index.html`.

d. For **Advanced health check settings**, use the default values.

12. **Instances**

a. Select **Add instances**, to bring up the instance selection screen.

b. Under **Available instances**, you can select from the current instances that are available to the load balancer, based on the network settings selected before.

c. After you're satisfied with your selections, select **Confirm** to add the instances to be registered to the load balancer.

13. **Attributes**

- For **Enable cross-zone load balancing**, **Enable connection draining**, and **Timeout (draining interval)** keep the default values.

14. **Load balancer tags (optional)**

    a. The **Key** field is required.

    b. The **Value** field is optional.

    c. To add another tag, select **Add new tag** then input your values into the **Key** field, and optionally the **Value** field.

    d. To remove an existing tag, select **Remove** next to the tag you want to remove.

15. **Summary and creation**

    a. If you need to change any settings, select **Edit** next to the setting needing to be changed.

    b. After you're satisfied with all the settings shown in the summary, select **Create load balancer** to begin creation of your load balancer.

    c. On the final creation page, select **View load balancer** to view your load balancer in the Amazon EC2 console.

16. **Verify**

    a. Select your new load balancer.

    b. On the **Target instances** tab, check the **Health status** column. After at least one of your EC2 instances is **In-service**, you can test your load balancer.

    c. In the **Details** section, copy the load balancers **DNS name**, which would look similar to `my-load-balancer-1234567890.us-east-1.elb.amazonaws.com`.

    d. Paste your load balancers **DNS name** into the address field of a public internet connected web browser. If your load balancer is functioning correctly, you will see the default page of your server.

17. **Delete (optional)**

    a. If you have a CNAME record for your domain that points to your load balancer, point it to a new location and wait for the DNS change to take effect before deleting your load balancer.

    b. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

    c. Select the load balancer.

    d. Choose **Actions**, **Delete load balancer**.

e. When prompted for confirmation, type `confirm` then select **Delete**.

f. After you delete a load balancer, the EC2 instances that were registered with the load balancer continue to run. You will be billed for each partial or full hour that they continue running. When you no longer need an EC2 instance, you can stop or terminate it to prevent incurring additional charges.

## Create an HTTPS load balancer using the AWS CLI

Use the following instructions to create an HTTPS/SSL load balancer using the AWS CLI.

**Tasks**



### Step 1: Configure listeners

A *listener* is a process that checks for connection requests. It is configured with a protocol and a port for front-end (client to load balancer) connections and a protocol and port for back-end (load balancer to instance) connections. For information about the ports, protocols, and listener configurations supported by Elastic Load Balancing, see Listeners for your Classic Load Balancer.

In this example, you configure two listeners for your load balancer by specifying the ports and protocols to use for front-end and back-end connections. The first listener accepts HTTP requests on port 80 and sends the requests to the instances on port 80 using HTTP. The second listener accepts HTTPS requests on port 443 and sends requests to instances using HTTP on port 80.

Because the second listener uses HTTPS for the front-end connection, you must deploy an SSL sever certificate on your load balancer. The load balancer uses the certificate to terminate and then decrypt requests before sending them to the instances.

**To configure listeners for your load balancer**

1. Get the Amazon Resource Name (ARN) of the SSL certificate. For example:

   **ACM**

   ```
   arn:aws:acm:region:123456789012:certificate/12345678-1234-1234-1234-123456789012
   ```

   **IAM**

   ```
   arn:aws:iam::123456789012:server-certificate/my-server-certificate
   ```

2. Use the following create-load-balancer command to configure the load balancer with the two listeners:

   ```
   aws elb create-load-balancer --load-balancer-name my-load-balancer --listeners
    "Protocol=http,LoadBalancerPort=80,InstanceProtocol=http,InstancePort=80"
    "Protocol=https,LoadBalancerPort=443,InstanceProtocol=http,InstancePort=80,SSLCertificateI
    --availability-zones us-west-2a
   ```

   The following is an example response:

   ```
   {
     "DNSName": "my-loadbalancer-012345678.us-west-2.elb.amazonaws.com"
   }
   ```

3. (Optional) Use the following describe-load-balancers command to view the details of your load balancer:

   ```
   aws elb describe-load-balancers --load-balancer-name my-load-balancer
   ```

## Step 2: Configure the SSL security policy

You can select one of the predefined security policies, or you can create your own custom security policy. Otherwise, Elastic Load Balancing configures your load balancer with the default predefined security policy, `ELBSecurityPolicy-2016-08`. For more information, see SSL negotiation configurations for Classic Load Balancers.

**To verify that your load balancer is associated with the default security policy**

Use the following describe-load-balancers command:

```
aws elb describe-load-balancers --load-balancer-name my-loadbalancer
```

The following is an example response. Note that `ELBSecurityPolicy-2016-08` is associated with the load balancer on port 443.

```
{
    "LoadBalancerDescriptions": [
        {
            ...
            "ListenerDescriptions": [
                {
                    "Listener": {
                        "InstancePort": 80,
                        "SSLCertificateId": "ARN",
                        "LoadBalancerPort": 443,
                        "Protocol": "HTTPS",
                        "InstanceProtocol": "HTTP"
                    },
                    "PolicyNames": [
                        "ELBSecurityPolicy-2016-08"
                    ]
                },
                {
                    "Listener": {
                        "InstancePort": 80,
                        "LoadBalancerPort": 80,
                        "Protocol": "HTTP",
                        "InstanceProtocol": "HTTP"
                    },
                    "PolicyNames": []
                }
            ],
            ...
        }
    ]
}
```

If you prefer, you can configure the SSL security policy for your load balancer instead of using the default security policy.

**(Optional) to use a predefined SSL security policy**

1. Use the following describe-load-balancer-policies command to list the names of the predefined security policies:

```
aws elb describe-load-balancer-policies
```

   For information about the configuration for the predefined security policies, see Predefined SSL security policies for Classic Load Balancers.

2. Use the following create-load-balancer-policy command to create an SSL negotiation policy using one of the predefined security policies that you described in the previous step:

```
aws elb create-load-balancer-policy --load-balancer-name my-loadbalancer
--policy-name my-SSLNegotiation-policy --policy-type-name SSLNegotiationPolicyType
--policy-attributes AttributeName=Reference-Security-
Policy,AttributeValue=predefined-policy
```

3. (Optional) Use the following describe-load-balancer-policies command to verify that the policy is created:

```
aws elb describe-load-balancer-policies --load-balancer-name my-loadbalancer --
policy-name my-SSLNegotiation-policy
```

   The response includes the description of the policy.

4. Use the following set-load-balancer-policies-of-listener command to enable the policy on load balancer port 443:

```
aws elb set-load-balancer-policies-of-listener --load-balancer-name my-loadbalancer
 --load-balancer-port 443 --policy-names my-SSLNegotiation-policy
```

> **Note**
>
> The `set-load-balancer-policies-of-listener` command replaces the current set of policies for the specified load balancer port with the specified set of policies. The `--policy-names` list must include all policies to be enabled. If you omit a policy that is currently enabled, it is disabled.

5. (Optional) Use the following describe-load-balancers command to verify that the policy is enabled:

```
aws elb describe-load-balancers --load-balancer-name my-loadbalancer
```

The following is an example response showing that the policy is enabled on port 443.

```
{
    "LoadBalancerDescriptions": [
        {
            ....
            "ListenerDescriptions": [
                {
                    "Listener": {
                        "InstancePort": 80,
                        "SSLCertificateId": "ARN",
                        "LoadBalancerPort": 443,
                        "Protocol": "HTTPS",
                        "InstanceProtocol": "HTTP"
                    },
                    "PolicyNames": [
                        "my-SSLNegotiation-policy"
                    ]
                },
                {
                    "Listener": {
                        "InstancePort": 80,
                        "LoadBalancerPort": 80,
                        "Protocol": "HTTP",
                        "InstanceProtocol": "HTTP"
                    },
                    "PolicyNames": []
                }
            ],
            ...
        }
    ]
}
```

When you create a custom security policy, you must enable at least one protocol and one cipher. The DSA and RSA ciphers are specific to the signing algorithm and are used to create

the SSL certificate. If you already have your SSL certificate, make sure to enable the cipher that was used to create your certificate. The name of your custom policy must not begin with `ELBSecurityPolicy-` or `ELBSample-`, as these prefixes are reserved for the names of the predefined security policies.

**(Optional) to use a custom SSL security policy**

1. Use the create-load-balancer-policy command to create an SSL negotiation policy using a custom security policy. For example:

```
aws elb create-load-balancer-policy --load-balancer-name my-loadbalancer
 --policy-name my-SSLNegotiation-policy --policy-type-name
 SSLNegotiationPolicyType
 --policy-attributes AttributeName=Protocol-TLSv1.2,AttributeValue=true
 AttributeName=Protocol-TLSv1.1,AttributeValue=true
 AttributeName=DHE-RSA-AES256-SHA256,AttributeValue=true
 AttributeName=Server-Defined-Cipher-Order,AttributeValue=true
```

2. (Optional) Use the following describe-load-balancer-policies command to verify that the policy is created:

```
aws elb describe-load-balancer-policies --load-balancer-name my-loadbalancer --
policy-name my-SSLNegotiation-policy
```

   The response includes the description of the policy.

3. Use the following set-load-balancer-policies-of-listener command to enable the policy on load balancer port 443:

```
aws elb set-load-balancer-policies-of-listener --load-balancer-name my-loadbalancer
 --load-balancer-port 443 --policy-names my-SSLNegotiation-policy
```

> **Note**
>
> The `set-load-balancer-policies-of-listener` command replaces the current set of policies for the specified load balancer port with the specified set of policies. The `--policy-names` list must include all policies to be enabled. If you omit a policy that is currently enabled, it is disabled.

4. (Optional) Use the following describe-load-balancers command to verify that the policy is enabled:

```
aws elb describe-load-balancers --load-balancer-name my-loadbalancer
```

The following is an example response showing that the policy is enabled on port 443.

```
{
    "LoadBalancerDescriptions": [
        {
            ....
            "ListenerDescriptions": [
                {
                    "Listener": {
                        "InstancePort": 80,
                        "SSLCertificateId": "ARN",
                        "LoadBalancerPort": 443,
                        "Protocol": "HTTPS",
                        "InstanceProtocol": "HTTP"
                    },
                    "PolicyNames": [
                        "my-SSLNegotiation-policy"
                    ]
                },
                {
                    "Listener": {
                        "InstancePort": 80,
                        "LoadBalancerPort": 80,
                        "Protocol": "HTTP",
                        "InstanceProtocol": "HTTP"
                    },
                    "PolicyNames": []
                }
            ],
            ...
        }
    ]
}
```

## Step 3: Configure back-end instance authentication (optional)

If you set up HTTPS/SSL on the back-end connection, you can optionally set up authentication of your instances.

When you set up back-end instance authentication, you create a public key policy. Next, you use this public key policy to create a back-end instance authentication policy. Finally, you set the back-end instance authentication policy with the instance port for the HTTPS protocol.

The load balancer communicates with an instance only if the public key that the instance presents to the load balancer matches a public key in the authentication policy for your load balancer.

**To configure back-end instance authentication**

1. Use the following command to retrieve the public key:

```
openssl x509 -in your X509 certificate PublicKey -pubkey -noout
```

2. Use the following create-load-balancer-policy command to create a public key policy:

```
aws elb create-load-balancer-policy --load-balancer-name my-loadbalancer --policy-
name my-PublicKey-policy \
--policy-type-name PublicKeyPolicyType --policy-attributes
 AttributeName=PublicKey,AttributeValue=MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w
 0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZ
 WF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIw
 EAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5
 jb20wHhcNMTEwNDI1MjA0NTIxWhcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBh
 MCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBb
 WF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMx
 HzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wgZ8wDQYJKoZIhvcNAQE
 BBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ21uUSfwfEvySWtC2XADZ4nB+BLYgVI
 k60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9TrDHudUZg3qX4waLG5M43q7Wgc/MbQ
 ITxOUSQv7c7ugFFDzQGBzZswY6786m86gpEIbb3OhjZnzcvQAaRHhdlQWIMm2nr
 AgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4nUhVVxYUntneD9+h8Mg9q6q+auN
 KyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0FkbFFBjvSfpJIlJ00zbhNYS5f6Guo
 EDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTbNYiytVbZPQUQ5Yaxu2jXnimvw
 3rrszlaEXAMPLE=
```

> **Note**
>
> To specify a public key value for `--policy-attributes`, remove the first and last lines of the public key (the line containing "`-----BEGIN PUBLIC KEY-----`" and the line containing "`-----END PUBLIC KEY-----`"). The AWS CLI does not accept white space characters in `--policy-attributes`.

3. Use the following create-load-balancer-policy command to create a back-end instance authentication policy using `my-PublicKey-policy`.

```
aws elb create-load-balancer-policy --load-balancer-name my-
loadbalancer --policy-name my-authentication-policy --policy-type-
name BackendServerAuthenticationPolicyType --policy-attributes
 AttributeName=PublicKeyPolicyName,AttributeValue=my-PublicKey-policy
```

   You can optionally use multiple public key policies. The load balancer tries all the keys, one at a time. If the public key presented by an instance matches one of these public keys, the instance is authenticated.

4. Use the following set-load-balancer-policies-for-backend-server command to set `my-authentication-policy` to the instance port for HTTPS. In this example, the instance port is port 443.

```
aws elb set-load-balancer-policies-for-backend-server --load-balancer-name my-
loadbalancer --instance-port 443 --policy-names my-authentication-policy
```

5. (Optional) Use the following describe-load-balancer-policies command to list all the policies for your load balancer:

```
aws elb describe-load-balancer-policies --load-balancer-name my-loadbalancer
```

6. (Optional) Use the following describe-load-balancer-policies command to view details of the policy:

```
aws elb describe-load-balancer-policies --load-balancer-name my-loadbalancer --
policy-names my-authentication-policy
```

## Step 4: Configure health checks (optional)

Elastic Load Balancing regularly checks the health of each registered EC2 instance based on the health checks that you configured. If Elastic Load Balancing finds an unhealthy instance, it stops sending traffic to the instance and routes traffic to the healthy instances. For more information, see Health checks for the instances for your Classic Load Balancer.

When you create your load balancer, Elastic Load Balancing uses default settings for the health checks. If you prefer, you can change the health check configuration for your load balancer instead of using the default settings.

**To configure the health checks for your instances**

Use the following configure-health-check command:

```
aws elb configure-health-check --load-balancer-name my-loadbalancer --health-check
 Target=HTTP:80/ping,Interval=30,UnhealthyThreshold=2,HealthyThreshold=2,Timeout=3
```

The following is an example response:

```
{
    "HealthCheck": {
        "HealthyThreshold": 2,
        "Interval": 30,
        "Target": "HTTP:80/ping",
        "Timeout": 3,
        "UnhealthyThreshold": 2
    }
}
```

## Step 5: Register EC2 instances

After you create your load balancer, you must register your EC2 instances with the load balancer. You can select EC2 instances from a single Availability Zone or multiple Availability Zones within the same Region as the load balancer. For more information, see Registered instances for your Classic Load Balancer.

Use the register-instances-with-load-balancer command as follows:

```
aws elb register-instances-with-load-balancer --load-balancer-name my-loadbalancer --
instances i-4f8cf126 i-0bb7ca62
```

The following is an example response:

```
{
    "Instances": [
        {
            "InstanceId": "i-4f8cf126"
        },
        {
            "InstanceId": "i-0bb7ca62"
        }
    ]
}
```

## Step 6: Verify the instances

Your load balancer is usable as soon as any one of your registered instances is in the `InService` state.

To check the state of your newly registered EC2 instances, use the following describe-instance-health command:

```
aws elb describe-instance-health  --load-balancer-name my-loadbalancer --
instances i-4f8cf126 i-0bb7ca62
```

The following is an example response:

```
{
    "InstanceStates": [
        {
            "InstanceId": "i-4f8cf126",
            "ReasonCode": "N/A",
            "State": "InService",
            "Description": "N/A"
        },
        {
            "InstanceId": "i-0bb7ca62",
            "ReasonCode": "Instance",
            "State": "OutOfService",
            "Description": "Instance registration is still in progress"
        }
    ]
}
```

If the `State` field for an instance is `OutOfService`, it's probably because your instances are still registering. For more information, see Troubleshoot a Classic Load Balancer: Instance registration.

After the state of at least one of your instances is `InService`, you can test your load balancer. To test your load balancer, copy the DNS name of the load balancer and paste it into the address field of an internet-connected web browser. If your load balancer is working, you see the default page of your HTTP server.

### Step 7: Delete your load balancer (optional)

Deleting a the load balancer automatically de-registers its associated EC2 instances. As soon as the load balancer is deleted, you stop incurring charges for that load balancer. However, the EC2 instances continue run and you continue to incur charges.

To delete your load balancer, use the following delete-load-balancer command:

```
aws elb delete-load-balancer --load-balancer-name my-loadbalancer
```

To stop your EC2 instances, use the stop-instances command. To terminate your EC2 instances, use the terminate-instances command.

## Configure an HTTPS listener for your Classic Load Balancer

A *listener* is a process that checks for connection requests. It is configured with a protocol and a port for front-end (client to load balancer) connections and a protocol and a port for back-end (load balancer to instance) connections. For information about the ports, protocols, and listener configurations supported by Elastic Load Balancing, see Listeners for your Classic Load Balancer.

If you have a load balancer with a listener that accepts HTTP requests on port 80, you can add a listener that accepts HTTPS requests on port 443. If you specify that the HTTPS listener sends requests to the instances on port 80, the load balancer terminates the SSL requests and communication from the load balancer to the instances is not encrypted. If the HTTPS listener sends requests to the instances on port 443, communication from the load balancer to the instances is encrypted.

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

For information about creating a new HTTPS listener, see Create a Classic Load Balancer with an HTTPS listener.

**Contents**



## Prerequisites

To enable HTTPS support for an HTTPS listener, you must deploy an SSL server certificate on your load balancer. The load balancer uses the certificate to terminate and then decrypt requests before sending them to the instances. If you do not have an SSL certificate, you can create one. For more information, see SSL/TLS certificates for Classic Load Balancers.

## Add an HTTPS listener using the console

You can add an HTTPS listener to an existing load balancer.

**To add an HTTPS listener to your load balancer using the console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Load Balancing**, choose **Load Balancers**.
3. Choose the name of the load balancer to open its detail page.
4. On the **Listeners** tab, choose **Manage listeners**.
5. On the **Manage listeners** page, in the **Listeners** section, choose **Add listener**.
6. For **Listener protocol**, select **HTTPS** .

   > **Important**
   >
   > By default, the **Instance protocol** is HTTP. If you want to set up back-end instance authentication, change **Instance protocol** to HTTPS .

7. For **Security policy**, we recommend that you use the latest predefined security policy. If you need to use a different predefined security policy or create a custom policy, see Update the SSL Negotiation Configuration.

8. For **Default SSL cert**, choose **Edit**, and then do one of the following:

   - If you created or imported a certificate using AWS Certificate Manager, choose **From ACM**, select the certificate from the list, and then choose **Save changes**.

     > **Note**
     >
     > This option is available only in Regions that support AWS Certificate Manager.

   - If you imported a certificate using IAM, choose **From IAM**, select the certificate from from the list, and then choose **Save changes**.
   - If you have an SSL certificate to import to ACM, select **Import** and **To ACM**. In **Certificate private key**, copy and paste the contents of the PEM-encoded private key file. In **Certificate body**, copy and paste the contents of the PEM-encoded public key certificate file. In **Certificate chain - optional**, copy and paste the contents of the PEM-encoded certificate chain file, unless you are using a self-signed certificate and it's not important that browsers implicitly accept the certificate.
   - If you have an SSL certificate to import but ACM is not supported in this Region, select **Import** and **To IAM**. In **Certificate name** type the name of the certificate. In **Certificate private key**, copy and paste the contents of the PEM-encoded private key file. In **Certificate body**, copy and paste the contents of the PEM-encoded public key certificate file. In **Certificate chain - optional**, copy and paste the contents of the PEM-encoded certificate chain file, unless you are using a self-signed certificate and it's not important that browsers implicitly accept the certificate.
   - Choose **Save changes**.

9. For **Cookie stickiness**, the default is **Disabled**. To change this choose **Edit**. If choosing **Generated by load balancer**, an **Expiration period** must be specified. If choosing **Generated by application**, a **Cookie name** must be specified. After making your selection choose **Save changes**.
10. (Optional) Choose **Add listener** to add additional listeners.
11. Choose **Save changes** to add the listeners you just configured.
12. (Optional) To set up back-end instance authentication for an existing load balancer, you must use the AWS CLI or an API, as this task is not supported using the console. For more information, see Configure Back-end Instance Authentication.

# Add an HTTPS listener using the AWS CLI

You can add an HTTPS listener to an existing load balancer.

**To add an HTTPS listener to your load balancer using the AWS CLI**

1. Get the Amazon Resource Name (ARN) of the SSL certificate. For example:

   **ACM**

   ```
   arn:aws:acm:region:123456789012:certificate/12345678-1234-1234-1234-123456789012
   ```

   **IAM**

   ```
   arn:aws:iam::123456789012:server-certificate/my-server-certificate
   ```

2. Use the following create-load-balancer-listeners command to add a listener to your load balancer that accepts HTTPS requests on port 443 and sends the requests to the instances on port 80 using HTTP:

   ```
   aws elb create-load-balancer-listeners --load-balancer-name my-load-balancer --
   listeners
    Protocol=HTTPS,LoadBalancerPort=443,InstanceProtocol=HTTP,InstancePort=80,SSLCertificateId
   ```

   If you want to set up back-end instance authentication, use the following command to add a listener that accepts HTTPS requests on port 443 and sends the requests to the instances on port 443 using HTTPS:

   ```
   aws elb create-load-balancer-listeners --load-balancer-name my-load-balancer --
   listeners
    Protocol=HTTPS,LoadBalancerPort=443,InstanceProtocol=HTTPS,InstancePort=443,SSLCertificate
   ```

3. (Optional) You can use the following describe-load-balancers command to view the updated details of your load balancer:

   ```
   aws elb describe-load-balancers --load-balancer-name my-load-balancer
   ```

   The following is an example response:

```
{
    "LoadBalancerDescriptions": [
        {
            ...
            "ListenerDescriptions": [
                {
                    "Listener": {
                        "InstancePort": 80,
                        "SSLCertificateId": "ARN",
                        "LoadBalancerPort": 443,
                        "Protocol": "HTTPS",
                        "InstanceProtocol": "HTTP"
                    },
                    "PolicyNames": [
                        "ELBSecurityPolicy-2016-08"
                    ]
                },
                {
                    "Listener": {
                        "InstancePort": 80,
                        "LoadBalancerPort": 80,
                        "Protocol": "HTTP",
                        "InstanceProtocol": "HTTP"
                    },
                    "PolicyNames": []
                }
            ],
            ...
        }
    ]
}
```

4. (Optional) Your HTTPS listener was created using the default security policy. If you want to specify a different predefined security policy or a custom security policy, use the create-load-balancer-policy and set-load-balancer-policies-of-listener commands. For more information, see Update the SSL negotiation configuration using the AWS CLI.

5. (Optional) To set up back-end instance authentication, use the set-load-balancer-policies-for-backend-server command. For more information, see Configure Back-end Instance Authentication.

# Replace the SSL certificate for your Classic Load Balancer

If you have an HTTPS listener, you deployed an SSL server certificate on your load balancer when you created the listener. Each certificate comes with a validity period. You must ensure that you renew or replace the certificate before its validity period ends.

Certificates provided by AWS Certificate Manager and deployed on your load balancer can be renewed automatically. ACM attempts to renew certificates before they expire. For more information, see Managed renewal in the *AWS Certificate Manager User Guide*. If you imported a certificate into ACM, you must monitor the expiration date of the certificate and renew it before it expires. For more information, see Importing certificates in the *AWS Certificate Manager User Guide*. After a certificate that is deployed on a load balancer is renewed, new requests use the renewed certificate.

To replace a certificate, you must first create a new certificate by following the same steps that you used when you created the current certificate. Then, you can replace the certificate. After a certificate that is deployed on a load balancer is replaced, new requests use the new certificate.

Note that renewing or replacing a certificate does not affect requests that were already received by a load balancer node and are pending routing to a healthy target.

**Contents**



## Replace the SSL certificate using the console

You can replace the certificate deployed on your load balancer with a certificate provided by ACM or a certificate uploaded to IAM.

**To replace the SSL certificate for an HTTPS load balancer using the console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Load Balancing**, choose **Load Balancers**.
3. Choose the name of the load balancer to open its detail page.
4. On the **Listeners** tab, choose **Manage listeners**.

5. On the **Manage listeners** page, locate the listener to be updated, choose **Edit** under **Default SSL cert** and do one of the following:

   - If you created or imported a certificate using AWS Certificate Manager, choose **From ACM**, select the certificate from the list, and then choose **Save changes**.

     > **Note**
     >
     > This option is available only in Regions that support AWS Certificate Manager.

   - If you imported a certificate using IAM, choose **From IAM**, select the certificate from from the list, and then choose **Save changes**.
   - If you have an SSL certificate to import to ACM, select **Import** and **To ACM**. In **Certificate private key**, copy and paste the contents of the PEM-encoded private key file. In **Certificate body**, copy and paste the contents of the PEM-encoded public key certificate file. In **Certificate chain - optional**, copy and paste the contents of the PEM-encoded certificate chain file, unless you are using a self-signed certificate and it's not important that browsers implicitly accept the certificate.
   - If you have an SSL certificate to import but ACM is not supported in this Region, select **Import** and **To IAM**. In **Certificate name** type the name of the certificate. In **Certificate private key**, copy and paste the contents of the PEM-encoded private key file. In **Certificate body**, copy and paste the contents of the PEM-encoded public key certificate file. In **Certificate chain - optional**, copy and paste the contents of the PEM-encoded certificate chain file, unless you are using a self-signed certificate and it's not important that browsers implicitly accept the certificate.
   - Choose **Save changes**.

## Replace the SSL certificate using the AWS CLI

You can replace the certificate deployed on your load balancer with a certificate provided by ACM or a certificate uploaded to IAM.

**To replace an SSL certificate with a certificate provided by ACM**

1. Use the following request-certificate command to request a new certificate:

```
aws acm request-certificate --domain-name www.example.com
```

2. Use the following set-load-balancer-listener-ssl-certificate command to set the certificate:

```
aws elb set-load-balancer-listener-ssl-certificate --load-balancer-
name my-load-balancer --load-balancer-port 443 --ssl-certificate-id
 arn:aws:acm:region:123456789012:certificate/12345678-1234-1234-1234-123456789012
```

**To replace an SSL certificate with a certificate uploaded to IAM**

1. If you have an SSL certificate but have not uploaded it, see Uploading a server certificate in the *IAM User Guide*.
2. Use the following get-server-certificate command to get the ARN of the certificate:

```
aws iam get-server-certificate --server-certificate-name my-new-certificate
```

3. Use the following set-load-balancer-listener-ssl-certificate command to set the certificate:

```
aws elb set-load-balancer-listener-ssl-certificate --load-balancer-
name my-load-balancer --load-balancer-port 443 --ssl-certificate-id
 arn:aws:iam::123456789012:server-certificate/my-new-certificate
```

# Update the SSL negotiation configuration of your Classic Load Balancer

Elastic Load Balancing provides security policies that have predefined SSL negotiation configurations to use to negotiate SSL connections between clients and your load balancer. If you are using the HTTPS/SSL protocol for your listener, you can use one of the predefined security policies, or use your own custom security policy.

For more information about the security policies, see SSL negotiation configurations for Classic Load Balancers. For information about the configurations of the security policies provided by Elastic Load Balancing, see Predefined SSL security policies for Classic Load Balancers.

If you create an HTTPS/SSL listener without associating a security policy, Elastic Load Balancing associates the default predefined security policy, `ELBSecurityPolicy-2016-08`, with your load balancer.

If you prefer, you can create a custom configuration. We strongly recommend that you test your security policy before you upgrade your load balancer configuration.

The following examples show you how to update the SSL negotiation configuration for an HTTPS/SSL listener. Note that the change does not affect requests that were received by a load balancer node and are pending routing to a healthy instance, but the updated configuration will be used with new requests that are received.

**Contents**



## Update the SSL negotiation configuration using the console

By default, Elastic Load Balancing associates the latest predefined policy with your load balancer. When a new predefined policy is added, we recommend that you update your load balancer to use the new predefined policy. Alternatively, you can select a different predefined security policy or create a custom policy.

**To update SSL negotiation configuration for an HTTPS/SSL load balancer using the console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Load Balancing**, choose **Load Balancers**.
3. Choose the name of the load balancer to open its detail page.
4. On the **Listeners** tab, choose **Manage listeners**.
5. On the **Manage listeners** page, locate the listener to be updated, choose **Edit** under **Security policy** select a security policy using one of the following options:
   - Keep the default policy, **ELBSecurityPolicy-2016-08**, and then choose **Save changes**.
   - Select a predefined policy other than the default, and then choose **Save changes**.
   - Select **Custom** and enable at least one protocol and one cipher as follows:
     a. For **SSL Protocols**, select one or more protocols to enable.
     b. For **SSL Options**, select **Server Order Preference** to use the order listed in the Predefined SSL security policies for Classic Load Balancers for SSL negotiation.

c. For **SSL Ciphers**, select one or more ciphers to enable. If you already have an SSL certificate, you must enable the cipher that was used to create the certificate, because DSA and RSA ciphers are specific to the signing algorithm.

d. Choose **Save changes**.

## Update the SSL negotiation configuration using the AWS CLI

You can use the default predefined security policy, `ELBSecurityPolicy-2016-08`, a different predefined security policy, or a custom security policy.

**To use a predefined SSL security policy**

1. Use the following describe-load-balancer-policies command to list the predefined security policies provided by Elastic Load Balancing. The syntax that you use depends on the operating system and shell that you are using.

   **Linux**

   ```
   aws elb describe-load-balancer-policies --query 'PolicyDescriptions[?
   PolicyTypeName==`SSLNegotiationPolicyType`].{PolicyName:PolicyName}' --output table
   ```

   **Windows**

   ```
   aws elb describe-load-balancer-policies --query "PolicyDescriptions[?
   PolicyTypeName==`SSLNegotiationPolicyType`].{PolicyName:PolicyName}" --output table
   ```

   The following is example output:

   ```
   ----------------------------------------
   |      DescribeLoadBalancerPolicies      |
   +----------------------------------------+
   |               PolicyName               |
   +----------------------------------------+
   |  ELBSecurityPolicy-2016-08             |
   |  ELBSecurityPolicy-TLS-1-2-2017-01     |
   |  ELBSecurityPolicy-TLS-1-1-2017-01     |
   |  ELBSecurityPolicy-2015-05             |
   |  ELBSecurityPolicy-2015-03             |
   |  ELBSecurityPolicy-2015-02             |
   ```

```
|  ELBSecurityPolicy-2014-10             |
|  ELBSecurityPolicy-2014-01             |
|  ELBSecurityPolicy-2011-08             |
|  ELBSample-ELBDefaultCipherPolicy      |
|  ELBSample-OpenSSLDefaultCipherPolicy  |
+----------------------------------------+
```

To determine which ciphers are enabled for a policy, use the following command:

```
aws elb describe-load-balancer-policies --policy-names ELBSecurityPolicy-2016-08 --
output table
```

For information about the configuration for the predefined security policies, see Predefined SSL security policies for Classic Load Balancers.

2. Use the create-load-balancer-policy command to create an SSL negotiation policy using one of the predefined security policies that you described in the previous step. For example, the following command uses the default predefined security policy:

```
aws elb create-load-balancer-policy --load-balancer-name my-loadbalancer
--policy-name my-SSLNegotiation-policy  --policy-type-name SSLNegotiationPolicyType
--policy-attributes AttributeName=Reference-Security-
Policy,AttributeValue=ELBSecurityPolicy-2016-08
```

If you exceed the limit on the number of policies for the load balancer, use the delete-load-balancer-policy command to delete any unused policies.

3. (Optional) Use the following describe-load-balancer-policies command to verify that the policy is created:

```
aws elb describe-load-balancer-policies --load-balancer-name my-loadbalancer --
policy-name my-SSLNegotiation-policy
```

The response includes the description of the policy.

4. Use the following set-load-balancer-policies-of-listener command to enable the policy on load balancer port 443:

```
aws elb set-load-balancer-policies-of-listener --load-balancer-name my-loadbalancer
 --load-balancer-port 443 --policy-names my-SSLNegotiation-policy
```

> **Note**
>
> The `set-load-balancer-policies-of-listener` command replaces the current set of policies for the specified load balancer port with the the specified set of policies. The `--policy-names` list must include all policies to be enabled. If you omit a policy that is currently enabled, it is disabled.

5. (Optional) Use the following describe-load-balancers command to verify that the new policy is enabled for the load balancer port:

```
aws elb describe-load-balancers --load-balancer-name my-loadbalancer
```

The response shows that the policy is enabled on port 443.

```
...
  {
      "Listener": {
          "InstancePort": 443,
          "SSLCertificateId": "ARN",
          "LoadBalancerPort": 443,
          "Protocol": "HTTPS",
          "InstanceProtocol": "HTTPS"
      },
      "PolicyNames": [
          "my-SSLNegotiation-policy"
      ]
  }
...
```

When you create a custom security policy, you must enable at least one protocol and one cipher. The DSA and RSA ciphers are specific to the signing algorithm and are used to create the SSL certificate. If you already have an SSL certificate, be sure to enable the cipher that was used to create the certificate. The name of your custom policy must not begin with `ELBSecurityPolicy-` or `ELBSample-`, as these prefixes are reserved for the names of the predefined security policies.

**To use a custom SSL security policy**

1. Use the create-load-balancer-policy command to create an SSL negotiation policy using a custom security policy. For example:

```
aws elb create-load-balancer-policy --load-balancer-name my-loadbalancer
 --policy-name my-SSLNegotiation-policy --policy-type-name
 SSLNegotiationPolicyType
 --policy-attributes AttributeName=Protocol-TLSv1.2,AttributeValue=true
 AttributeName=Protocol-TLSv1.1,AttributeValue=true
 AttributeName=DHE-RSA-AES256-SHA256,AttributeValue=true
 AttributeName=Server-Defined-Cipher-Order,AttributeValue=true
```

   If you exceed the limit on the number of policies for the load balancer, use the delete-load-balancer-policy command to delete any unused policies.

2. (Optional) Use the following describe-load-balancer-policies command to verify that the policy is created:

```
aws elb describe-load-balancer-policies --load-balancer-name my-loadbalancer --
policy-name my-SSLNegotiation-policy
```

   The response includes the description of the policy.

3. Use the following set-load-balancer-policies-of-listener command to enable the policy on load balancer port 443:

```
aws elb set-load-balancer-policies-of-listener --load-balancer-name my-loadbalancer
 --load-balancer-port 443 --policy-names my-SSLNegotiation-policy
```

   **Note**

   The `set-load-balancer-policies-of-listener` command replaces the current set of policies for the specified load balancer port with the the specified set of policies. The `--policy-names` list must include all policies to be enabled. If you omit a policy that is currently enabled, it is disabled.

4. (Optional) Use the following describe-load-balancers command to verify that the new policy is enabled for the load balancer port:

```
aws elb describe-load-balancers --load-balancer-name my-loadbalancer
```

The response shows that the policy is enabled on port 443.

```
...
  {
      "Listener": {
          "InstancePort": 443,
          "SSLCertificateId": "ARN",
          "LoadBalancerPort": 443,
          "Protocol": "HTTPS",
          "InstanceProtocol": "HTTPS"
      },
      "PolicyNames": [
          "my-SSLNegotiation-policy"
      ]
  }
...
```

# Registered instances for your Classic Load Balancer

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

**Contents**



## Best practices for your instances

- You must ensure that the load balancer can communicate with your instances on both the listener port and the health check port. For more information, see Configure security groups for your Classic Load Balancer. The security group for your instances must allow traffic in both directions on both ports for each subnet for your load balancer.
- Install a web server, such as Apache or Internet Information Services (IIS), on all instances that you plan to register with your load balancer.
- For HTTP and HTTPS listeners, we recommend that you enable the keep-alive option in your EC2 instances, which enables the load balancer to re-use the connections to your instances for multiple client requests. This reduces the load on your web server and improves the throughput of the load balancer. The keep-alive timeout should be at least 60 seconds to ensure that the load balancer is responsible for closing the connection to your instance.
- Elastic Load Balancing supports Path Maximum Transmission Unit (MTU) Discovery. To ensure that Path MTU Discovery can function correctly, you must ensure that the security group for your instance allows ICMP fragmentation required (type 3, code 4) messages. For more information, see Path MTU Discovery in the *Amazon EC2 User Guide*.

# Recommendations for your VPC

**Virtual private cloud (VPC)**

Unless you created your AWS account before 2014, you have a default VPC in each Region. You can use a default VPC for your load balancer, if you have one, or you can create a new VPC. For more information, see the *Amazon VPC User Guide*.

**Subnets for your load balancer**

To ensure that your load balancer can scale properly, verify that each subnet for your load balancer has a CIDR block with at least a `/27` bitmask (for example, `10.0.0.0/27`) and has at least 8 free IP addresses. Your load balancer uses these IP addresses to establish connections with the instances, and to scale out when necessary. If there are insufficient IP addresses, the load balancer might be unable to scale, causing 503 errors due to insufficient capacity.

Create a subnet in each Availability Zone where you want to launch instances. Depending on your application, you can launch your instances in public subnets, private subnets, or a combination of public and private subnets. A public subnet has a route to an internet gateway. Note that default VPCs have one public subnet per Availability Zone by default.

When you create a load balancer, you must add one or more public subnets to the load balancer. If your instances are in private subnets, create public subnets in the same Availability Zones as the subnets with your instances; you will add these public subnets to the load balancer.

**Network ACLs**

The network ACLs for your VPC must allow traffic in both directions on the listener port and the health check port. For more information, see Network ACLs for the instances for your Classic Load Balancer.

# Register instances with your Classic Load Balancer

Registering an EC2 instance adds it to your load balancer. The load balancer continuously monitors the health of registered instances in its enabled Availability Zones, and routes requests to the instances that are healthy. If demand on your instances increases, you can register additional instances with the load balancer to handle the demand.

Deregistering an EC2 instance removes it from your load balancer. The load balancer stops routing requests to an instance as soon as it is deregistered. If demand decreases, or you need to

service your instances, you can deregister instances from the load balancer. An instance that is deregistered remains running, but no longer receives traffic from the load balancer, and you can register it with the load balancer again when you are ready.

When you deregister an instance, Elastic Load Balancing waits until in-flight requests have completed if connection draining is enabled. For more information, see Configure connection draining for your Classic Load Balancer.

If your load balancer is attached to an Auto Scaling group, instances in the group are automatically registered with the load balancer. If you detach a load balancer from your Auto Scaling group, the instances in the group are deregistered.

Elastic Load Balancing registers your EC2 instance with your load balancer using its IP address.

[EC2-VPC] When you register an instance with an elastic network interface (ENI) attached, the load balancer routes requests to the primary IP address of the primary interface (eth0) of the instance.

**Contents**



## Register an instance

When you are ready, register your instance with your load balancer. If the instance is an in Availability Zone that is enabled for the load balancer, the instance is ready to receive traffic from the load balancer as soon as it passes the required number of health checks.

**To register your instances using the console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Load Balancing**, choose **Load Balancers**.
3. Choose the name of the load balancer to open its detail page.
4. On the **Target instances** tab, select **Manage instances**.
5. On the **Manage instances** page, within the **Available instances** table, select the instances to register with your load balancer.

6. Ensure the instances needing to be registered are populated within the **Review selected instances** table.
7. Choose **Save changes**.

### To register your instances using the AWS CLI

Use the following register-instances-with-load-balancer command:

```
aws elb register-instances-with-load-balancer --load-balancer-name my-loadbalancer --instances i-4e05f721
```

The following is an example response that lists the instances registered with the load balancer:

```
{
    "Instances": [
        {
            "InstanceId": "i-315b7e51"
        },
        {
            "InstanceId": "i-4e05f721"
        }
    ]
}
```

## View the instances registered with a load balancer

Use the following describe-load-balancers command to list the instances registered with the specified load balancer:

```
aws elb describe-load-balancers --load-balancer-names my-load-balancer --output text --query "LoadBalancerDescriptions[*].Instances[*].InstanceId"
```

The following is example output:

```
i-e905622e
i-315b7e51
i-4e05f721
```

## Determine the load balancer for a registered instance

Use the following describe-load-balancers command to get the name of the load balancer to which the specified instance is registered:

```
aws elb describe-load-balancers --output text --query "LoadBalancerDescriptions[?
Instances[?InstanceId=='i-e905622e']].[LoadBalancerName]"
```

The following is example output:

```
my-load-balancer
```

## Deregister an instance

You can deregister an instance from your load balancer if you no longer need the capacity or if you need to service the instance.

If your load balancer is attached to an Auto Scaling group, detaching the instance from the group also deregisters it from the load balancer. For more information, see Detach EC2 instances from your Auto Scaling group in the *Amazon EC2 Auto Scaling User Guide*.

**To deregister your instances using the console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Load Balancing**, choose **Load Balancers**.
3. Choose the name of the load balancer to open its detail page.
4. On the **Target instances** tab, select **Manage instances**.
5. On the **Manage instances** page, within the **Available instances** table, deselect the instances to deregister from your load balancer.
6. Ensure the instances needing to be deregistered are not populated within the **Review selected instances** table.
7. Choose **Save changes**.

**To deregister your instances using the AWS CLI**

Use the following deregister-instances-from-load-balancer command:

```
aws elb deregister-instances-from-load-balancer --load-balancer-name my-loadbalancer --
instances i-4e05f721
```

The following is an example response that lists the remaining instances registered with the load balancer:

```
{
    "Instances": [
        {
            "InstanceId": "i-315b7e51"
        }
    ]
}
```

# Health checks for the instances for your Classic Load Balancer

Your Classic Load Balancer periodically sends requests to its registered instances to test their status. These tests are called *health checks*. The status of the instances that are healthy at the time of the health check is `InService`. The status of any instances that are unhealthy at the time of the health check is `OutOfService`. The load balancer performs health checks on all registered instances, whether the instance is in a healthy state or an unhealthy state.

The load balancer routes requests only to the healthy instances. When the load balancer determines that an instance is unhealthy, it stops routing requests to that instance. The load balancer resumes routing requests to the instance when it has been restored to a healthy state.

The load balancer checks the health of the registered instances using either the default health check configuration provided by Elastic Load Balancing or a health check configuration that you configure.

If you have associated your Auto Scaling group with a Classic Load Balancer, you can use the load balancer health check to determine the health state of instances in your Auto Scaling group. By default an Auto Scaling group periodically determines the health state of each instance. For more information, see Add Elastic Load Balancing health checks to your Auto Scaling group in the *Amazon EC2 Auto Scaling User Guide*.

**Contents**

- Check the health of your instances
- Troubleshoot health checks

## Health check configuration

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

| Field | Description |
| --- | --- |
| Protocol | The protocol to use to connect with the instance.<br><br>Valid values: TCP, HTTP, HTTPS, and SSL<br><br>Console default: HTTP<br><br>CLI/API default: TCP |
| Port | The port to use to connect with the instance, as a `protocol:port` pair. If the load balancer fails to connect with the instance at the specified port within the configured response timeout period, the instance is considered unhealthy.<br><br>Protocols: TCP, HTTP, HTTPS, and SSL<br><br>Port range: 1 to 65535<br><br>Console default: `HTTP:80`<br><br>CLI/API default: `TCP:80` |
| Path | The destination for the HTTP or HTTPS request.<br><br>An HTTP or HTTPS GET request is issued to the instance on the port and the path. If the load balancer receives |

| Field | Description |
| --- | --- |
| | any response other than "200 OK" within the response timeout period, the instance is considered unhealthy. If the response includes a body, your application must either set the Content-Length header to a value greater than or equal to zero, or specify Transfer-Encoding with a value set to 'chunked'.<br><br>Default: `/index.html` |
| Response Timeout | The amount of time to wait when receiving a response from the health check, in seconds.<br><br>Valid values: 2 to 60<br><br>Default: 5 |
| HealthCheck Interval | The amount of time between health checks of an individual instance, in seconds.<br><br>Valid values: 5 to 300<br><br>Default: 30 |
| Unhealthy Threshold | The number of consecutive failed health checks that must occur before declaring an EC2 instance unhealthy.<br><br>Valid values: 2 to 10<br><br>Default: 2 |
| Healthy Threshold | The number of consecutive successful health checks that must occur before declaring an EC2 instance healthy.<br><br>Valid values: 2 to 10<br><br>Default: 10 |

The load balancer sends a health check request to each registered instance every `Interval` seconds, using the specified port, protocol, and path. Each health check request is independent and lasts the entire interval. The time it takes for the instance to respond does not affect the interval for the next health check. If the health checks exceed **UnhealthyThresholdCount** consecutive failures, the load balancer takes the instance out of service. When the health checks exceed **HealthyThresholdCount** consecutive successes, the load balancer puts the instance back in service.

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

## Update the health check configuration

You can update the health check configuration for your load balancer at any time.

**To update the health check configuration for your load balancer using the console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Load Balancing**, choose **Load Balancers**.
3. Choose the name of the load balancer to open its detail page.
4. On the **Health checks** tab, choose **Edit**.
5. On the **Edit health check settings** page, under **Health checks**, update the configuration as needed.
6. After you're satisfied with your selections, choose **Save changes**.

**To update the health check configuration for your load balancer using the AWS CLI**

Use the following configure-health-check command:

```
aws elb configure-health-check --load-balancer-name my-load-balancer --health-check
 Target=HTTP:80/path,Interval=30,UnhealthyThreshold=2,HealthyThreshold=2,Timeout=3
```

## Check the health of your instances

You can check the health status of your registered instances.

**To check the health status of your instances using the console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

2. On the navigation pane, under **Load Balancing**, choose **Load Balancers**.
3. Choose the name of the load balancer to open its detail page.
4. In the **Details** section, **Status** indicates how many instances are in service.
5. On the **Target instances** tab, within the **Target instances** table, the **Health status** column indicates the specific status of each registered instance.

**To check the health status of your instances using the AWS CLI**

Use the following describe-instance-health command:

```
aws elb describe-instance-health --load-balancer-name my-load-balancer
```

## Troubleshoot health checks

Your registered instances can fail the load balancer health check for several reasons. The most common reasons for failing a health check are where EC2 instances close connections to your load balancer or where the response from the EC2 instances times out. For information about potential causes and steps you can take to resolve failed health check issues, see Troubleshoot a Classic Load Balancer: Health checks.

# Security groups for the instances for your Classic Load Balancer

A *security group* acts as a firewall that controls the traffic allowed to and from one or more instances. When you launch an EC2 instance, you can associate one or more security groups with the instance. For each security group, you add one or more rules to allow traffic. You can modify the rules for a security group at any time; the new rules are automatically applied to all instances associated with the security group. For more information, see Amazon EC2 security groups in the *Amazon EC2 User Guide*.

The security groups for your instances must allow them to communicate with the load balancer. The following table shows the recommended inbound rules.

| Source | Protocol | Port Range | Comment |
| --- | --- | --- | --- |
| *load balancer security group* | TCP | *instance listener* | Allow traffic from the load balancer on the instance listener port |

| Source | Protocol | Port Range | Comment |
| --- | --- | --- | --- |
| *load balancer security group* | TCP | *health check* | Allow traffic from the load balancer on the health check port |

We also recommend that you allow inbound ICMP traffic to support Path MTU Discovery. For more information, see Path MTU Discovery in the *Amazon EC2 User Guide*.

## Network ACLs for the instances for your Classic Load Balancer

A *network access control list* (ACL) allows or denies specific inbound or outbound traffic at the subnet level. You can use the default network ACL for your VPC, or you can create a custom network ACL for your VPC with rules that are similar to the rules for your security groups in order to add an additional layer of security to your VPC.

The default network access control list (ACL) for the VPC allows all inbound and outbound traffic. If you create custom network ACLs, you must add rules that allow the load balancer and instances to communicate.

The recommended rules for the subnet for your instances depend on whether the subnet is private or public. The following rules are for a private subnet. If your instances are in a public subnet, change the source and destination from the CIDR of the VPC to `0.0.0.0/0`.

The following are the recommended inbound rules.

| Source | Protocol | Port Range | Comment |
| --- | --- | --- | --- |
| *VPC CIDR* | TCP | *instance listener* | Allow inbound traffic from the VPC CIDR on the instance listener port |
| *VPC CIDR* | TCP | *health check* | Allow inbound traffic from the VPC CIDR on the health check port |

The following are the recommended outbound rules.

| Destination | Protocol | Port Range | Comment |
| --- | --- | --- | --- |
| *VPC CIDR* | TCP | 1024-65535 | Allow outbound traffic to the VPC CIDR on the ephemeral ports |

# Monitor your Classic Load Balancer

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

**CloudWatch metrics**

Elastic Load Balancing publishes data points to Amazon CloudWatch about your load balancers and back-end instances. CloudWatch enables you to retrieve statistics about those data points as an ordered set of time-series data, known as *metrics*. You can use these metrics to verify that your system is performing as expected. For more information, see CloudWatch metrics for your Classic Load Balancer.

**Elastic Load Balancing access logs**

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

**CloudTrail logs**

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

# CloudWatch metrics for your Classic Load Balancer

Elastic Load Balancing publishes data points to Amazon CloudWatch for your load balancers and your back-end instances. CloudWatch enables you to retrieve statistics about those data points as an ordered set of time-series data, known as *metrics*. Think of a metric as a variable to monitor, and the data points as the values of that variable over time. For example, you can monitor the total number of healthy EC2 instances for a load balancer over a specified time period. Each data point has an associated time stamp and an optional unit of measurement.

You can use metrics to verify that your system is performing as expected. For example, you can create a CloudWatch alarm to monitor a specified metric and initiate an action (such as sending a notification to an email address) if the metric goes outside what you consider an acceptable range.

Elastic Load Balancing reports metrics to CloudWatch only when requests are flowing through the load balancer. If there are requests flowing through the load balancer, Elastic Load Balancing measures and sends its metrics in 60-second intervals. If there are no requests flowing through the load balancer or no data for a metric, the metric is not reported.

For more information about Amazon CloudWatch, see the *Amazon CloudWatch User Guide*.

**Contents**



## Classic Load Balancer metrics

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

| Metric | Description |
| --- | --- |
| `BackendConnectionErrors` | The number of connections that were not successfully establ ished between the load balancer and the registered instances . Because the load balancer retries the connection when there are errors, this count can exceed the request rate. Note that this count also includes any connection errors related to health checks.<br><br>**Reporting criteria**: There is a nonzero value<br><br>**Statistics**: The most useful statistic is `Sum`. Note that `Average`, `Minimum`, and `Maximum` are reported per load balancer node and are not typically useful. However, the difference between the minimum and maximum (or peak to average or average to |

| Metric | Description |
| --- | --- |
| | trough) might be useful to determine whether a load balancer node is an outlier.<br><br>**Example**: Suppose that your load balancer has 2 instances in us-west-2a and 2 instances in us-west-2b, and that attempts to connect to 1 instance in us-west-2a result in back-end connection errors. The sum for us-west-2a includes these connection errors, while the sum for us-west-2b does not include them. Therefore, the sum for the load balancer equals the sum for us-west-2a. |
| `DesyncMitigationMode_NonCompliant_Request_Count` | [HTTP listener] The number of requests that do not comply with RFC 7230.<br><br>**Reporting criteria**: There is a nonzero value<br><br>**Statistics**: The most useful statistic is `Sum`. |

| Metric | Description |
| --- | --- |
| HealthyHostCount | The number of healthy instances registered with your load balancer. A newly registered instance is considered healthy after it passes the first health check. If cross-zone load balancing is enabled, the number of healthy instances for the LoadBalancerName dimension is calculated across all Availability Zones. Otherwise, it is calculated per Availability Zone.<br><br>**Reporting criteria**: There are registered instances<br><br>**Statistics**: The most useful statistics are Average and Maximum. These statistics are determined by the load balancer nodes. Note that some load balancer nodes might determine that an instance is unhealthy for a brief period while other nodes determine that it is healthy.<br><br>**Example**: Suppose that your load balancer has 2 instances in us-west-2a and 2 instances in us-west-2b, us-west-2a has 1 unhealthy instance, and us-west-2b has no unhealthy instances. With the AvailabilityZone dimension, there is an average of 1 healthy and 1 unhealthy instance in us-west-2a, and an average of 2 healthy and 0 unhealthy instances in us-west-2b. |

| Metric | Description |
| --- | --- |
| HTTPCode_Backend_2XX , HTTPCode_Backend_3XX , HTTPCode_Backend_4XX , HTTPCode_Backend_5XX | [HTTP listener] The number of HTTP response codes generated by registered instances. This count does not include any response codes generated by the load balancer.<br><br>**Reporting criteria**: There is a nonzero value<br><br>**Statistics**: The most useful statistic is Sum. Note that Minimum, Maximum, and Average are all 1.<br><br>**Example**: Suppose that your load balancer has 2 instances in us-west-2a and 2 instances in us-west-2b, and that requests sent to 1 instance in us-west-2a result in HTTP 500 responses. The sum for us-west-2a includes these error responses, while the sum for us-west-2b does not include them. Therefore, the sum for the load balancer equals the sum for us-west-2a. |
| HTTPCode_ELB_4XX | [HTTP listener] The number of HTTP 4XX client error codes generated by the load balancer. Client errors are generated when a request is malformed or incomplete.<br><br>**Reporting criteria**: There is a nonzero value<br><br>**Statistics**: The most useful statistic is Sum. Note that Minimum, Maximum, and Average are all 1.<br><br>**Example**: Suppose that your load balancer has us-west-2a and us-west-2b enabled, and that client requests include a malformed request URL. As a result, client errors would likely increase in all Availability Zones. The sum for the load balancer is the sum of the values for the Availability Zones. |

| Metric | Description |
| --- | --- |
| HTTPCode_ELB_5XX | [HTTP listener] The number of HTTP 5XX server error codes generated by the load balancer. This count does not include any response codes generated by the registered instances. The metric is reported if there are no healthy instances registered to the load balancer, or if the request rate exceeds the capacity of the instances (spillover) or the load balancer.<br><br>**Reporting criteria**: There is a nonzero value<br><br>**Statistics**: The most useful statistic is `Sum`. Note that `Minimum`, `Maximum`, and `Average` are all 1.<br><br>**Example**: Suppose that your load balancer has us-west-2a and us-west-2b enabled, and that instances in us-west-2a are experiencing high latency and are slow to respond to requests. As a result, the surge queue for the load balancer nodes in us-west-2a fills and clients receive a 503 error. If us-west-2b continues to respond normally, the sum for the load balancer equals the sum for us-west-2a. |

| Metric | Description |
| --- | --- |
| `Latency` | [HTTP listener] The total time elapsed, in seconds, from the time the load balancer sent the request to a registered instance until the instance started to send the response headers.<br><br>[TCP listener] The total time elapsed, in seconds, for the load balancer to successfully establish a connection to a registered instance.<br><br>**Reporting criteria**: There is a nonzero value<br><br>**Statistics**: The most useful statistic is `Average`. Use `Maximum` to determine whether some requests are taking substantially longer than the average. Note that `Minimum` is typically not useful.<br><br>**Example**: Suppose that your load balancer has 2 instances in us-west-2a and 2 instances in us-west-2b, and that requests sent to 1 instance in us-west-2a have a higher latency. The average for us-west-2a has a higher value than the average for us-west-2b. |

| Metric | Description |
| --- | --- |
| RequestCount | The number of requests completed or connections made during the specified interval (1 or 5 minutes).<br><br>[HTTP listener] The number of requests received and routed, including HTTP error responses from the registered instances.<br><br>[TCP listener] The number of connections made to the registered instances.<br><br>**Reporting criteria**: There is a nonzero value<br><br>**Statistics**: The most useful statistic is Sum. Note that Minimum, Maximum, and Average all return 1.<br><br>**Example**: Suppose that your load balancer has 2 instances in us-west-2a and 2 instances in us-west-2b, and that 100 requests are sent to the load balancer. There are 60 requests sent to us-west-2a, with each instance receiving 30 requests, and 40 requests sent to us-west-2b, with each instance receiving 20 requests. With the AvailabilityZone dimension, there is a sum of 60 requests in us-west-2a and 40 requests in us-west-2b. With the LoadBalancerName dimension, there is a sum of 100 requests. |

| Metric | Description |
| --- | --- |
| `SpilloverCount` | The total number of requests that were rejected because the surge queue is full.<br><br>[HTTP listener] The load balancer returns an HTTP 503 error code.<br><br>[TCP listener] The load balancer closes the connection.<br><br>**Reporting criteria**: There is a nonzero value<br><br>**Statistics**: The most useful statistic is `Sum`. Note that `Average`, `Minimum`, and `Maximum` are reported per load balancer node and are not typically useful.<br><br>**Example**: Suppose that your load balancer has us-west-2a and us-west-2b enabled, and that instances in us-west-2a are experiencing high latency and are slow to respond to requests. As a result, the surge queue for the load balancer node in us-west-2a fills, resulting in spillover. If us-west-2b continues to respond normally, the sum for the load balancer will be the same as the sum for us-west-2a. |

| Metric | Description |
| --- | --- |
| SurgeQueueLength | The total number of requests (HTTP listener) or connections (TCP listener) that are pending routing to a healthy instance. The maximum size of the queue is 1,024. Additional requests or connections are rejected when the queue is full. For more information, see SpilloverCount .<br><br>**Reporting criteria**: There is a nonzero value.<br><br>**Statistics**: The most useful statistic is Maximum, because it represents the peak of queued requests. The Average statistic can be useful in combination with Minimum and Maximum to determine the range of queued requests. Note that Sum is not useful.<br><br>**Example**: Suppose that your load balancer has us-west-2a and us-west-2b enabled, and that instances in us-west-2a are experiencing high latency and are slow to respond to requests. As a result, the surge queue for the load balancer nodes in us-west-2a fills, with clients likely experiencing increased response times. If this continues, the load balancer will likely have spillovers (see the SpilloverCount metric). If us-west-2b continues to respond normally, the max for the load balancer will be the same as the max for us-west-2a. |

| Metric | Description |
| --- | --- |
| UnHealthyHostCount | The number of unhealthy instances registered with your load balancer. An instance is considered unhealthy after it exceeds the unhealthy threshold configured for health checks. An unhealthy instance is considered healthy again after it meets the healthy threshold configured for health checks.<br><br>**Reporting criteria**: There are registered instances<br><br>**Statistics**: The most useful statistics are Average and Minimum. These statistics are determined by the load balancer nodes. Note that some load balancer nodes might determine that an instance is unhealthy for a brief period while other nodes determine that it is healthy.<br><br>**Example**: See HealthyHostCount . |

The following metrics enable you to estimate your costs if you migrate a Classic Load Balancer to an Application Load Balancer. These metrics are intended for informational use only, not for use with CloudWatch alarms. Note that if your Classic Load Balancer has multiple listeners, these metrics are aggregated across the listeners.

These estimates are based on a load balancer with one default rule and a certificate that is 2K in size. If you use a certificate that is 4K or greater in size, we recommend that you estimate your costs as follows: create an Application Load Balancer based on your Classic Load Balancer using the migration tool and monitor the ConsumedLCUs metric for the Application Load Balancer. For more information, see Migrate your Classic Load Balancer in the *Elastic Load Balancing User Guide*.

| Metric | Description |
| --- | --- |
| EstimatedALBActiveConnectionCount | The estimated number of concurrent TCP connections active from clients to the load balancer and from the load balancer to targets. |
| EstimatedALBConsumedLCUs | The estimated number of load balancer capacity units (LCU) used by an Application Load Balancer. You pay for the number of LCUs |

| Metric | Description |
| --- | --- |
| | that you use per hour. For more information, see Elastic Load Balancing Pricing. |
| EstimatedALBNewConnectionCount | The estimated number of new TCP connections established from clients to the load balancer and from the load balancer to targets. |
| EstimatedProcessedBytes | The estimated number of bytes processed by an Application Load Balancer. |

## Metric dimensions for Classic Load Balancers

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

| Dimension | Description |
| --- | --- |
| AvailabilityZone | Filters the metric data by the specified Availability Zone. |
| LoadBalancerName | Filters the metric data by the specified load balancer. |

## Statistics for Classic Load Balancer metrics

CloudWatch provides statistics based on the metric data points published by Elastic Load Balancing. Statistics are metric data aggregations over specified period of time. When you request statistics, the returned data stream is identified by the metric name and dimension. A dimension is a name/value pair that uniquely identifies a metric. For example, you can request statistics for all the healthy EC2 instances behind a load balancer launched in a specific Availability Zone.

The `Minimum` and `Maximum` statistics reflect the minimum and maximum reported by the individual load balancer nodes. For example, suppose there are 2 load balancer nodes. One node has `HealthyHostCount` with a `Minimum` of 2, a `Maximum` of 10, and an `Average` of 6, while the other node has `HealthyHostCount` with a `Minimum` of 1, a `Maximum` of 5, and an `Average` of 3. Therefore, the load balancer has a `Minimum` of 1, a `Maximum` of 10, and an `Average` of about 4.

The Sum statistic is the aggregate value across all load balancer nodes. Because metrics include multiple reports per period, Sum is only applicable to metrics that are aggregated across all load balancer nodes, such as RequestCount, HTTPCode_ELB_XXX, HTTPCode_Backend_XXX, BackendConnectionErrors, and SpilloverCount.

The SampleCount statistic is the number of samples measured. Because metrics are gathered based on sampling intervals and events, this statistic is typically not useful. For example, with HealthyHostCount, SampleCount is based on the number of samples that each load balancer node reports, not the number of healthy hosts.

A percentile indicates the relative standing of a value in a data set. You can specify any percentile, using up to two decimal places (for example, p95.45). For example, the 95th percentile means that 95 percent of the data is below this value and 5 percent is above. Percentiles are often used to isolate anomalies. For example, suppose that an application serves the majority of requests from a cache in 1-2 ms, but in 100-200 ms if the cache is empty. The maximum reflects the slowest case, around 200 ms. The average doesn't indicate the distribution of the data. Percentiles provide a more meaningful view of the application's performance. By using the 99th percentile as an Auto Scaling trigger or a CloudWatch alarm, you can target that no more than 1 percent of requests take longer than 2 ms to process.

## View CloudWatch metrics for your load balancer

You can view the CloudWatch metrics for your load balancers using the Amazon EC2 console. These metrics are displayed as monitoring graphs. The monitoring graphs show data points if the load balancer is active and receiving requests.

Alternatively, you can view metrics for your load balancer using the CloudWatch console.

**To view metrics using the console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Load Balancing**, choose **Load Balancers**.
3. Choose the name of the load balancer to open its detail page.
4. Choose the **Monitoring** tab.
5. To get a larger view of a single metric, hover over its graph then choose the Maximize icon. The following metrics are available:

   - Healthy Hosts — HealthyHostCount

- Unhealthy Hosts — `UnHealthyHostCount`
- Average Latency — `Latency`
- Requests — `RequestCount`
- Backend Connection Errors — `BackendConnectionErrors`
- Surge Queue Length — `SurgeQueueLength`
- Spillover Count — `SpilloverCount`
- HTTP 2XXs — `HTTPCode_Backend_2XX`
- HTTP 3XXs — `HTTPCode_Backend_3XX`
- HTTP 4XXs — `HTTPCode_Backend_4XX`
- HTTP 5XXs — `HTTPCode_Backend_5XX`
- ELB HTTP 4XXs — `HTTPCode_ELB_4XX`
- ELB HTTP 5XXs — `HTTPCode_ELB_5XX`
- Estimated processed bytes — `EstimatedProcessedBytes`
- Estimated ALB consumed LCUs — `EstimatedALBConsumedLCUs`
- Estimated ALB active connection count — `EstimatedALBActiveConnectionCount`
- Estimated ALB new connection count — `EstimatedALBNewConnectionCount`

**To view metrics using the CloudWatch console**

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Metrics**.
3. Select the **ELB** namespace.
4. Do one of the following:
   - Select a metric dimension to view metrics by load balancer, by Availability Zone, or across all load balancers.
   - To view a metric across all dimensions, type its name in the search field.
   - To view the metrics for a single load balancer, type its name in the search field.
   - To view the metrics for a single Availability Zone, type its name in the search field.

# Access logs for your Classic Load Balancer

Elastic Load Balancing provides access logs that capture detailed information about requests sent to your load balancer. Each log contains information such as the time the request was received, the client's IP address, latencies, request paths, and server responses. You can use these access logs to analyze traffic patterns and to troubleshoot issues.

Access logs are an optional feature of Elastic Load Balancing that is disabled by default. After you enable access logs for your load balancer, Elastic Load Balancing captures the logs and stores them in the Amazon S3 bucket that you specify. You can disable access logging at any time.

Each access log file is automatically encrypted using SSE-S3 before it is stored in your S3 bucket and decrypted when you access it. You do not need to take any action; the encryption and decryption is performed transparently. Each log file is encrypted with a unique key, which is itself encrypted with a KMS key that is regularly rotated. For more information, see Protecting data using server-side encryption with Amazon S3-managed encryption keys (SSE-S3) in the *Amazon S3 User Guide*.

There is no additional charge for access logs. You will be charged storage costs for Amazon S3, but will not be charged for the bandwidth used by Elastic Load Balancing to send log files to Amazon S3. For more information about storage costs, see Amazon S3 Pricing.

**Contents**



## Access log files

Elastic Load Balancing publishes a log file for each load balancer node at the interval you specify. You can specify a publishing interval of either 5 minutes or 60 minutes when you enable the access log for your load balancer. By default, Elastic Load Balancing publishes logs at a 60-minute interval. If the interval is set for 5 minutes, the logs are published at 1:05, 1:10, 1:15, and so on. The start of log delivery is delayed up to 5 minutes if the interval is set to 5 minutes, and up to 15 minutes if the interval is set to 60 minutes. You can modify the publishing interval at any time.

The load balancer can deliver multiple logs for the same period. This usually happens if the site has high traffic, multiple load balancer nodes, and a short log publishing interval.

The file names of the access logs use the following format:

```
amzn-s3-demo-loadbalancer-logs[/logging-prefix]/AWSLogs/aws-account-id/
elasticloadbalancing/region/yyyy/mm/dd/aws-account-id_elasticloadbalancing_region_load-
balancer-name_end-time_ip-address_random-string.log
```

*amzn-s3-demo-loadbalancer-logs*

The name of the S3 bucket.

*prefix*

(Optional) The prefix (logical hierarchy) for the bucket. The prefix that you specify must not include the string `AWSLogs`. For more information, see Organizing objects using prefixes.

`AWSLogs`

We add the portion of the file name starting with `AWSLogs` after the bucket name and optional prefix that you specify.

*aws-account-id*

The AWS account ID of the owner.

*region*

The Region for your load balancer and S3 bucket.

*yyyy/mm/dd*

The date that the log was delivered.

*load-balancer-name*

The name of the load balancer.

*end-time*

The date and time that the logging interval ended. For example, an end time of 20140215T2340Z contains entries for requests made between 23:35 and 23:40 if the publishing interval is 5 minutes.

*ip-address*

The IP address of the load balancer node that handled the request. For an internal load balancer, this is a private IP address.

*random-string*

A system-generated random string.

The following is an example log file name with a prefix of "my-app":

```
s3://amzn-s3-demo-loadbalancer-logs/my-app/AWSLogs/123456789012/elasticloadbalancing/
us-west-2/2018/02/15/123456789012_elasticloadbalancing_us-west-2_my-
loadbalancer_20180215T2340Z_172.160.001.192_20sg8hgm.log
```

The following is an example log file name without a prefix:

```
s3://amzn-s3-demo-loadbalancer-logs/AWSLogs/123456789012/elasticloadbalancing/
us-west-2/2018/02/15/123456789012_elasticloadbalancing_us-west-2_my-
loadbalancer_20180215T2340Z_172.160.001.192_20sg8hgm.log
```

You can store your log files in your bucket for as long as you want, but you can also define Amazon S3 lifecycle rules to archive or delete log files automatically. For more information, see Object lifecycle management in the *Amazon S3 User Guide*.

## Access log entries

Elastic Load Balancing logs requests sent to the load balancer, including requests that never made it to the back-end instances. For example, if a client sends a malformed request, or there are no healthy instances to respond, the requests are still logged.

**Important**

Elastic Load Balancing logs requests on a best-effort basis. We recommend that you use access logs to understand the nature of the requests, not as a complete accounting of all requests.

## Syntax

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

```
timestamp elb client:port backend:port request_processing_time backend_processing_time
 response_processing_time elb_status_code backend_status_code received_bytes sent_bytes
 "request" "user_agent" ssl_cipher ssl_protocol
```

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

| Field | Description |
| --- | --- |
| time | The time when the load balancer received the request from the client, in ISO 8601 format. |
| elb | The name of the load balancer |
| client:port | The IP address and port of the requesting client. |
| backend:port | The IP address and port of the registered instance that processed this request.<br><br>If the load balancer can't send the request to a registered instance, or if the instance closes the connection before a response can be sent, this value is set to `-`.<br><br>This value can also be set to `-` if the registered instance does not respond before the idle timeout. |
| request_processing_time | [HTTP listener] The total time elapsed, in seconds, from the time the load balancer received the request until the time it sent it to a registered instance.<br><br>[TCP listener] The total time elapsed, in seconds, from the time the load balancer accepted a TCP/SSL connection from a client to the time the load balancer sends the first byte of data to a registered instance.<br><br>This value is set to `-1` if the load balancer can't dispatch the request to a registered instance. This can happen if the registered instance |

| Field | Description |
| --- | --- |
| | closes the connection before the idle timeout or if the client sends a malformed request. Additionally, for TCP listeners, this can happen if the client establishes a connection with the load balancer but does not send any data.<br><br>This value can also be set to -1 if the registered instance does not respond before the idle timeout. |
| backend_processing_time | [HTTP listener] The total time elapsed, in seconds, from the time the load balancer sent the request to a registered instance until the instance started to send the response headers.<br><br>[TCP listener] The total time elapsed, in seconds, for the load balancer to successfully establish a connection to a registered instance.<br><br>This value is set to -1 if the load balancer can't dispatch the request to a registered instance. This can happen if the registered instance closes the connection before the idle timeout or if the client sends a malformed request.<br><br>This value can also be set to -1 if the registered instance does not respond before the idle timeout. |

| Field | Description |
| --- | --- |
| response_processing_time | [HTTP listener] The total time elapsed (in seconds) from the time the load balancer received the response header from the registered instance until it started to send the response to the client. This includes both the queuing time at the load balancer and the connection acquisition time from the load balancer to the client.<br><br>[TCP listener] The total time elapsed, in seconds, from the time the load balancer received the first byte from the registered instance until it started to send the response to the client.<br><br>This value is set to -1 if the load balancer can't dispatch the request to a registered instance. This can happen if the registered instance closes the connection before the idle timeout or if the client sends a malformed request.<br><br>This value can also be set to -1 if the registered instance does not respond before the idle timeout. |
| elb_status_code | [HTTP listener] The status code of the response from the load balancer. |
| backend_status_code | [HTTP listener] The status code of the response from the registered instance. |
| received_bytes | The size of the request, in bytes, received from the client (requester).<br><br>[HTTP listener] The value includes the request body but not the headers.<br><br>[TCP listener] The value includes the request body and the headers. |
| sent_bytes | The size of the response, in bytes, sent to the client (requester).<br><br>[HTTP listener] The value includes the response body but not the headers.<br><br>[TCP listener] The value includes the request body and the headers. |

| Field | Description |
| --- | --- |
| request | The request line from the client enclosed in double quotes and logged in the following format: HTTP Method + Protocol://Host header:port + Path + HTTP version. The load balancer preserves the URL sent by the client, as is, when recording the request URI. It does not set the content type for the access log file. When you process this field, consider how the client sent the URL.<br><br>[TCP listener] The URL is three dashes, each separated by a space, and ending with a space ("- - - "). |
| user_agent | [HTTP/HTTPS listener] A User-Agent string that identifies the client that originated the request. The string consists of one or more product identifiers, product[/version]. If the string is longer than 8 KB, it is truncated. |
| ssl_cipher | [HTTPS/SSL listener] The SSL cipher. This value is recorded only if the incoming SSL/TLS connection was established after a successful negotiation. Otherwise, the value is set to `-`. |
| ssl_protocol | [HTTPS/SSL listener] The SSL protocol. This value is recorded only if the incoming SSL/TLS connection was established after a successful negotiation. Otherwise, the value is set to `-`. |

## Examples

### Example HTTP entry

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

```
2015-05-13T23:39:43.945958Z my-loadbalancer 192.168.131.39:2817 10.0.0.1:80 0.000073
 0.001048 0.000057 200 200 0 29 "GET http://www.example.com:80/ HTTP/1.1" "curl/7.38.0"
 - -
```

### Example HTTPS entry

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

```
2015-05-13T23:39:43.945958Z my-loadbalancer 192.168.131.39:2817 10.0.0.1:80
 0.000086 0.001048 0.001337 200 200 0 57 "GET https://www.example.com:443/ HTTP/1.1"
 "curl/7.38.0" DHE-RSA-AES128-SHA TLSv1.2
```

**Example TCP entry**

The following is an example log entry for an TCP listener (port 8080 to port 80):

```
2015-05-13T23:39:43.945958Z my-loadbalancer 192.168.131.39:2817 10.0.0.1:80 0.001069
 0.000028 0.000041 - - 82 305 "- - - " "-" - -
```

**Example SSL entry**

The following is an example log entry for an SSL listener (port 8443 to port 80):

```
2015-05-13T23:39:43.945958Z my-loadbalancer 192.168.131.39:2817 10.0.0.1:80 0.001065
 0.000015 0.000023 - - 57 502 "- - - " "-" ECDHE-ECDSA-AES128-GCM-SHA256 TLSv1.2
```

## Processing access logs

If there is a lot of demand on your website, your load balancer can generate log files with gigabytes of data. You might not be able to process such a large amount of data using line-by-line processing. Therefore, you might have to use analytical tools that provide parallel processing solutions. For example, you can use the following analytical tools to analyze and process access logs:

- Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL. For more information, see Querying Classic Load Balancer logs in the *Amazon Athena User Guide*.
- Loggly
- Splunk
- Sumo Logic

## Enable access logs for your Classic Load Balancer

To enable access logs for your load balancer, you must specify the name of the Amazon S3 bucket where the load balancer will store the logs. You must also attach a bucket policy to this bucket that grants Elastic Load Balancing permission to write to the bucket.

**Tasks**

- Step 1: Create an S3 bucket
- Step 2: Attach a policy to your S3 bucket
- Step 3: Configure access logs
- Step 4: Verify bucket permissions
- Troubleshooting

## Step 1: Create an S3 bucket

When you enable access logs, you must specify an S3 bucket for the access log files. The bucket must meet the following requirements.

**Requirements**

- The bucket must be located in the same Region as the load balancer. The bucket and the load balancer can be owned by different accounts.
- The only server-side encryption option that's supported is Amazon S3-managed keys (SSE-S3). For more information, see Amazon S3-managed encryption keys (SSE-S3).

**To create an S3 bucket using the Amazon S3 console**

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose **Create bucket**.
3. On the **Create bucket** page, do the following:
    a. For **Bucket name**, enter a name for your bucket. This name must be unique across all existing bucket names in Amazon S3. In some Regions, there might be additional restrictions on bucket names. For more information, see Bucket quotas, limitations, and restrictions in the *Amazon S3 User Guide*.
    b. For **AWS Region**, select the Region where you created your load balancer.
    c. For **Default encryption**, choose **Amazon S3-managed keys (SSE-S3)**.
    d. Choose **Create bucket**.

## Step 2: Attach a policy to your S3 bucket

Your S3 bucket must have a bucket policy that grants Elastic Load Balancing permission to write the access logs to the bucket. Bucket policies are a collection of JSON statements written in the access policy language to define access permissions for your bucket. Each statement includes information about a single permission and contains a series of elements.

If you're using an existing bucket that already has an attached policy, you can add the statement for Elastic Load Balancing access logs to the policy. If you do so, we recommend that you evaluate the resulting set of permissions to ensure that they are appropriate for the users that need access to the bucket for access logs.

### Bucket policy

This policy grants permissions to the log delivery service.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "logdelivery.elasticloadbalancing.amazonaws.com"
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/prefix/AWSLogs/123456789012/*"
    }
  ]
}
```

For `Resource`, enter the ARN of the location for the access logs, using the format shown in the example policy. Always include the account ID of the account with the load balancer in the resource path of the S3 bucket ARN. This ensures that only load balancers from the specified account can write access logs to the S3 bucket.

The ARN that you specify depends on whether you plan to include a prefix when you enable access logs in step 3.

### Example S3 bucket ARN with a prefix

The S3 bucket name is amzn-s3-demo-logging-bucket and the prefix is logging-prefix.

```
arn:aws:s3:::amzn-s3-demo-logging-bucket/logging-prefix/AWSLogs/123456789012/*
```

[AWS GovCloud (US)] The following example uses the ARN syntax for the AWS GovCloud (US) Regions.

```
arn:aws-us-gov:s3:::amzn-s3-demo-logging-bucket/logging-prefix/AWSLogs/123456789012/*
```

**Example S3 bucket ARN with no prefix**

The S3 bucket name is amzn-s3-demo-logging-bucket. There is no prefix portion in the S3 bucket ARN.

```
arn:aws:s3:::amzn-s3-demo-logging-bucket/AWSLogs/123456789012/*
```

[AWS GovCloud (US)] The following example uses the ARN syntax for the AWS GovCloud (US) Regions.

```
arn:aws-us-gov:s3:::amzn-s3-demo-logging-bucket/AWSLogs/123456789012/*
```

**Legacy bucket policy**

Previously, for Regions available before August 2022, we required a policy that granted permissions to an Elastic Load Balancing account that was specific to the Region. This legacy policy is still supported, but we recommend that you replace it with the newer policy above. If you prefer to keep using the legacy bucket policy, which is not shown here, you can.

For reference, here are the IDs of the Elastic Load Balancing accounts to specify in `Principal`. Note that Regions that are not in this list never supported the legacy bucket policy.

- US East (N. Virginia) – 127311923021
- US East (Ohio) – 033677994240
- US West (N. California) – 027434742980
- US West (Oregon) – 797873946194
- Africa (Cape Town) – 098369216593
- Asia Pacific (Hong Kong) – 754344448648

- Asia Pacific (Jakarta) – 589379963580
- Asia Pacific (Mumbai) – 718504428378
- Asia Pacific (Osaka) – 383597477331
- Asia Pacific (Seoul) – 600734575887
- Asia Pacific (Singapore) – 114774131450
- Asia Pacific (Sydney) – 783225319266
- Asia Pacific (Tokyo) – 582318560864
- Canada (Central) – 985666609251
- Europe (Frankfurt) – 054676820928
- Europe (Ireland) – 156460612806
- Europe (London) – 652711504416
- Europe (Milan) – 635631232127
- Europe (Paris) – 009996457667
- Europe (Stockholm) – 897822967062
- Middle East (Bahrain) – 076674570225
- South America (São Paulo) – 507241528517
- AWS GovCloud (US-East) – 190560391635
- AWS GovCloud (US-West) – 048591011584

**Security best practices**

To enhance security, use precise S3 bucket ARNs.

- Use the full resource path, not just the S3 bucket ARN.
- Include the account ID portion of the S3 bucket ARN.
- Don't use wildcards (*) in the account ID portion of the S3 bucket ARN.

After you create your bucket policy, use an Amazon S3 interface, such as the Amazon S3 console or AWS CLI commands, to attach your bucket policy to your S3 bucket.

**To attach your bucket policy to your bucket using the console**

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. Select the name of the bucket to open its details page.
3. Choose **Permissions** and then choose **Bucket policy**, **Edit**.
4. Update the bucket policy to grant the required permissions.
5. Choose **Save changes**.

#### To attach your bucket policy to your S3 bucket using the AWS CLI

Use the put-bucket-policy command. In this example, the bucket policy was saved to the specified .json file.

```
aws s3api put-bucket-policy \
    --bucket amzn-s3-demo-bucket \
    --policy file://access-log-policy.json
```

## Step 3: Configure access logs

Use the following procedure to configure access logs to capture request information and deliver log files to your S3 bucket.

#### Requirements

The bucket must meet the requirements described in step 1, and you must attach a bucket policy as described in step 2. If you specify a prefix, it must not include the string "AWSLogs".

#### To configure access logs for your load balancer using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Load Balancing**, choose **Load Balancers**.
3. Select the name of your load balancer to open its details page.
4. On the **Attributes** tab, choose **Edit**.
5. On the **Edit load balancer attributes** page, in the **Monitoring** section, do the following:
   a. Enable **Access logs**.
   b. For **S3 URI**, enter the S3 URI for your log files. The URI that you specify depends on whether you're using a prefix.
      - URI with a prefix: `s3://amzn-s3-demo-logging-bucket/logging-prefix`
      - URI without a prefix: `s3://amzn-s3-demo-logging-bucket`

c. Keep **Logging interval** as `60 minutes - default`.

d. Choose **Save changes**.

### To configure access logs for your load balancer using the AWS CLI

First, create a .json file that enables Elastic Load Balancing to capture and deliver logs every 60 minutes to the S3 bucket that you created for the logs:

```
{
  "AccessLog": {
    "Enabled": true,
    "S3BucketName": "amzn-s3-demo-logging-bucket",
    "EmitInterval": 60,
    "S3BucketPrefix": "my-app"
  }
}
```

Next, specify the .json file in the modify-load-balancer-attributes command as follows:

```
aws elb modify-load-balancer-attributes --load-balancer-name my-loadbalancer --load-
balancer-attributes file://my-json-file.json
```

The following is an example response.

```
{
    "LoadBalancerAttributes": {
        "AccessLog": {
            "Enabled": true,
            "EmitInterval": 60,
            "S3BucketName": "amzn-s3-demo-logging-bucket",
            "S3BucketPrefix": "my-app"
        }
    },
    "LoadBalancerName": "my-loadbalancer"
}
```

### To manage the S3 bucket for your access logs

Be sure to disable access logs before you delete the bucket that you configured for access logs. Otherwise, if there is a new bucket with the same name and the required bucket policy created in

an AWS account that you don't own, Elastic Load Balancing could write the access logs for your load balancer to this new bucket.

## Step 4: Verify bucket permissions

After access logs are enabled for your load balancer, Elastic Load Balancing validates the S3 bucket and creates a test file to ensure that the bucket policy specifies the required permissions. You can use the S3 console to verify that the test file was created. The test file is not an actual access log file; it doesn't contain example records.

**To verify that Elastic Load Balancing created a test file in your S3 bucket**

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Select the name of the S3 bucket that you specified for access logs.
3. Navigate to the test file, `ELBAccessLogTestFile`. The location depends on whether you're using a prefix.
   - Location with a prefix: *amzn-s3-demo-loadbalancer-logs*/*logging-prefix*/AWSLogs/*123456789012*/ELBAccessLogTestFile
   - Location without a prefix: *amzn-s3-demo-loadbalancer-logs*/AWSLogs/*123456789012*/ELBAccessLogTestFile

## Troubleshooting

**Access Denied for bucket: *bucket-name*. Please check S3bucket permission**

If you receive this error, the following are possible causes:

- The bucket policy does not grant Elastic Load Balancing permission to write access logs to the bucket. Verify that you are using the correct bucket policy for the Region. Verify that the resource ARN uses the same bucket name that you specified when you enabled access logs. Verify that the resource ARN does not include a prefix if you did not specify a prefix when you enabled access logs.
- The bucket uses an unsupported server-side encryption option. The bucket must use Amazon S3-managed keys (SSE-S3).

# Disable access logs for your Classic Load Balancer

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

### To disable access logs for your load balancer using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Load Balancing**, choose **Load Balancers**.
3. Select the name of your load balancer to open its details page.
4. On the **Attributes** tab, choose **Edit**.
5. On the **Edit load balancer attributes** page, in the **Monitoring** section, disable **Access logs**.

### To disable access logs using the AWS CLI

Use the following modify-load-balancer-attributes command to disable access logs:

```
aws elb modify-load-balancer-attributes --load-balancer-name my-loadbalancer --load-balancer-attributes "{\"AccessLog\":{\"Enabled\":false}}"
```

The following is an example response:

```
{
    "LoadBalancerName": "my-loadbalancer",
    "LoadBalancerAttributes": {
        "AccessLog": {
            "S3BucketName": "amzn-s3-demo-loadbalancer-logs",
            "EmitInterval": 60,
            "Enabled": false,
            "S3BucketPrefix": "my-app"
        }
    }
}
```

# Troubleshoot your Classic Load Balancer

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

### API errors

| Error |
| --- |
| CertificateNotFound: Undefined |
| OutofService: A transient error occurred |

### HTTP errors

| Error |
| --- |
| HTTP 400: BAD_REQUEST |
| HTTP 405: METHOD_NOT_ALLOWED |
| HTTP 408: Request timeout |
| HTTP 502: Bad gateway |
| HTTP 503: Service unavailable |
| HTTP 504: Gateway timeout |

### Response code metrics

| Response code metric |
| --- |
| HTTPCode_ELB_4XX |
| HTTPCode_ELB_5XX |
| HTTPCode_Backend_2XX |

| Response code metric |
| --- |
| HTTPCode_Backend_3XX |
| HTTPCode_Backend_4XX |
| HTTPCode_Backend_5XX |

### Health check issues

| Issue |
| --- |
| Health check target page error |
| Connection to the instances has timed out |
| Public key authentication is failing |
| Instance is not receiving traffic from the load balancer |
| Ports on instance are not open |
| Instances in an Auto Scaling group are failing the ELB health check |

### Connectivity issues

| Issue |
| --- |
| Clients cannot connect to an internet-facing load balancer |
| Requests sent to a custom domain aren't received by the load balancer |
| HTTPS requests sent to the load balancer return "NET::ERR_CERT_COMMON_NAME_INVALID" |

### Instance registration issues

| Issue |
| --- |
| Taking too long to register an EC2 instance |

| Issue |
| --- |
| Unable to register an instance launched from a paid AMI |

# Troubleshoot a Classic Load Balancer: API errors

The following are error messages returned by Elastic Load Balancing API, the potential causes, and the steps you can take to resolve the issues.

**Error Messages**

- CertificateNotFound: Undefined
- OutofService: A transient error occurred

## CertificateNotFound: Undefined

**Cause 1**: There is a delay in propagating the certificate to all Regions when it is created using the AWS Management Console. When this delay occurs, the error message is shown in the last step in the process of creating the load balancer.

**Solution 1**: Wait approximately 15 minutes and then try again. If the problem persists, go to the AWS Support Center for assistance.

**Cause 2**: If you are using the AWS CLI or API directly, you can receive this error if you provide an Amazon Resource Name (ARN) for a certificate that does not exist.

**Solution 2**: Use the AWS Identity and Access Management (IAM) action GetServerCertificate to get the certificate ARN and verify that you provided the correct value for the ARN.

## OutofService: A transient error occurred

**Cause**: There is a transient internal problem within the Elastic Load Balancing service or the underlying network. This temporary issue might also occur when Elastic Load Balancing queries the health of the load balancer and its registered instances.

**Solution**: Retry the API call. If the problem persists, go to the AWS Support Center for assistance.

# Troubleshoot a Classic Load Balancer: HTTP errors

The HTTP method (also called the *verb*) specifies the action to be performed on the resource receiving an HTTP request. The standard methods for HTTP requests are defined in RFC 2616, Method Definitions. The standard methods include GET, POST, PUT, HEAD, and OPTIONS. Some web applications require (and sometimes introduce) methods that are extensions of HTTP/1.1 methods. Common examples of HTTP extended methods include PATCH, REPORT, MKCOL, PROPFIND, MOVE, and LOCK. Elastic Load Balancing accepts all standard and non-standard HTTP methods.

HTTP requests and responses use header fields to send information about the HTTP messages. Header fields are colon-separated name-value pairs that are separated by a cariage return (CR) and a line feed (LF). A standard set of HTTP header fields is definied in RFC 2616, Message Headers. For more information, see HTTP headers and Classic Load Balancers.

When a load balancer receives an HTTP request, it checks for malformed requests and for the length of the method. The total method length in an HTTP request to a load balancer must not exceed 127 characters. If the HTTP request passes both checks, the load balancer sends the request to the EC2 instance. If the method field in the request is malformed, the load balancer responds with an HTTP 400: BAD_REQUEST error. If the length of the method in the request exceeds 127 characters, the load balancer responds with an HTTP 405: METHOD_NOT_ALLOWED error.

The EC2 instance processes a valid request by implementing the method in the request and sending a response back to the client. Your instances must be configured to handle both supported and unsupported methods.

The following are error messages returned by your load balancer, the potential causes, and the steps you can take to resolve the issues.

**Error Messages**

- HTTP 400: BAD_REQUEST
- HTTP 405: METHOD_NOT_ALLOWED
- HTTP 408: Request timeout
- HTTP 502: Bad gateway
- HTTP 503: Service unavailable
- HTTP 504: Gateway timeout

## HTTP 400: BAD_REQUEST

**Description**: Indicates that the client sent a bad request.

**Cause 1**: The client sent a malformed request that does not meet HTTP specifications. For example, a request can't have spaces in the URL.

**Cause 2**: The client used the HTTP CONNECT method, which is not supported by Elastic Load Balancing.

**Solution**: Connect directly to your instance and capture the details of the client request. Review the headers and the URL for malformed requests. Verify that the request meets HTTP specifications. Verify that HTTP CONNECT is not used.

## HTTP 405: METHOD_NOT_ALLOWED

**Description**: Indicates that the method length is not valid.

**Cause**: The length of the method in the request header exceeds 127 characters.

**Solution**: Check the length of the method.

## HTTP 408: Request timeout

**Description**: Indicates that the client cancelled the request or failed to send a full request.

**Cause 1**: A network interruption or a bad request construction, such as partially formed headers; specified content size doesn't match the actual content size transmitted; and so on.

**Solution 1**: Inspect the code that is making the request and try sending it directly to your registered instances (or a development / test environment) where you have more control over inspecting the actual request.

**Cause 2**: Connection to the client is closed (load balancer could not send a response)

**Solution 2**: Verify that the client is not closing the connection before a response is sent by using a packet sniffer on the machine making the request.

## HTTP 502: Bad gateway

**Description**: Indicates that the load balancer was unable to parse the response sent from a registered instance.

**Cause**: Malformed response from the instance or potentially an issue with the load balancer.

**Solution**: Verify that the response being sent from the instance conforms to HTTP specifications. Go to the AWS Support Center for assistance.

## HTTP 503: Service unavailable

**Description**: Indicates that either the load balancer or the registered instances are causing the error.

**Cause 1**: Insufficient capacity in the load balancer to handle the request.

**Solution 1**: This should be a transient issue and should not last more than a few minutes. If it persists, go to the AWS Support Center for assistance.

**Cause 2**: There are no registered instances.

**Solution 2**: Register at least one instance in every Availability Zone that your load balancer is configured to respond in. Verify this by looking at the `HealthyHostCount` metrics in CloudWatch. If you can't ensure that an instance is registered in each Availability Zone, we recommend enabling cross-zone load balancing. For more information, see Configure cross-zone load balancing for your Classic Load Balancer.

**Cause 3**: There are no healthy instances.

**Solution 3**: Ensure that you have healthy instances in every Availability Zone that your load balancer is configured to respond in. Verify this by looking at the `HealthyHostCount` metric.

**Cause 4**: The surge queue is full.

**Solution 4**: Ensure that your instances have sufficient capacity to handle the request rate. Verify this by looking at the `SpilloverCount` metric.

## HTTP 504: Gateway timeout

**Description**: Indicates that the load balancer closed a connection because a request did not complete within the idle timeout period.

**Cause 1**: The application takes longer to respond than the configured idle timeout.

**Solution 1**: Monitor the `HTTPCode_ELB_5XX` and `Latency` metrics. If there is an increase in these metrics, it could be due to the application not responding within the idle timeout period. For details about the requests that are timing out, enable access logs on the load balancer and review

the 504 response codes in the logs that are generated by Elastic Load Balancing. If necessary, you can increase your capacity or increase the configured idle timeout so that lengthy operations (such as uploading a large file) can complete. For more information, see Configure the idle connection timeout for your Classic Load Balancer and How do I troubleshoot Elastic Load Balancing high latency.

**Cause 2**: Registered instances closing the connection to Elastic Load Balancing.

**Solution 2**: Enable keep-alive settings on your EC2 instances and make sure that the keep-alive timeout is greater than the idle timeout settings of your load balancer.

# Troubleshoot a Classic Load Balancer: Response code metrics

Your load balancer sends metrics to Amazon CloudWatch for the HTTP response codes sent to clients, identifying the source of the errors as either the load balancer or the registered instances. You can use the metrics returned by CloudWatch for your load balancer to troubleshoot issues. For more information, see CloudWatch metrics for your Classic Load Balancer.

The following are response code metrics returned by CloudWatch for your load balancer, the potential causes, and the steps you can take to resolve the issues.

**Response Code Metrics**

- HTTPCode_ELB_4XX
- HTTPCode_ELB_5XX
- HTTPCode_Backend_2XX
- HTTPCode_Backend_3XX
- HTTPCode_Backend_4XX
- HTTPCode_Backend_5XX

## HTTPCode_ELB_4XX

**Cause**: A malformed or canceled request from the client.

**Solutions**

- See HTTP 400: BAD_REQUEST.
- See HTTP 405: METHOD_NOT_ALLOWED.

- See HTTP 408: Request timeout.

## HTTPCode_ELB_5XX

**Cause**: Either the load balancer or the registered instance is causing the error or the load balancer is unable to parse the response.

**Solutions**

- See HTTP 502: Bad gateway.
- See HTTP 503: Service unavailable.
- See HTTP 504: Gateway timeout.

## HTTPCode_Backend_2XX

**Cause**: A normal, successful response from the registered instances.

**Solution**: None.

## HTTPCode_Backend_3XX

**Cause**: A redirect response sent from the registered instances.

**Solution**: View the access logs or the error logs on your instance to determine the cause. Send requests directly to the instance (bypassing the load balancer) to view the responses.

## HTTPCode_Backend_4XX

**Cause**: A client error response sent from the registered instances.

**Solution**: View the access or error logs on your instances to determine the cause. Send requests directly to the instance (bypass the load balancer) to view the responses.

> **Note**
>
> If the client cancels an HTTP request that was initiated with a `Transfer-Encoding: chunked` header, there is a known issue where the load balancer forwards the request to the instance even though the client canceled the request. This can cause backend errors.

### HTTPCode_Backend_5XX

**Cause**: A server error response sent from the registered instances.

**Solution**: View the access logs or the error logs on your instances to determine the cause. Send requests directly to the instance (bypass the load balancer) to view the responses.

> **Note**
>
> If the client cancels an HTTP request that was initiated with a `Transfer-Encoding: chunked` header, there is a known issue where the load balancer forwards the request to the instance even though the client canceled the request. This can cause backend errors.

## Troubleshoot a Classic Load Balancer: Health checks

Your load balancer checks the health of its registered instances using either the default health check configuration provided by Elastic Load Balancing or a custom health check configuration that you specify. The health check configuration contains information such as the protocol, ping port, ping path, response timeout, and health check interval. An instance is considered healthy if it returns a 200 response code within the health check interval. For more information, see Health checks for the instances for your Classic Load Balancer.

If the current state of some or all your instances is `OutOfService` and the description field displays the message that the `Instance has failed at least the Unhealthy Threshold number of health checks consecutively`, the instances have failed the load balancer health check. The following are the issues to look for, the potential causes, and the steps you can take to resolve the issues.

**Issues**

- Health check target page error
- Connection to the instances has timed out
- Public key authentication is failing
- Instance is not receiving traffic from the load balancer
- Ports on instance are not open
- Instances in an Auto Scaling group are failing the ELB health check

## Health check target page error

**Problem**: An HTTP GET request issued to the instance on the specified ping port and the ping path (for example, HTTP:80/index.html) receives a non-200 response code.

**Cause 1**: No target page is configured on the instance.

**Solution 1**: Create a target page (for example, `index.html`) on each registered instance and specify its path as the ping path.

**Cause 2**: The value of the Content-Length header in the response is not set.

**Solution 2**: If the response includes a body, then either set the Content-Length header to a value greater than or equal to zero, or set the Transfer-Encoding value to 'chunked'.

**Cause 3**: The application is not configured to receive requests from the load balancer or to return a 200 response code.

**Solution 3**: Check the application on your instance to investigate the cause.

## Connection to the instances has timed out

**Problem**: Health check requests from your load balancer to your EC2 instances are timing out or failing intermittently.

First, verify the issue by connecting directly with the instance. We recommend that you connect to your instance from within the network using the private IP address of the instance.

Use the following command for a TCP connection:

```
telnet private-IP-address-of-the-instance port
```

Use the following command for an HTTP or HTTPS connection:

```
curl -I private-IP-address-of-the-instance:port/health-check-target-page
```

If you are using an HTTP/HTTPS connection and getting a non-200 response, see Health check target page error. If you are able to connect directly to the instance, check for the following:

**Cause 1**: The instance is failing to respond within the configured response timeout period.

**Solution 1**: Adjust the response timeout settings in your load balancer health check configuration.

**Cause 2**: The instance is under significant load and is taking longer than your configured response timeout period to respond.

**Solution 2**:

- Check the monitoring graph for over-utilization of CPU. For information, see Get statistics for a specific EC2 instance in the *Amazon EC2 User Guide*.
- Check the utilization of other application resources, such as memory or limits, by connecting to your EC2 instances.
- If necessary, add more instances or enable Auto Scaling. For more information, see the Amazon EC2 Auto Scaling User Guide.

**Cause 3**: If you are using an HTTP or an HTTPS connection and the health check is being performed on a target page specified in the ping path field (for example, `HTTP:80/index.html`), the target page might be taking longer to respond than your configured timeout.

**Solution 3**: Use a simpler health check target page or adjust the health check interval settings.

## Public key authentication is failing

**Problem**: A load balancer configured to use the HTTPS or SSL protocol with back-end authentication enabled fails public key authentication.

**Cause**: The public key on the SSL certificate does not match the public key configured on the load balancer. Use the `s_client` command to see the list of server certificates in the certificate chain. For more information, see s_client in the OpenSSL documentation.

**Solution**: Your might need to update your SSL certificate. If your SSL certificate is current, try re-installing it on your load balancer. For more information, see Replace the SSL certificate for your Classic Load Balancer.

## Instance is not receiving traffic from the load balancer

**Problem**: The security group for the instance is blocking the traffic from the load balancer.

Do a packet capture on the instance to verify the issue. Use the following command:

```
# tcpdump port health-check-port
```

**Cause 1**: The security group associated with the instance does not allow traffic from the load balancer.

**Solution 1**: Edit the instance security group to allow traffic from the load balancer. Add a rule to allow all traffic from the load balancer security group.

**Cause 2**: The security group for your load balancer does not allow traffic to the EC2 instances.

**Solution 2**: Edit the security group of your load balancer to allow traffic to the subnets and the EC2 instances.

For information about managing security groups, see Configure security groups for your Classic Load Balancer.

## Ports on instance are not open

**Problem**: The health check sent to the EC2 instance by the load balancer is blocked by the port or a firewall.

Verify the issue by using the following command:

```
netstat -ant
```

**Cause**: The specified health port or the listener port (if configured differently) is not open. Both the port specified for the health check and the listener port must be open and listening.

**Solution**: Open up the listener port and the port specified in your health check configuration (if configured differently) on your instances to receive load balancer traffic.

## Instances in an Auto Scaling group are failing the ELB health check

**Problem**: Instances in your Auto Scaling group pass the default Auto Scaling health check but fail the ELB health check.

**Cause**: Auto Scaling uses EC2 status checks to detect hardware and software issues with the instances, but the load balancer performs health checks by sending a request to the instance and waiting for a 200 response code, or by establishing a TCP connection (for a TCP-based health check) with the instance.

An instance might fail the ELB health check because an application running on the instance has issues that cause the load balancer to consider the instance out of service. This instance might pass the Auto Scaling health check; it would not be replaced by the Auto Scaling policy because it is considered healthy based on the EC2 status check.

**Solution**: Use the ELB health check for your Auto Scaling group. When you use the ELB health check, Auto Scaling determines the health status of your instances by checking the results of both the instance status check and the ELB health check. For more information, see Add Elastic Load Balancing health checks to your Auto Scaling group in the *Amazon EC2 Auto Scaling User Guide*.

# Troubleshoot a Classic Load Balancer: Client connectivity

## Clients cannot connect to an internet-facing load balancer

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

**Your internet-facing load balancer is attached to a private subnet**

You must specify public subnets for your load balancer. A public subnet has a route to the internet gateway for your virtual private cloud (VPC).

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

The security group for the load balancer and any network ACLs for the load balancer subnets must allow inbound traffic from the clients and outbound traffic to the clients on the listener ports. For more information, see Configure security groups for your Classic Load Balancer.

## Requests sent to a custom domain aren't received by the load balancer

If the load balancer is not receiving requests sent to a custom domain, check for the following issues:

**The custom domain name does not resolve to the load balancer IP address**

- Confirm what IP address the custom domain name resolves to using a command line interface.
  - **Linux, macOS, or Unix** – You can use the `dig` command within Terminal. Ex.`dig example.com`
  - **Windows** – You can use the `nslookup` command within Command Prompt. Ex.`nslookup example.com`

- Confirm what IP address the load balancers DNS name resolves to using a command line interface.
- Compare the results of the two outputs. The IP addresses need to match.

### HTTPS requests sent to the load balancer return "NET::ERR_CERT_COMMON_NAME_INVALID"

If HTTPS requests are receiving `NET::ERR_CERT_COMMON_NAME_INVALID` from the load balancer, check the following possible causes:

- The domain name used in the HTTPS request does not match the alternate name specified in the listeners associated ACM certificate.
- The load balancers default DNS name is being used. The default DNS name cannot be used to make HTTPS requests as a public certificate cannot be requested for the `*.amazonaws.com` domain.

## Troubleshoot a Classic Load Balancer: Instance registration

When you register an instance with your load balancer, there are a number of steps that are taken before the load balancer can begin to send requests to your instance.

The following are issues your load balancer might encounter when registering your EC2 instances, the potential causes, and the steps you can take to resolve the issues.

**Issues**

- Taking too long to register an EC2 instance
- Unable to register an instance launched from a paid AMI

### Taking too long to register an EC2 instance

**Problem**: Registered EC2 instances are taking longer than expected to be in the `InService` state.

**Cause**: Your instance might be failing the health check. After the initial instance registration steps are completed (it can take up to approximately 30 seconds), the load balancer starts sending health check requests. Your instance is not `InService` until one health check succeeds.

**Solution**: See Connection to the instances has timed out.

## Unable to register an instance launched from a paid AMI

**Problem**: Elastic Load Balancing is not registering an instance launched using a paid AMI.

**Cause**: Your instances might have been launched using a paid AMI from Amazon DevPay.

**Solution**: Elastic Load Balancing does not support registering instances launched using paid AMIs from Amazon DevPay. Note that you can use paid AMIs from AWS Marketplace. If you are already using a paid AMI from AWS Marketplace and are unable to register an instance launched from that paid AMI, go to the AWS Support Center for assistance.

# Quotas for your Classic Load Balancer

Your AWS account has default quotas, formerly referred to as limits, for each AWS service. Unless otherwise noted, each quota is Region-specific.

To view the quotas for your Classic Load Balancers, open the Service Quotas console. In the navigation pane, choose **AWS services** and select **Elastic Load Balancing**. You can also use the describe-account-limits (AWS CLI) command for Elastic Load Balancing.

To request a quota increase, see Requesting a quota increase in the *Service Quotas User Guide*.

Your AWS account has the following quotas related to Classic Load Balancers.

| Name | Default | Adjustable |
| --- | --- | --- |
| Classic Load Balancers per Region | 20 | Yes |
| Listeners per Classic Load Balancer | 100 | Yes |
| Registered Instances per Classic Load Balancer | 1,000 | Yes |

# Document history for Classic Load Balancers

The following table describes the releases for Classic Load Balancers.

| Change | Description | Date |
| --- | --- | --- |
| Bucket policies for access logs and connection logs | Prior to this release, the bucket policy that you used depended on whether the Region was available before or after August 2022. With this release, the newer bucket policy is supported in all Regions. Note that the legacy bucket policy is still supported. | September 10, 2025 |
| Desync mitigation mode | Added support for desync mitigation mode. For more information, see Configure desync mitigation mode for your Classic Load Balancer. | August 17, 2020 |
| Classic Load Balancers | With the introduction of Application Load Balancers and Network Load Balancers , load balancers created with the 2016-06-01 API are now known as *Classic Load Balancers*. For more inf ormation about the differenc es between these types of load balancers, see Elastic Load Balancing features . | August 11, 2016 |
| Support for AWS Certificate Manager (ACM) | You can request an SSL/ TLS certificate from ACM | January 21, 2016 |

| | | |
|---|---|---|
| | and deploy it to your load balancer. For more informati on, see SSL/TLS certificates for Classic Load Balancers. | |
| Support for additional ports | Load balancers can listen on any port in the range 1-65535. For more informati on, see Listeners for your Classic Load Balancer. | September 15, 2015 |
| Additional fields for access log entries | Added the `user_agen t` , `ssl_cipher` , and `ssl_protocol` fields. For more information, see Access log files. | May 18, 2015 |
| Support for tagging your load balancer | Starting with this release, Elastic Load Balancing CLI (ELB CLI) has been replaced by AWS Command Line Interface (AWS CLI), a unified tool to manage multiple AWS services. New features rel eased after ELB CLI version 1.0.35.0 (dated 7/24/14) will be included in the AWS CLI only. If you are currently using the ELB CLI, we recommend that you start using the AWS CLI instead. For more information, see the *AWS Command Line Interface User Guide*. | August 11, 2014 |

| | | |
|---|---|---|
| Idle connection timeout | You can configure the idle connection timeout for your load balancer. | July 24, 2014 |
| Support for granting users and groups access to specific load balancers or API actions | You can create a policy to grant users and groups access to specific load balancers or API actions. | May 12, 2014 |
| Support for AWS CloudTrail | You can use CloudTrail to capture API calls made by or on behalf of your AWS account using the ELB API, the AWS Management Console, the ELB CLI, or the AWS CLI. | April 4, 2014 |
| Connection draining | Added information about connection draining. With this support you can enable your load balancer to stop sending new requests to the registere d instance when the instance is de-registering or when the instance becomes unhealthy , while keeping the existing connections open. For more information, see Configure connection draining for your Classic Load Balancer. | March 20, 2014 |

| | | |
|---|---|---|
| Access logs | You can enable your load balancer to capture detailed information about the requests sent to your load balancer and store it in an Amazon S3 bucket. For more information, see Access logs for your Classic Load Balancer. | March 6, 2014 |
| Support for TLSv1.1-1.2 | Added information about TLSv1.1-1.2 protocol support for load balancers configured with HTTPS/SSL listeners. With this support, Elastic Load Balancing also updates the predefined SSL negotiation configurations. For information about the updated predefined SSL negotiation configurations, see SSL negotiation configurations for Classic Load Balancers. For information about updating your current SSL negotiation configuration, see Update the SSL negotiation configuration of your Classic Load Balancer. | February 19, 2014 |

| | | |
|---|---|---|
| Cross-zone load balancing | Added information about enabling cross-zone load balancing for your load balancer. For more informati on, see Configure cross-zon e load balancing for your Classic Load Balancer. | November 6, 2013 |
| Additional CloudWatch Metrics | Added information about the additional Cloudwatch metrics reported by Elastic Load Balancing. For more inf ormation, see CloudWatch metrics for your Classic Load Balancer. | October 28, 2013 |
| Support for proxy protocol | Added information about proxy protocol support for load balancers configured for TCP/SSL connections. For more information, see Proxy protocol header. | July 30, 2013 |
| Support for DNS failover | Added information about configuring Amazon Route 53 DNS failover for load balancers. For more inf ormation, see Using Amazon Route 53 DNS failover for your load balancer. | June 3, 2013 |

| | | |
|---|---|---|
| Console support for viewing CloudWatch metrics and creating alarms | Added information about viewing CloudWatch metrics and creating alarms for a specified load balancer using the console. For more information, see CloudWatch metrics for your Classic Load Balancer. | March 28, 2013 |
| Support for registering EC2 instances in a default VPC | Added support for EC2 instances launched in a default VPC. | March 11, 2013 |
| Internal load balancers | With this release, a load balancer in a virtual private cloud (VPC) can be made either internal or internet-facing. An internal load balancer has a publicly resolvable DNS name tha t resolves to private IP addresses. An internet-facing load balancer has a publicly resolvable DNS name that resolves to public IP addresses . For more information, see Create an internal Classic Load Balancer. | June 10, 2012 |
| Console support for managing listeners, cipher settings, and SSL certificates | For information, see Configure an HTTPS listener for your Classic Load Balancer and Replace the SSL certifica te for your Classic Load Balancer. | May 18, 2012 |

| | | |
|---|---|---|
| Support for Elastic Load Balancing in Amazon VPC | Added support for creating a load balancer in a virtual private cloud (VPC). | November 21, 2011 |
| Amazon CloudWatch | You can monitor your load balancer using CloudWatch. For more information, see CloudWatch metrics for your Classic Load Balancer. | October 17, 2011 |
| Additional security features | You can configure SSL ciphers, back-end SSL, and back-end server authentication. For more information, see Create a Classic Load Balancer with an HTTPS listener. | August 30, 2011 |
| Zone apex domain name | For more information, see Configure a custom domain name for your Classic Load Balancer. | May 24, 2011 |

| | | |
|---|---|---|
| Support for X-Forwarded-Proto and X-Forwarded-Port headers | The X-Forwarded-Proto header indicates the protocol of the originating request, and the X-Forwarded-Port header indicates the port of the originating request. The addition of these headers to requests enables customers to determine if an incoming request to their load balancer is encrypted, and the specific port on the load balancer that the request was received on. For more information, see HTTP headers and Classic Load Balancers. | October 27, 2010 |
| Support for HTTPS | With this release, you can leverage the SSL/TLS protocol for encrypting traffic and offload SSL processin g from the application instance to the load balancer. This feature also provides centralized management of SSL server certificates at the load balancer, rather than managing certificates on individual application instances. | October 14, 2010 |
| Support for AWS Identity and Access Management (IAM) | Added support for IAM. | September 2, 2010 |
| Sticky sessions | For more information, see Configure sticky sessions for your Classic Load Balancer. | April 7, 2010 |

| | | |
|---|---|---|
| AWS SDK for Java | Added support for the SDK for Java. | March 22, 2010 |
| AWS SDK for .NET | Added support for the SDK for .NET. | November 11, 2009 |
| New service | Initial public beta release of Elastic Load Balancing. | May 18, 2009 |