
Elastic Load Balancing

Developer Guide



Elastic Load Balancing: Developer Guide

Copyright © 2015 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

The following are trademarks of Amazon Web Services, Inc.: Amazon, Amazon Web Services Design, AWS, Amazon CloudFront, AWS CloudTrail, AWS CodeDeploy, Amazon Cognito, Amazon DevPay, DynamoDB, ElastiCache, Amazon EC2, Amazon Elastic Compute Cloud, Amazon Glacier, Amazon Kinesis, Kindle, Kindle Fire, AWS Marketplace Design, Mechanical Turk, Amazon Redshift, Amazon Route 53, Amazon S3, Amazon VPC, and Amazon WorkDocs. In addition, Amazon.com graphics, logos, page headers, button icons, scripts, and service names are trademarks, or trade dress of Amazon in the U.S. and/or other countries. Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon.

All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What Is Elastic Load Balancing?	1
Features of Elastic Load Balancing	1
How to Get Started with Elastic Load Balancing	2
Related Services	2
Accessing Elastic Load Balancing	2
Pricing	3
How Elastic Load Balancing Works	3
Request Routing	4
Availability Zones and Instances	4
Related Topics	4
Setting Up	5
Sign Up	5
Prepare Your VPC and Back-end Instances	5
Access Elastic Load Balancing Using the AWS Management Console	7
Access Elastic Load Balancing Using the AWS CLI	7
Getting Started	8
Before You Begin	8
Step 1: Define Your Load Balancer	9
Step 2: Assign Security Groups to Your Load Balancer in a VPC	10
Step 3: Configure Security Settings (Optional)	10
Step 4: Configure Health Checks for Your EC2 Instances	11
Step 5: Register EC2 Instances with Your Load Balancer	11
Step 6: Tag Your Load Balancer (Optional)	36
Step 7: Create and Verify Your Load Balancer	12
Step 8: Delete Your Load Balancer (Optional)	12
Internet-Facing Load Balancers	13
Public DNS Names for Your Load Balancer	13
Create an Internet-Facing Load Balancer	14
Internal Load Balancers	15
Public DNS Name for Your Load Balancer	16
Create an Internal Load Balancer	16
Prerequisites	16
Create an Internal Load Balancer Using the Console	16
Create an Internal Load Balancer Using the AWS CLI	17
HTTPS Load Balancers	20
SSL Certificates	20
Prerequisite: Install Certificate Tools	21
Create a Server Certificate	21
Upload the Signed Certificate	22
Verify Server Certificate	25
Install the SSL Certificate on Your Load Balancer	25
SSL Negotiation Configurations	26
SSL Security Policies	27
Predefined SSL Security Policies	27
Create an HTTPS Load Balancer	31
Prerequisites	31
Create an HTTPS/SSL Load Balancer Using the Console	32
Create an HTTPS/SSL Load Balancer Using the AWS CLI	37
Configure an HTTPS Listener	45
Prerequisites	46
Add an HTTPS Listener Using the Console	46
Add an HTTPS Listener Using the AWS CLI	47
Update the SSL Certificate	48
Prerequisites	48
Updating an SSL Certificate Using the Console	49

Updating an SSL Certificate Using the AWS CLI	49
Update the SSL Negotiation Configuration	50
Update the SSL Negotiation Configuration Using the Console	50
Update the SSL Negotiation Configuration Using the AWS CLI	51
Back-end Instances	53
Best Practices for Your Back-end Instances	53
Configure Health Checks	54
Health Check Configuration	54
Update the Health Check Configuration	56
Check the Health of Your Instances	56
Troubleshoot Health Checks	56
Configure Security Groups	57
Security Groups for Load Balancers in EC2-Classic	57
Security Groups for Load Balancers in a VPC	60
Add or Remove Availability Zones	61
Add or Remove an Availability Zone Using the AWS Management Console	62
Add or Remove an Availability Zone Using the AWS CLI	63
Add or Remove Subnets	65
Add or Remove Subnets Using the Console	65
Attach or Detach Subnets Using the AWS CLI	66
De-register and Register Instances	67
De-register and Register Your Instances Using the Console	68
De-register and Register Your Instances Using the AWS CLI	68
Listeners	70
Protocols	70
TCP/SSL Protocol	71
HTTP/HTTPS Protocol	71
HTTPS/SSL Listeners	72
SSL Server Certificates	72
SSL Negotiation	72
Back-End Server Authentication	72
Listener Configurations Quick Reference	72
X-Forwarded Headers	74
X-Forwarded-For	74
X-Forwarded-Proto	75
X-Forwarded-Port	75
Configure Your Load Balancer	76
Configure the Idle Timeout	76
Configure the Idle Timeout Using the Console	77
Configure the Idle Timeout Using the AWS CLI	77
Configure Cross-Zone Load Balancing	77
Enable Cross-Zone Load Balancing	78
Disable Cross-Zone Load Balancing	79
Configure Connection Draining	80
Enable and Disable Connection Draining Using the Console	81
Enable and Disable Connection Draining Using the AWS CLI	81
Configure Proxy Protocol	83
Proxy Protocol Header	84
Prerequisites for Enabling Proxy Protocol	84
Enable Proxy Protocol Using the AWS CLI	84
Disable Proxy Protocol Using the AWS CLI	86
Configure Sticky Sessions	87
Duration-Based Session Stickiness	87
Application-Controlled Session Stickiness	89
Tag Your Load Balancer	91
Tag Restrictions	91
Add and Remove Tags Using the Console	92
Add and Remove Tags Using the AWS CLI	93

Configure the Domain Name	94
Associating Your Custom Domain Name with Your Load Balancer Name	94
Configure DNS Failover for Your Load Balancer	95
Disassociating Your Custom Domain Name from Your Load Balancer	96
Monitor Your Load Balancer	97
Monitor CloudWatch Metrics	97
CloudWatch Metrics for Elastic Load Balancing	98
Statistics for Elastic Load Balancing Metrics	100
View CloudWatch Metrics for Your Load Balancer	105
Create CloudWatch Alarms for Your Load Balancer	106
Log Load Balancer Requests	107
Access Log Files	107
Access Log Entries	108
Processing Access Logs	111
Enable Access Logs	111
Disable Access Logs	115
Log API Calls	116
Configure CloudTrail Event Logging	116
Elastic Load Balancing Event Entries in CloudTrail Log Files	116
Control Access to Your Load Balancer	119
Using an IAM Policy to Grant Permissions	119
Specifying Actions in an IAM Policy	120
Specifying Resources in an IAM Policy	121
Specifying Condition Keys in an IAM Policy	121
Example IAM Policies for Elastic Load Balancing	122
Troubleshoot Your Load Balancer	125
API Errors	126
CertificateNotFound: undefined	126
OutOfService: A Transient Error Occurred	127
HTTP Errors	127
HTTP 400: BAD_REQUEST	127
HTTP 405: METHOD_NOT_ALLOWED	128
HTTP 408: Request Timeout	128
HTTP 502: Bad Gateway	128
HTTP 503: Service Unavailable	128
HTTP 504: Gateway Timeout	128
Response Code Metrics	129
HTTPCode_ELB_4XX	129
HTTPCode_ELB_5XX	129
HTTPCode_Backend_2XX	130
HTTPCode_Backend_3XX	130
HTTPCode_Backend_4XX	130
HTTPCode_Backend_5XX	130
Health Check	130
Connection to the instances has timed out	131
Connection to your Internet-facing load balancer launched in a VPC has timed out	132
Health check target page error	132
Public key authentication is failing	132
Stopped and started instances failing load balancer health check	132
Instance is not receiving traffic from the load balancer	133
Ports on instance are not open	133
Instances in the Auto Scaling group are failing load balancer health check	133
Registering Instances	134
Taking too long to register back-end instances.	134
Unable to register instance launched from a paid AMI.	134
Limits	135
Command Line Interface	136
Task 1: Download the Command Line Interface	136

Task 2: Set the JAVA_HOME Environment Variable	136
Task 3: Set the AWS_ELB_HOME Environment Variable	138
Task 4: Set the AWS_CREDENTIAL_FILE Environment Variable	139
Task 6: Verify the Installation	140
Document History	142

What Is Elastic Load Balancing?

Elastic Load Balancing automatically distributes incoming traffic across multiple EC2 instances. You create a load balancer and register instances with the load balancer in one or more Availability Zones. The load balancer serves as a single point of contact for clients. This enables you to increase the availability of your application. You can add and remove EC2 instances from your load balancer as your needs change, without disrupting the overall flow of information. If an EC2 instance fails, Elastic Load Balancing automatically reroutes the traffic to the remaining running EC2 instances. If a failed EC2 instance is restored, Elastic Load Balancing restores the traffic to that instance. Elastic Load Balancing can also serve as the first line of defense against attacks on your network. You can offload the work of encryption and decryption to your load balancer so that your EC2 instances can focus on their main work.

For more information, see [Elastic Load Balancing](#).

Features of Elastic Load Balancing

Elastic Load Balancing provides the following features:

- You can use the operating systems and instance types supported by Amazon EC2. You can configure your EC2 instances to accept traffic only from your load balancer.
- You can configure the load balancer to accept traffic using the following protocols: HTTP, HTTPS (secure HTTP), TCP, and SSL (secure TCP).
- You can configure your load balancer to distribute requests to EC2 instances in multiple Availability Zones, minimizing the risk of overloading one single instance. If an entire Availability Zone goes offline, the load balancer routes traffic to instances in other Availability Zones.
- There is no limit on the number of connections that your load balancer can attempt to make with your EC2 instances. The number of connections scales with the number of concurrent requests that the load balancer receives.
- You can configure the health checks that Elastic Load Balancing uses to monitor the health of the EC2 instances registered with the load balancer so that it can send requests only to the healthy instances.
- You can use end-to-end traffic encryption on those networks that use secure (HTTPS/SSL) connections.
- [EC2-VPC] You can create an *Internet-facing* load balancer, which takes requests from clients over the Internet and routes them to EC2 instances in your public subnets, or an *internal-facing* load balancer, which takes requests from clients in your VPC and routes them to EC2 instances in your private subnets. Load balancers in EC2-Classic are always Internet-facing.
- [EC2-Classic] Load balancers for EC2-Classic support both IPv4 and IPv6 addresses. Load balancers for a VPC do not support IPv6 addresses.

- You can monitor your load balancer using CloudWatch metrics, access logs, and AWS CloudTrail.
- You can associate your Internet-facing load balancer with your domain name. Because the load balancer receives all requests from clients, you don't need to create and manage public domain names for the EC2 instances to which the load balancer routes traffic. You can point the instance's domain records at the load balancer instead and scale as needed (either adding or removing capacity) without having to update the records with each scaling activity.

How to Get Started with Elastic Load Balancing

- Before you explore Elastic Load Balancing, you must sign up for AWS and prepare to create your load balancer. For more information, see [Setting Up Elastic Load Balancing \(p. 5\)](#).
- To learn how to create a basic load balancer and register EC2 instances with your load balancer, see [Getting Started with Elastic Load Balancing \(p. 8\)](#).
- To learn how to create an HTTPS load balancer and register EC2 instances with your load balancer, see [Create an HTTPS Load Balancer \(p. 31\)](#).
- To learn how to use the various features supported by Elastic Load Balancing, see [Configure Your Load Balancer \(p. 76\)](#).

Related Services

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

- **Amazon EC2** — Virtual servers that run your applications in the cloud. For more information, see the [Amazon EC2 User Guide for Linux Instances](#) or the [Amazon EC2 User Guide for Microsoft Windows Instances](#).
- **Auto Scaling** — Ensures that you are running your desired number of instances, even if an instance fails, and enables you to automatically increase or decrease the number of instances as the demand on your instances changes. If you enable Auto Scaling with Elastic Load Balancing, instances that are launched by Auto Scaling are automatically registered with the load balancer, and instances that are terminated by Auto Scaling are automatically de-registered from the load balancer. For more information, see [Load Balance Your Auto Scaling Group](#) in the [Auto Scaling Developer Guide](#).
- **Amazon CloudWatch** — Enables you to monitor the health state of your instances and take action as needed. For more information, see the [Amazon CloudWatch Developer Guide](#).
- **Amazon Route 53** — Provides a reliable and cost-effective way to route visitors to websites by translating domain names (such as `www.example.com`) into the numeric IP addresses (such as `192.0.2.1`) that computers use to connect to each other. AWS assigns URLs to your AWS resources, such as your load balancers. However, you might want a URL that is easy for your users to remember. For example, you can map your domain name to your load balancer. If you don't have a domain name, you can search for available domains and register them using Amazon Route 53. If you have an existing domain name, you can transfer it to Amazon Route 53. For more information, see the [Amazon Route 53 Developer Guide](#).

Accessing Elastic Load Balancing

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

- **AWS Management Console**— Provides a web interface that you can use to access Elastic Load Balancing.

- **AWS Command Line Interface (CLI)** — Provides commands for a broad set of AWS services, including Elastic Load Balancing, and is supported on Windows, Mac, and Linux. For more information, see [AWS Command Line Interface](#).
- **AWS SDKs** — Provides language-specific APIs and takes care of many of the connection details, such as calculating signatures, handling request retries, and error handling. For more information, see [AWS SDKs](#).
- **Query API** — Provides low-level APIs that you call using HTTPS requests. Using the Query API is the most direct way to access Elastic Load Balancing, but it requires that your application handle low-level details such as generating the hash to sign the request, and error handling. For more information, see the [Elastic Load Balancing API Reference](#).
- **SOAP API** — Provides access to the Elastic Load Balancing web service using the SOAP web services messaging protocol. This interface is described by a Web Services Description Language (WSDL) document that defines the operations and security model. The WSDL references an XML schema document, which strictly defines the data types that might appear in SOAP requests and responses. For more information, see the [Elastic Load Balancing API Reference](#).
- **Elastic Load Balancing Command Line Interface (CLI)** — Provides commands to access Elastic Load Balancing.

Important

We are no longer adding new functionality to the ELB CLI. We recommend that you use the AWS CLI instead.

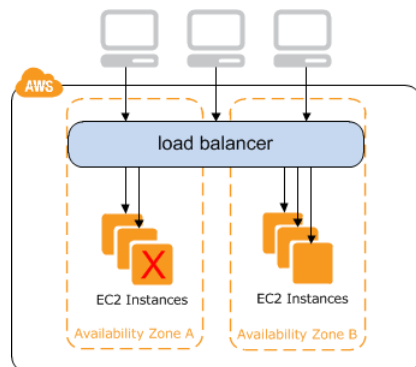
Pricing

With Amazon Web Services, you pay only for what you use. For Elastic Load Balancing, you pay for each hour or portion of an hour that the service is running, and you pay for each gigabyte of data that is transferred through your load balancer. For current pricing information, see [Elastic Load Balancing Pricing](#).

If your AWS account is less than 12 months old, you are eligible to use the free tier. The free tier includes 750 hours per month of Amazon EC2 usage, and 750 hours per month of Elastic Load Balancing, plus 15 GB of data processing. For more information, see [AWS Free Tier](#).

How Elastic Load Balancing Works

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



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

When you attach an Availability Zone to your load balancer, Elastic Load Balancing creates a load balancer node in the Availability Zone that forwards traffic to the healthy registered instances in that Availability Zone. We recommend that you configure your load balancer across multiple Availability Zones. If one Availability Zone becomes unavailable or has no healthy instances, the load balancer can route traffic to the healthy registered instances in another Availability Zone.

Request Routing

Before a client sends a request to your load balancer, it resolves the load balancer's domain name using a Domain Name System (DNS) server. The DNS entry is controlled by Amazon, because your instances are in the `amazonaws.com` domain. The Amazon DNS servers return one or more IP addresses to the client. These are the IP addresses of the load balancer nodes for your load balancer. The client uses DNS round robin to determine which IP address to use to send the request to the load balancer.

The load balancer node that receives the request uses a routing algorithm to select a healthy instance. It uses the round robin routing algorithm for TCP listeners, and the least outstanding requests routing algorithm (favors the instances with the fewest outstanding requests) for HTTP and HTTPS listeners.

The *cross-zone load balancing* setting also determines how the load balancer selects an instance. If cross-zone load balancing is disabled, the load balancer node selects the instance from the same Availability Zone that it is in. If cross-zone load balancing is enabled, the load balancer node selects the instance regardless of Availability Zone. The load balancer node routes the client request to the selected instance.

Availability Zones and Instances

To ensure that your back-end 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 traffic is equally distributed 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 distribute your instances so that you have six instances in each Availability Zone.

To distribute traffic evenly across all back-end instances, regardless of the Availability Zone, 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.

Related Topics

For more information about Elastic Load Balancing, see the following documentation:

- [Internet-Facing Load Balancers](#) (p. 13)
- [Internal Load Balancers](#) (p. 15)
- [Listeners for Your Load Balancer](#) (p. 70)
- [Configure Health Checks](#) (p. 54)
- [Configure Your Load Balancer](#) (p. 76)
- [Monitor Your Load Balancer](#) (p. 97)
- [Troubleshoot Your Load Balancer](#) (p. 125)

Setting Up Elastic Load Balancing

Complete the following tasks to get ready to use Elastic Load Balancing.

Tasks

- [Sign Up for Amazon Web Services \(AWS\)](#) (p. 5)
- [Prepare Your VPC and Back-end Instances](#) (p. 5)
- [Access Elastic Load Balancing Using the AWS Management Console](#) (p. 7)
- [Access Elastic Load Balancing Using the AWS CLI](#) (p. 7)

Sign Up for Amazon Web Services (AWS)

When you sign up for AWS, we automatically sign up your AWS account for all services, including Elastic Load Balancing. You pay only for the services that you use.

If you already have an AWS account, skip to the next task. If you don't have an AWS account, use the following procedure to create one.

To sign up for an AWS account

1. Open <http://aws.amazon.com/>, and then click **Sign Up**.
2. Follow the on-screen instructions.

Part of the sign-up procedure involves receiving a phone call and entering a PIN using the phone keypad.

Prepare Your VPC and Back-end Instances

If you haven't used Amazon EC2 before, complete the tasks described in the Amazon EC2 documentation. For more information, see [Setting Up with Amazon EC2](#) in the *Amazon EC2 User Guide for Linux Instances* or [Setting Up with Amazon EC2](#) in the *Amazon EC2 User Guide for Microsoft Windows Instances*, depending on which operating system you plan to use for your EC2 instances.

Load Balancers in a VPC

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

We recommend that you launch your instances and create your load balancer in a VPC. If you have a new AWS account or plan to use a region that you haven't used before, you have a default VPC. You can use a default VPC if you have one, or create your own VPC.

Each default VPC has one public subnet per Availability Zone. If you create your own VPC, create a subnet in each Availability Zone where you want to launch instances. Depending on your application, you can create public subnets, private subnets, or a combination of public and private subnets.

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 back-end instances.

Security Groups

You must ensure that the load balancer can communicate with your back-end instances on both the listener port and the health check port. The security group for your instances must allow traffic in both directions on both ports for each subnet attached to your load balancer. For more information, see [Configure Security Groups for Your Load Balancer \(p. 57\)](#).

Network ACLs

The network ACLs for your VPC must allow traffic in both directions on the listener port and the health check port for each subnet attached to your load balancer. For example, the network ACLs must include the rules described in the following table, and must not include an inbound DENY rule for all traffic with a source of 0.0.0.0/0.

Inbound			
Source	Port	Allow/Deny	Comments
CIDR for subnet 1	<i>listener</i>	ALLOW	Allow inbound traffic from subnet 1 on the listener port.
CIDR for subnet 2	<i>listener</i>	ALLOW	Allow inbound traffic from subnet 2 on the listener port.
CIDR for subnet 1	<i>health check</i>	ALLOW	Allow inbound traffic from subnet 1 on the health check port.
CIDR for subnet 2	<i>health check</i>	ALLOW	Allow inbound traffic from subnet 2 on the health check port.
Outbound			
Destination	Port	Allow/Deny	Comments
CIDR for subnet 1	1024-65535	ALLOW	Allow outbound traffic from subnet 1 on the ephemeral ports.
CIDR for subnet 2	1024-65535	ALLOW	Allow outbound traffic from subnet 2 on the ephemeral ports.

ClassicLink

ClassicLink enables your EC2-Classic instances to communicate with VPC instances using private IP addresses, provided that the VPC security groups allow it. If you plan to register linked EC2-Classic instances with your load balancer, you must enable ClassicLink for your VPC, and then create your load balancer in the ClassicLink-enabled VPC. For more information, see [ClassicLink Basics](#) and [Working with ClassicLink](#) in the *Amazon EC2 User Guide for Linux Instances*.

Access Elastic Load Balancing Using the AWS Management Console

The AWS Management Console is a point-and-click web-based interface you can use to access Elastic Load Balancing and other AWS products. You can access Elastic Load Balancing resources using the Amazon EC2 console.

To access Elastic Load Balancing using the Amazon EC2 console

1. Sign in to the AWS Management Console and open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. From the navigation bar, select a region. Each load balancer is tied to the region in which you create it. You can select any region that's available to you, regardless of your location.



3. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.

To learn how to create a load balancer using the AWS Management Console, see [Getting Started with Elastic Load Balancing \(p. 8\)](#).

Access Elastic Load Balancing Using the AWS CLI

The AWS CLI provides commands for a broad set of AWS products, including Elastic Load Balancing, and is supported on Windows, Mac, and Linux.

To get started using the AWS CLI, see the [AWS Command Line Interface User Guide](#). For details about the commands for Elastic Load Balancing, see [AWS CLI elb](#).

Getting Started with Elastic Load Balancing

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

Note that you can create your load balancer for use with EC2-Classic or a VPC. Some of the tasks described in this tutorial apply only to load balancers in a VPC.

Tasks

- [Before You Begin](#) (p. 8)
- [Step 1: Define Your Load Balancer](#) (p. 9)
- [Step 2: Assign Security Groups to Your Load Balancer in a VPC](#) (p. 10)
- [Step 3: Configure Security Settings \(Optional\)](#) (p. 10)
- [Step 4: Configure Health Checks for Your EC2 Instances](#) (p. 11)
- [Step 5: Register EC2 Instances with Your Load Balancer](#) (p. 11)
- [Step 6: Tag Your Load Balancer \(Optional\)](#) (p. 36)
- [Step 7: Create and Verify Your Load Balancer](#) (p. 12)
- [Step 8: Delete Your Load Balancer \(Optional\)](#) (p. 12)

Before You Begin

- Complete the steps in [Setting Up Elastic Load Balancing](#) (p. 5).
- Launch the EC2 instances that you plan to register with your load balancer in the Availability Zones that you plan to use for your load balancer. Ensure that the security groups for these instances allow HTTP access on port 80.
- If your EC2 instances run in a VPC, be sure to launch them in public subnets. Note that by definition, the route table for a public subnet must have a route to the Internet gateway for the VPC.
- Install a web server, such as Apache or Internet Information Services (IIS), on the EC2 instances that you plan to register with your load balancer.

Step 1: Define Your Load Balancer

First, provide some basic configuration information for your load balancer, such as a name, a network, and a listener.

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. In this tutorial, you configure a listener that accepts HTTP requests on port 80 and sends them to the back-end instances on port 80 using HTTP.

To define your load balancer

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. From the navigation bar, select a region for your load balancers. Be sure to select the same region that you selected for your EC2 instances.
3. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.
4. Click **Create Load Balancer**.
5. In **Load Balancer name**, enter a name for your load balancer.

The name of your load balancer must be unique within your set of load balancers, can have a maximum of 32 characters, and can contain only alphanumeric characters and hyphens.

6. From **Create LB inside**, select the same network that you selected for your instances: EC2-Classic or a specific VPC.
7. [Default VPC] If you selected a default VPC and would like to choose the subnets for your load balancer, select **Enable advanced VPC configuration**.
8. Leave the default listener configuration.

Load Balancer name:

Create LB inside:

Create an internal load balancer: ☐ (what's this?)

Enable advanced VPC configuration: ☒

Listener Configuration:

Load Balancer Protocol	Load Balancer Port	Instance Protocol	Instance Port
HTTP	80	HTTP	80

Add

9. [EC2-VPC] Under **Select Subnets**, select at least one available subnet. The available subnets for the VPC for your load balancer are displayed under **Available Subnets**. Click the icon in the **Action** column for each subnet to attach. These subnets are moved under **Selected Subnets**.

Note

If you selected EC2-Classic as your network, or you have a default VPC but did not select **Enable advanced VPC configuration**, you do not see **Select Subnets**.

You can select at most one subnet per Availability Zone. If you select a second subnet in an Availability Zone, it replaces the previously selected subnet for that Availability Zone. To improve the availability of your load balancer, select subnets from more than one Availability Zone.

Elastic Load Balancing Developer Guide

Step 2: Assign Security Groups to Your Load Balancer in a VPC

Available Subnets				
Actions	Availability Zone	Subnet ID	Subnet CIDR	Name
	us-west-2c	subnet-cb663da2	10.0.1.0/24	
	us-west-2c	subnet-c9663da0	10.0.0.0/24	
Selected Subnets				
Actions	Availability Zone	Subnet ID	Subnet CIDR	Name
	us-west-2a	subnet-e4f33493	10.0.2.0/24	
	us-west-2b	subnet-5264e837	10.0.3.0/24	

10. Click **Next: Assign Security Groups**.

Step 2: Assign Security Groups to Your Load Balancer in a VPC

If you selected a VPC as your network, you must assign your load balancer a security group that allows inbound traffic to the ports that you specified for your load balancer and the health checks for your load balancer.

Note

If you selected EC2-Classic as your network, you can continue to the next step. By default, Elastic Load Balancing provides a security group for load balancers in EC2-Classic.

To assign security group to your load balancer

1. On the **Assign Security Groups** page, select **Create a new security group**.
2. Enter a name and description for your security group, or leave the default name and description. This new security group contains a rule that allows traffic to the port that you configured your load balancer to use.

Assign a security group: ☒ Create a new security group
☐ Select an existing security group

Security group name:

Description:

Type	Protocol	Port Range	Source
HTTP	TCP	80	Anywhere 0.0.0.0/0

3. Click **Next: Configure Security Settings**.

Step 3: Configure Security Settings (Optional)

For this tutorial, you can click **Next: Configure Health Check** to continue to the next step. For more information about creating a HTTPS load balancer and using additional security features, see [HTTPS Load Balancers \(p. 20\)](#).

Step 4: Configure Health Checks for Your EC2 Instances

Elastic Load Balancing automatically checks the health of the EC2 instances for your load balancer. If Elastic Load Balancing finds an unhealthy instance, it stops sending traffic to the instance and reroutes traffic to healthy instances. In this step, you customize the health checks for your load balancer.

To configure health checks for your instances

1. On the **Configure Health Check** page, do the following:
 - a. Leave **Ping Protocol** set to its default value, `HTTP`.
 - b. Leave **Ping Port** set to its default value, `80`.
 - c. In the **Ping Path** field, replace the default value with a single forward slash (`/`). This tells Elastic Load Balancing to send health check queries to the default home page for your web server, such as `index.html` or `default.html`.



The screenshot shows three input fields for configuring health checks. The 'Ping Protocol' is a dropdown menu with 'HTTP' selected. The 'Ping Port' is a text box containing '80'. The 'Ping Path' is a text box containing a single forward slash '/'. Each field has a small 'x' icon to clear the input.

- d. Leave the other fields set to their default values.
2. Click **Next: Add EC2 Instances**.

Step 5: Register EC2 Instances with Your Load Balancer

Your load balancer distributes traffic between the instances that are registered to it.

Note

When you register an instance with an elastic network interface (ENI) attached, the load balancer routes traffic to the primary IP address of the primary interface (eth0) of the instance.

To register EC2 instances with your load balancer

1. On the **Add EC2 Instances** page, select the instances to register with your load balancer.
2. Click **Next: Add Tags**.

Alternatively, you can register instances with your load balancer later on using the following options:

- Select running instances after you create the load balancer. For more information, see [Register Instances with Your Load Balancer \(p. 67\)](#).
- Set up Auto Scaling to register the instances automatically when it launches them. For more information, see [Set Up a Scaled and Load-Balanced Application](#) in the *Auto Scaling Developer Guide*.

Step 6: Tag Your Load Balancer (Optional)

You can tag your load balancer, or continue to the next step. Note that you can tag your load balancer later on; for more information, see [Tag Your Load Balancer \(p. 91\)](#).

To add tags to your load balancer

1. On the **Add Tags** page, specify a key and a value for the tag.
2. To add another tag, click **Create Tag** and specify a key and a value for the tag.
3. After you are finished adding tags, click **Review and Create**.

Step 7: Create and Verify Your Load Balancer

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

To finish creating your load balancer

1. On the **Review** page, check your settings. If you need to make changes, click the corresponding link to edit the settings.
2. Click **Create** to create your load balancer.
3. After you are notified that your load balancer was created, click **Close**.
4. Select your new load balancer.
5. In the bottom pane, on the **Description** tab, check the **Status** row. If it indicates that some of your instances are not in service, it's probably because they are still in the registration process. For more information, see [Troubleshooting Elastic Load Balancing: Registering Instances \(p. 134\)](#).
6. (Optional) After you've verified that at least one of your EC2 instances is `InService`, you can test your load balancer. Copy the string from the **DNS Name** field and paste it into the address field of an Internet-connected web browser. (For example, `my-load-balancer-1234567890.us-west-2.elb.amazonaws.com`.) If your load balancer is working, you see the default page of your HTTP server.

Step 8: Delete Your Load Balancer (Optional)

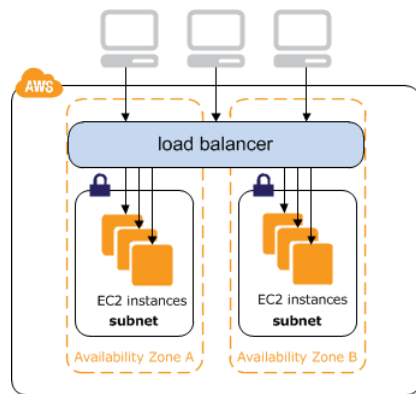
As soon as your load balancer becomes available, you are billed for each hour or partial hour that you keep it running. When you no longer need a load balancer, you can delete it. As soon as the load balancer is deleted, you stop incurring charges for it.

To delete your load balancer

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.
3. Select the load balancer.
4. Click **Actions**, and then click **Delete**.
5. When prompted for confirmation, click **Yes, Delete**.
6. (Optional) After you delete a load balancer, the EC2 instances associated with the load balancer continue to run, and you are billed for each hour or partial hour that you keep them running. For information about stopping or terminating your instances, see [Stop and Start Your Instance](#) or [Terminate Your Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

Internet-Facing Load Balancers

An Internet-facing load balancer takes requests from clients over the Internet and distributes them across the EC2 instances that are registered with the load balancer.



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

Contents

- [Public DNS Names for Your Load Balancer](#) (p. 13)
- [Create an Internet-Facing Load Balancer](#) (p. 14)

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.

EC2-VPC

Load balancers in a VPC support IPv4 addresses only. They receive a public DNS name with the following form:

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

EC2-Classic

Load balancers in EC2-Classic support both IPv4 and IPv6 addresses. They receive the following public DNS names:

```
name-123456789.region.elb.amazonaws.com  
ipv6.name-123456789.region.elb.amazonaws.com  
dualstack.name-123456789.region.elb.amazonaws.com
```

The base public DNS name returns only IPv4 records. The public DNS name with the `ipv6` prefix returns only IPv6 records. The public DNS name with the `dualstack` prefix returns both IPv4 and IPv6 records. We recommend that you enable IPv6 support by using the DNS name with the `dualstack` prefix to ensure that clients can access the load balancer using either IPv4 or IPv6.

Clients can connect to your load balancer in EC2-Classic using either IPv4 or IPv6. However, communication between the load balancer and its back-end instances uses only IPv4, regardless of how the client communicates with your load balancer.

Create an Internet-Facing Load Balancer

When you create a load balancer in a VPC, you can make it an internal load balancer or an Internet-facing load balancer. You create an Internet-facing load balancer in a public subnet. Load balancers in EC2-Classic are always Internet-facing load balancers.

When you create your 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.

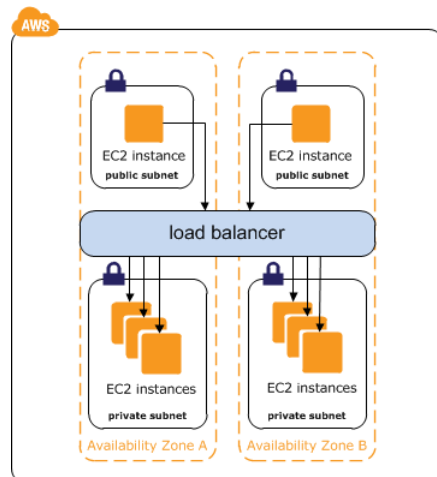
To create a basic Internet-facing load balancer, see [Getting Started with Elastic Load Balancing \(p. 8\)](#).

To create an HTTPS load balancer, see [Create an HTTPS Load Balancer \(p. 31\)](#).

Internal Load Balancers

When you create a load balancer in a VPC, you can make it an internal load balancer or an Internet-facing load balancer.

An internal load balancer routes traffic to your EC2 instances in private subnets. The clients must have access to the private subnets.



An Internet-facing load balancer in a VPC takes requests from clients over the Internet and distributes them across EC2 instances in your public subnets. For more information, see [Internet-Facing Load Balancers](#) (p. 13).

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](#) (p. 16)
- [Create an Internal Load Balancer](#) (p. 16)

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 that are in its Availability Zone using elastic network interfaces.

Create an Internal Load Balancer

You can create an internal load balancer to distribute traffic to your EC2 instances in private subnets.

Contents

- [Prerequisites \(p. 16\)](#)
- [Create an Internal Load Balancer Using the Console \(p. 16\)](#)
- [Create an Internal Load Balancer Using the AWS CLI \(p. 17\)](#)

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 [Prepare Your VPC and Back-end Instances \(p. 5\)](#).
- Launch the EC2 instances that you plan to register with your internal load balancer. Ensure that you launch them in a private subnet in the VPC intended for the load balancer.

Create an Internal Load Balancer Using the Console

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. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.
3. Click **Create Load Balancer**.
4. On the **Define Load Balancer** page, do the following:
 - a. In **Load Balancer name**, enter a name for your load balancer.

The name of your load balancer must be unique within your set of load balancers, can have a maximum of 32 characters, and can contain only alphanumeric characters and hyphens.
 - b. From **Create LB inside**, select a VPC for your load balancer.
 - c. Click **Create an internal load balancer**.
 - d. [Default VPC] If you selected a default VPC and would like to choose the subnets for your load balancer, select **Enable advanced VPC configuration**.

- e. Leave the default listener configuration.
- f. Under **Select Subnets**, select at least one available subnet. The available subnets for the VPC for your load balancer are displayed under **Available Subnets**. Click the icon in the **Action** column for each subnet to attach. These subnets are moved under **Selected Subnets**.

Note

If you select a default VPC as your network, but did not select **Enable advanced VPC configuration**, you do not see **Select Subnets**.

You can select at most one subnet per Availability Zone. If you select a second subnet in an Availability Zone, it replaces the previously selected subnet for that Availability Zone. To improve the availability of your load balancer, select subnets from more than one Availability Zone.

- g. Click **Next: Assign Security Groups**.
5. On the **Assign Security Groups** page, click **Create a new security group**. Enter a name and description for your security group, or leave the default name and description. This new security group contains a rule that allows traffic to the port that you configured your load balancer to use. If you specified a different port for the health checks, you must click **Add Rule** to add a rule that allows inbound traffic to that port as well. Click **Next: Configure Security Settings**.
6. On the **Configure Security Settings** page, click **Next: Configure Health Check** to continue to the next step. If you prefer to create a HTTPS load balancer, see [HTTPS Load Balancers \(p. 20\)](#).
7. On the **Configure Health Check** page, configure the health check settings that your application requires, and then click **Next: Add EC2 Instances**.
8. On the **Add EC2 Instances** page, select the instances to register with your load balancer, and then click **Next: Add Tags**.

Note

When you register an instance with an elastic network interface (ENI) attached, the load balancer routes traffic to the primary IP address of the primary interface (eth0) of the instance.

9. (Optional) You can add tags to your load balancer. When you are finished adding tags, click **Review and Create**.
10. On the **Review** page, check your settings. If you need to make changes, click the corresponding link to edit the settings. When you are finished, click **Create** to create your load balancer.
11. After you are notified that your load balancer was created, click **Close**.
12. Select your new load balancer.
13. In the bottom pane, on the **Description** tab, note that **DNS Name** and **Scheme** indicate that the load balancer is `internal`.

Check the **Status** row. If it indicates that some of your instances are not in service, its probably because they are still in the registration process. For more information, see [Troubleshooting Elastic Load Balancing: Registering Instances \(p. 134\)](#).

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",  
    ...  
  }  
]  
}
```

HTTPS Load Balancers

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 back-end instances.

Elastic Load Balancing provides Secure Sockets Layer (SSL) negotiation configurations, known as *security policy*, to negotiate connections between the clients and the load balancer. When you use HTTPS/SSL for your front-end connection, you can use either a predefined security policy or a custom security policy. You must install 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 back-end instances. You can optionally choose to enable authentication on your back-end instances.

Contents

- [SSL Certificates for Elastic Load Balancing \(p. 20\)](#)
- [SSL Negotiation Configurations for Elastic Load Balancing \(p. 26\)](#)
- [Create an HTTPS Load Balancer \(p. 31\)](#)
- [Configure an HTTPS Listener for Your Load Balancer \(p. 45\)](#)
- [Update the SSL Certificate for Your Load Balancer \(p. 48\)](#)
- [Update the SSL Negotiation Configuration of Your Load Balancer \(p. 50\)](#)

SSL Certificates for Elastic Load Balancing

If you use HTTPS or SSL for your front-end listener, you must install 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 back-end instances.

The SSL protocol uses an X.509 certificate (SSL 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.

Before you can install an SSL certificate on your load balancer, you must create the certificate, get the certificate signed by a CA, and then upload the certificate using the AWS Identity and Access Management (IAM) service, which manages your SSL certificates. By default, IAM allows 20 server certificates per AWS account. For more information about this limit and how to request an increase, see [Limitations on IAM Entities](#) in *Using IAM*.

Each SSL certificate has a validity period. You must replace a certificate before its validity period ends. To replace a certificate, you must create and upload a new certificate.

Complete the following tasks to create an SSL server certificate, upload it to IAM, and install it on your load balancer.

Tasks

- [Prerequisite: Install Certificate Tools \(p. 21\)](#)
- [Create a Server Certificate \(p. 21\)](#)
- [Upload the Signed Certificate \(p. 22\)](#)
- [Verify Server Certificate \(p. 25\)](#)
- [Install the SSL Certificate on Your Load Balancer \(p. 25\)](#)

Prerequisite: Install Certificate Tools

Creating a server certificate requires a tool that supports the SSL and TLS protocols.

Linux

OpenSSL is an open-source tool that provides extensive cryptographic functions. To check whether this tool is already installed, run `openssl version`. If OpenSSL is not installed, you must install it. For more information, see the documentation for your Linux distribution.

Windows

There are several tools that you can use, such as IIS Manager, SelfSSL, OpenSSL, Windows PowerShell cmdlets. If you don't have one of these tools already, select a tool and install it.

Create a Server Certificate

To create an SSL server certificate, you must generate an RSA private key and create a Certificate Signing Request (CSR). Next, either have your certificate signed by a Certificate Authority (CA), or generate a self-signed certificate so that you can test your SSL implementation while waiting for the CA to sign your certificate. Follow the directions for the tool that you are using. In this example, we provide directions for OpenSSL, and pointers to directions for IIS Manager.

To create a server certificate using OpenSSL

1. Create a private key and save it in a secure place, as there is no way to get your private key if you lose it.

Private keys are created using standard key algorithms. Choose the algorithm based on the ciphers that you plan to use when negotiating SSL connections from the client to your load balancer.

RSA-based Ciphers

Use the following `genrsa` command. Note that AWS supports RSA keys that are 1024, 2048, and 4096 bits. However, we recommend that you specify 2048 bits.

```
openssl genrsa -out my-private-key.pem 2048
```

ECDHE-ECDSA-based Ciphers

Use the following `ecparam` command:

```
openssl ecparam -name prime256v1 -out my-private-key.pem -genkey
```

2. Create a CSR using the following `req` command:

```
openssl req -sha256 -new -key my-private-key.pem -out csr.pem
```

The command runs interactively, prompting you to enter the following information:

Country Name

The two-letter [ISO code](#) for your country. For example, US.

State or Province Name

The full name of the state or province where your organization is located. Do not use an abbreviation.

Locality Name

The name of the city where your organization is located.

Organization Name

The full legal name of your organization.

Organizational Unit Name

(Optional) Additional information, such as a product name or division.

Common Name

The fully-qualified domain name for your CNAME. This name must be an exact match. For example, `www.mycompany.com`, `mycompany.com`, or `*.mycompany.com`.

Email Address

The server administrator's email address.

3. You can either apply to have your certificate signed by a CA, or generate a self-signed certificate to use for testing purposes.
 - To apply for a server certificate, send your CSR to an CA. Your CSR contains information that identifies you. The CA might require other credentials or proof of identity. Upon success, the CA returns a public (identity) certificate and possibly a chain certificate that is digitally signed. AWS does not recommend a specific CA. For a partial listing of available CAs, see [Third-Party Certificate Authorities](#).
 - To create a self-signed certificate, use the following command:

```
openssl x509 -req -days 365 -in csr.pem -signkey my-private-key.pem -out my-certificate.pem
```

To create a server certificate using IIS Manager

For information about using IIS Manager to create a certificate, see [Request an Internet Server Certificate](#) or [Create a Self-Signed Server Certificate](#) in the Microsoft TechNet Library.

Upload the Signed Certificate

Typically the certificate authority (CA) sends you a public certificate, one or more intermediate certificates, and a root certificate. The intermediate certificates and the root certificate can come bundled in a file or as separate files. The file names may vary depending on the type of SSL certificate you purchase and the certificate authority. Your public certificate is the domain-specific file.

Prerequisites

- The private key must be created using the algorithm based on the ciphers you plan to use for negotiating SSL connections and must be in PEM format.

RSA-based Ciphers

Use the following command to convert a private key generated for RSA based ciphers:

```
openssl rsa -in my-private-key -outform PEM
```

ECDSA-based Ciphers

Use the following command to convert a private key generated for ECDSA based ciphers:

```
openssl ecparam -in my-private-key -outform PEM
```

- The private key cannot be encrypted with a password.
- When you receive your server certificate from the CA, the files might not be in the PEM format that is required by IAM. For more information about the PEM format, see [pem DESCRIPTION](#).
- Use the following command to convert the server certificate you received from the CA:

```
openssl x509 -inform PEM -in my-certificate
```

- Use the following command to convert your certificate chain:

```
openssl x509 -inform PEM -in my-certificate-chain
```

- The current date must be between the certificate's start and end dates.
- The public and private certificate files must contain a single certificate.
- The private key must match the public key in the certificate.
- The certificate chain must include all of your CA's intermediary certificates that lead to the root certificate, and can optionally end with your CA's root certificate. Typically, both intermediate and root certificates are provided by the CA in a bundled file with the proper chained order. The order of intermediate certificates should be documented by the CA. Although the root certificate is optional, you can include it so that you can run a full chain of trust verification, such as [SSL Checker](#).

If a certificate bundle is not available or not available in the required order, you can create your own certificate chain file using the intermediary certificates, as shown in [Example RSA Private Key \(p. 24\)](#).

Uploading the Server Certificate

After you have your certificate files in PEM format, use the following `upload-server-certificate` command to upload them:

```
aws iam upload-server-certificate --server-certificate-name my-server-cert  
--certificate-body file://my-certificate.pem --private-key file://my-private-  
key.pem  
--certificate-chain file://my-certificate-chain.pem
```

Note that if you are uploading a self-signed certificate, you do not need to specify a certificate chain.

The following are examples of the server certificates, private keys, and certificate chains accepted by IAM.

The server certificate associates your public key with your identity. When you submit your CSR to a certificate authority (CA), the CA returns a server certificate.

-----END CERTIFICATE-----

The private key enables you to decrypt messages that are encrypted with your public key.

rDHudUZq3qX4waLG5M43q7Wqc/MbQITxOUSQv7c7uqFFDzQGBzZswY6786m86qpE

```
Ibb3OhjZnzcVQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJilJ00zbbNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvJx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END RSA PRIVATE KEY-----
```

Example Certificate Chain

The certificate chain enables a browser to build a certificate chain to a root certificate that it trusts. As a result, the browser can implicitly trust your certificate.

The certificate chain includes the intermediate certificates and optionally the root certificate, one after the other without any blank lines, as shown in the following example. If you include the root certificate, your certificate chain must start with intermediate certificates and end with the root certificate. Use the intermediate certificates that were provided by your CA. Do not include any intermediaries that are not in the chain of trust path.

```
-----BEGIN CERTIFICATE-----
Intermediate certificate 2
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
Intermediate certificate 1
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
Optional: Root certificate
-----END CERTIFICATE-----
```

Verify Server Certificate

After the server certificate is uploaded, you can verify that the information is stored in IAM. Each certificate object has a unique Amazon Resource Name (ARN) and ID.

Use the following [get-server-certificate](#) command to verify the certificate object:

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

The following is an example response. The first line is the ARN of the server certificate and the second line is the ID.

```
arn:aws:iam::123456789012:server-certificate/my-server-certificate
ASCACexampleKEZUQ4K
```

Install the SSL Certificate on Your Load Balancer

To install an SSL certificate, see [Configure Listeners](#) (p. 32).

To update an SSL certificate, see [Update the SSL Certificate for Your Load Balancer](#) (p. 48).

SSL Negotiation Configurations for Elastic Load Balancing

Elastic Load Balancing uses an 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 Load Balancer \(p. 70\)](#).

SSL Protocols

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.

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.

Elastic Load Balancing supports the following versions of the SSL protocol:

- TLS 1.2
- TLS 1.1
- TLS 1.0
- SSL 3.0
- SSL 2.0

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. For information about the SSL ciphers and SSL protocols supported by Elastic Load Balancing, see [Predefined SSL Security Policies \(p. 27\)](#).

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.

For information about the order of ciphers used by Elastic Load Balancing, see [Predefined SSL Security Policies \(p. 27\)](#).

Contents

- [SSL Security Policies for Elastic Load Balancing \(p. 27\)](#)
- [Predefined SSL Security Policies for Elastic Load Balancing \(p. 27\)](#)

SSL Security Policies for Elastic Load Balancing

A security policy determines which ciphers and protocols are supported during SSL negotiations between a client and a load balancer. Elastic Load Balancing supports configuring your load balancer to use either predefined or custom security policies.

Predefined Security Policies

You can use the predefined SSL negotiation configurations provided by Elastic Load Balancing. The naming convention of a predefined security policy includes version information based on the year and month that it was released. For example, the current predefined security policy is `ELBSecurityPolicy-2015-05`. Note that some of the older predefined security policies do not follow this naming convention.

Whenever a new predefined security policy is released, you can update your configuration to use the current version.

Elastic Load Balancing provides the following predefined security policies:

- `ELBSecurityPolicy-2015-05`
- `ELBSecurityPolicy-2015-03`
- `ELBSecurityPolicy-2015-02`
- `ELBSecurityPolicy-2014-10`
- `ELBSecurityPolicy-2014-01`
- `ELBSecurityPolicy-2011-08`
- `ELBSample-ELBDefaultNegotiationPolicy` or `ELBSample-ELBDefaultCipherPolicy`
- `ELBSample-OpenSSLDefaultNegotiationPolicy` or `ELBSample-OpenSSLDefaultCipherPolicy`

For information about the SSL protocols, SSL ciphers, and SSL options enabled for the predefined security policies, see [Predefined SSL Security Policies \(p. 27\)](#).

Custom Security Policy

You can create a custom negotiation configuration with the ciphers and protocols that you need. Note that some security compliance standards (such as PCI, SOX, and so on) 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 the protocols and ciphers supported by Elastic Load Balancing, see [Predefined SSL Security Policies \(p. 27\)](#).

Predefined SSL Security Policies for Elastic Load Balancing

The following table describes the most recent predefined security policies, including their enabled SSL protocols and SSL ciphers. 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 in this table 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.

We recommend that you always use the current predefined security policy.

To describe all predefined policies, including the deprecated ones, use the `describe-load-balancer-policies` command or the `DescribeLoadBalancerPolicies` action.

Elastic Load Balancing Developer Guide
Predefined SSL Security Policies

Security Policy	2015-05	2015-03	2015-02	2014-10	2014-01	2011-08
SSL Protocols						
Protocol-SSLv2						
Protocol-SSLv3					◆	◆
Protocol-TLSv1	◆	◆	◆	◆	◆	◆
Protocol-TLSv1.1	◆	◆	◆	◆	◆	
Protocol-TLSv1.2	◆	◆	◆	◆	◆	
SSL Options						
Server Order Preference	◆	◆	◆	◆	◆	
SSL 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	◆	◆				◆

Elastic Load Balancing Developer Guide
Predefined SSL Security Policies

Security Policy	2015-05	2015-03	2015-02	2014-10	2014-01	2011-08
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						

Elastic Load Balancing Developer Guide
Predefined SSL Security Policies

Security Policy	2015-05	2015-03	2015-02	2014-10	2014-01	2011-08
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						
Deprecated SSL Ciphers						
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						

Security Policy	2015-05	2015-03	2015-02	2014-10	2014-01	2011-08
EXP-KRB5-RC4-SHA						
EXP-KRB5-RC4-MD5						

Deprecated SSL Ciphers: If you had previously enabled these ciphers in a custom policy or the `ELBSample-OpenSSLDefaultCipherPolicy`, we recommend that you update your security policy to the current predefined security policy.

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.

For more information about updating the SSL negotiation configuration for your HTTPS/SSL listener, see [Update the SSL Negotiation Configuration of Your Load Balancer \(p. 50\)](#).

Create an HTTPS Load Balancer

You can create an HTTPS load balancer with SSL negotiation configurations and with back-end application instance authentication.

For information about adding an HTTPS listener to an existing load balancer, see [Configure an HTTPS Listener for Your Load Balancer \(p. 45\)](#).

Contents

- [Prerequisites \(p. 31\)](#)
- [Create an HTTPS/SSL Load Balancer Using the Console \(p. 32\)](#)
- [Create an HTTPS/SSL Load Balancer Using the AWS CLI \(p. 37\)](#)

Prerequisites

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

- Complete the steps in [Setting Up Elastic Load Balancing \(p. 5\)](#).
- Launch the EC2 instances that you plan to register with your load balancer in the Availability Zones you plan to use for your load balancer. These instances must be configured to receive requests 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 [Configure Health Checks \(p. 54\)](#).
- 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 back-end 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 Load Balancer \(p. 76\)](#).
- To enable HTTPS support for our listeners, you must create an SSL certificate and then install it on your load balancer. The load balancer uses the certificate to terminate and then decrypt requests before sending them to the back-end instances. For more information, see [SSL Certificates for Elastic Load Balancing \(p. 20\)](#).

Create an HTTPS/SSL Load Balancer Using the Console

To create an HTTPS/SSL load balancer, complete the following tasks.

Tasks

- [Step 1: Define Your Load Balancer \(p. 32\)](#)
- [Step 2: Assign Security Groups to Your Load Balancer in a VPC \(p. 33\)](#)
- [Step 3: Configure Security Settings \(p. 34\)](#)
- [Step 4: Configure Health Checks \(p. 35\)](#)
- [Step 5: Register EC2 Instances with Your Load Balancer \(p. 36\)](#)
- [Step 6: Tag Your Load Balancer \(Optional\) \(p. 36\)](#)
- [Step 7: Create and Verify Your Load Balancer \(p. 36\)](#)
- [Step 8: Delete Your Load Balancer \(Optional\) \(p. 37\)](#)

Step 1: Define Your Load Balancer

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

A *listener* is a process that checks for connection requests. It is configured with a protocol and a port for front-end (client to load balancer) connections and a protocol and a port for back-end (load balancer to back-end instance) connections. For information about the ports, protocols, and listener configurations supported by Elastic Load Balancing, see [Listeners for Your Load Balancer \(p. 70\)](#).

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 back-end instances on port 80 using HTTP. The second listener accepts HTTPS requests on port 443 and sends them to the back-end instances using HTTPS on port 443.

Because the second listener uses HTTPS for the front-end connection, you must install 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 back-end instances. Before you can install the SSL certificate on your load balancer, you must create the certificate, get the certificate signed by a Certificate Authority (CA), and then upload the certificate. For more information about creating and uploading SSL certificates, see [SSL Certificates for Elastic Load Balancing \(p. 20\)](#)

To define your load balancer

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.
3. Click **Create Load Balancer**.
4. In **Load Balancer name**, enter a name for your load balancer.

The name of your load balancer must be unique within your set of load balancers, can have a maximum of 32 characters, and can contain only alphanumeric characters and hyphens.

5. From **Create LB inside**, select the same network that you selected for your instances: EC2-Classic or a specific VPC.
6. [Default VPC] If you selected a default VPC and would like to choose the subnets for your load balancer, select **Enable advanced VPC configuration**.
7. Under **Listener Configuration**, leave the default listener, and click **Add** to add another listener. From the **Load Balancer Protocol** column for the new listener, select **HTTPS (Secure HTTP)**. This updates the **Load Balancer Port**, **Instance Protocol**, and **Instance Port** columns. From the

Instance Protocol column, select **HTTPS (Secure HTTP)**. This also updates the **Instance Port** column.

Load Balancer Protocol	Load Balancer Port	Instance Protocol	Instance Port	
HTTP	80	HTTP	80	✕
HTTPS (Secure HTTP)	443	HTTPS (Secure HTTP)	443	✕

Add

8. [EC2-VPC] Under **Select Subnets**, select at least one available subnet. The available subnets for the VPC for your load balancer are displayed under **Available Subnets**. Click the icon in the **Action** column for each subnet to attach. These subnets are moved under **Selected Subnets**.

Note

If you selected EC2-Classic as your network, or you have a default VPC but did not select **Enable advanced VPC configuration**, you do not see **Select Subnets**.

You can select at most one subnet per Availability Zone. If you select a second subnet in an Availability Zone, it replaces the previously selected subnet for that Availability Zone. To improve the availability of your load balancer, select subnets from more than one Availability Zone.

Available Subnets

Actions	Availability Zone	Subnet ID	Subnet CIDR	Name
+	us-west-2c	subnet-cb663da2	10.0.1.0/24	
+	us-west-2c	subnet-c9663da0	10.0.0.0/24	

Selected Subnets

Actions	Availability Zone	Subnet ID	Subnet CIDR	Name
−	us-west-2a	subnet-e4f33493	10.0.2.0/24	
−	us-west-2b	subnet-5264e837	10.0.3.0/24	

9. Click **Next: Assign Security Groups**.

Step 2: Assign Security Groups to Your Load Balancer in a VPC

If you selected a VPC as your network, you must assign your load balancer a security group that allows inbound traffic to the ports that you specified for your load balancer and the health checks for your load balancer.

Note

If you selected EC2-Classic as your network, you can continue to the next step. By default, Elastic Load Balancing provides a security group for load balancers in EC2-Classic.

To assign security group to your load balancer

1. On the **Assign Security Groups** page, select **Create a new security group**.
2. Enter a name and description for your security group, or leave the default name and description. This new security group contains a rule that allows traffic to the port that you configured your load balancer to use.

Assign a security group: ☒ Create a new security group
☐ Select an existing security group

Security group name:

Description:

Type <small>i</small>	Protocol <small>i</small>	Port Range <small>i</small>	Source <small>i</small>	
Custom TCP Rule ▾	TCP	80	Anywhere ▾ 0.0.0.0/0	✕
Custom TCP Rule ▾	TCP	443	Anywhere ▾ 0.0.0.0/0	✕

3. Click **Next: Configure Security Settings**.

Step 3: Configure Security Settings

When you use HTTPS or SSL for your front-end listener, you must install 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 back-end instances.

You must also specify a security policy. Elastic Load Balancing provides security policies that have predefined SSL negotiation configurations. You can select one of the predefined security policies, or you can create your own custom security policy. For more information, see [SSL Negotiation Configurations for Elastic Load Balancing \(p. 26\)](#).

If you have HTTPS/SSL on the back-end connection, you can enable authentication on your back-end instance. You can use this authentication to ensure that the back-end instances accept only encrypted communication and that they have the correct certificates.

To configure security settings

1. Under **Select Certificate**, do one of the following:
 - If you have uploaded your SSL certificate, select **Choose from an existing SSL Certificate**. Select your certificate from **Certificate Name**.
 - If you have a signed certificate ready to upload, select **Upload a new SSL Certificate**. Enter the name of the certificate. In **Private Key**, copy and paste the contents of the private key file (PEM-encoded). In **Public Key Certificate**, 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.

Select Certificate

An SSL Certificate allows you to configure the HTTPS/SSL listeners of your load balancer. You may select a previously uploaded certificate below, or define a new SSL Certificate. [Learn more](#) about setting up HTTPS load balancers and certificate management.

Certificate Type:

☐ Choose an existing SSL Certificate

☒ Upload a new SSL Certificate

Certificate Name:

my-server-certificate

Private Key:

-----BEGIN RSA PRIVATE KEY-----
MIIC1TCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UE
BhMCVVMxZzA5BjBhNVBAgTA1dBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQK
EzA5BjBhNVBAgTA1dBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQK
(pem encoded)

Public Key Certificate:

-----BEGIN CERTIFICATE-----
MIIE+TCCA+GgAwIBAgIQUS06HIX4Ks1oTW1s2A2krTANBgkqhkiG9w0BAQUF
cTELMAkGA1UEBhMCVVMxZzA5BjBhNVBAgTA1dBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQK
(pem encoded)

Certificate Chain:

Optional

(pem encoded)

Cancel

Save

2. Under **Select a Cipher**, do one of the following:
 - (Recommended) Verify that **Predefined Security Policy** is selected and set to **ELBSecurityPolicy-2015-05**.
 - Click **Predefined Security Policy**, and then select a policy.
 - Click **Custom Security Policy** and enable at least one protocol and one cipher. Under **SSL Protocols**, select one or more protocols to enable or disable. Under **SSL Options**, leave **Server Order Preference** selected, unless you do not want to use server order preference for SSL negotiation. Under **SSL Ciphers**, select one or more ciphers to enable or disable.

Tip

The DSA and RSA ciphers are specific to the signing algorithm. If you already have an SSL certificate, you must enable the cipher that was used to create the certificate.

3. (Optional) Under **Backend Certificate**, do the following:
 - a. Select **Enable backend authentication**.
 - b. In **Certificate Name**, enter the name of the public key certificate.
 - c. In **Certificate Body (pem encoded)**, copy and paste the contents of the certificate.
 - d. To add another certificate, click **Add another backend certificate**.
4. Click **Next: Configure Health Check**.

Step 4: Configure Health Checks

Elastic Load Balancing automatically checks the health of the registered EC2 instances for your load balancer. If Elastic Load Balancing finds an unhealthy instance, it stops sending traffic to the instance and reroutes traffic to the healthy instances. For more information about configuring health checks, see [Configure Health Checks](#) (p. 54).

To configure health checks for your instances

1. On the **Configure Health Check** page, select a ping protocol and ping port. Your EC2 instances must accept the specified traffic on the specified ping port.
2. In the **Ping Path** field, replace the default value with a single forward slash ("/"). This tells Elastic Load Balancing to send health check queries to the default home page for your web server, such as `index.html` or `default.html`.



Ping Protocol

Ping Port

Ping Path

3. Leave the other fields at their default values.
4. Click **Next: Add EC2 Instances**.

Step 5: Register EC2 Instances with Your Load Balancer

Your load balancer distributes traffic between the instances that are registered to it. You can select EC2 instances in a single Availability Zone or multiple Availability Zones within the same region as the load balancer. For more information, see [Back-end Instances for Your Load Balancer \(p. 53\)](#).

Note

When you register an instance with an elastic network interface (ENI) attached, the load balancer routes traffic to the primary IP address of the primary interface (eth0) of the instance.

To register EC2 instances with your load balancer

1. On the **Add EC2 Instances** page, select the instances to register with your load balancer.
2. Click **Next: Add Tags**.

Step 6: Tag Your Load Balancer (Optional)

You can tag your load balancer, or continue to the next step.

To add tags to your load balancer

1. On the **Add Tags** page, specify a key and a value for the tag.
2. To add another tag, click **Create Tag** and specify a key and a value for the tag.
3. After you are finished adding tags, click **Review and Create**.

Step 7: Create and Verify Your Load Balancer

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

To finish creating your load balancer

1. On the **Review** page, check your settings. If you need to make changes, click the corresponding link to edit the settings.
2. Click **Create** to create your load balancer.
3. After you are notified that your load balancer was created, click **Close**.
4. Select your new load balancer.

5. In the bottom pane, on the **Description** tab, check the **Status** row. If it indicates that some of your instances are not in service, its probably because they are still in the registration process. For more information, see [Troubleshooting Elastic Load Balancing: Registering Instances](#) (p. 134).
6. (Optional) After you've verified that at least one of your EC2 instances is `InService`, you can test your load balancer. Copy the string from the **DNS Name** field and paste it into the address field of an Internet-connected web browser. (For example, `my-load-balancer-1234567890.us-west-2.elb.amazonaws.com`.) If your load balancer is working, you see the default page of your HTTP server.

Step 8: Delete Your Load Balancer (Optional)

As soon as your load balancer becomes available, you are billed for each hour or partial hour that you keep it running. When you no longer need a load balancer, you can delete it. As soon as the load balancer is deleted, you stop incurring charges for it.

To delete your load balancer

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.
3. Select the load balancer.
4. Click **Actions**, and then click **Delete**.
5. When prompted for confirmation, click **Yes, Delete**.
6. (Optional) After you delete a load balancer, the EC2 instances associated with the load balancer continue to run, and you are billed for each hour or partial hour that you keep them running. For information about stopping or terminating your instances, see [Stop and Start Your Instance](#) or [Terminate Your Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

Create an HTTPS/SSL 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](#) (p. 37)
- [Step 2: Configure the SSL Security Policy](#) (p. 38)
- [Step 3: Configure Back-end Server Authentication \(Optional\)](#) (p. 42)
- [Step 4: Configure Health Checks \(Optional\)](#) (p. 43)
- [Step 5: Register EC2 Instances](#) (p. 44)
- [Step 6: Verify the Instances](#) (p. 44)
- [Step 7: Delete Your Load Balancer \(Optional\)](#) (p. 45)

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 back-end instance) connections. For information about the ports, protocols, and listener configurations supported by Elastic Load Balancing, see [Listeners for Your Load Balancer](#) (p. 70).

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 back-end instances on port 80 using HTTP. The second listener accepts HTTPS requests on port 443 and sends requests to back-end instances using HTTPS on port 443.

Because the second listener uses HTTPS for the front-end connection, you must install 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 back-end instances. Before you can install the SSL certificate on your load balancer, you must create the certificate, get the certificate signed by a Certificate Authority (CA), and then upload the certificate using the AWS Identity and Access Management (IAM) service. For information about creating and uploading SSL certificates, see [SSL Certificates for Elastic Load Balancing \(p. 20\)](#).

To configure listeners for your load balancer

1. If you have an SSL certificate and have uploaded it using IAM, use the [get-server-certificate](#) command to get the ARN of the certificate.

If you have not created and uploaded an SSL server certificate, complete the instructions described in [SSL Certificates for Elastic Load Balancing \(p. 20\)](#). Note the ARN of the certificate.

2. Use the following [create-load-balancer](#) command to configure the listener:

```
aws elb create-load-balancer --load-balancer-name my-loadbalancer --listeners
  "Protocol=http,LoadBalancerPort=80,InstanceProtocol=http,InstancePort=80"
  "Protocol=https,LoadBalancerPort=443,InstanceProtocol=https,InstancePort=443,
  SSLCertificateId=arn:aws:iam::123456789012:server-certificate/my-server-
certificate" --availability-zones us-west-2a
```

The following is an example response:

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

3. Save the DNS name in a safe place. You'll need it to connect to the 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. By default, Elastic Load Balancing configures your load balancer with the current predefined security policy, `ELBSecurityPolicy-2015-05`. We recommend that you use the default security policy. For more information about security policies, see [SSL Negotiation Configurations for Elastic Load Balancing \(p. 26\)](#).

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-2015-05` is associated with the load balancer on port 443.

```
{
  "LoadBalancerDescriptions": [
    {
```

```
...
"ListenerDescriptions": [
  {
    "Listener": {
      "InstancePort": 443,
      "SSLCertificateId": "arn:aws:iam::123456789012:server-
certificate/my-server-certificate",
      "LoadBalancerPort": 443,
      "Protocol": "HTTPS",
      "InstanceProtocol": "HTTPS"
    },
    "PolicyNames": [
      "ELBSecurityPolicy-2015-05"
    ]
  },
  {
    "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.

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](#) (p. 27).

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 SSLNegotiationPoli
cyType
--policy-attributes AttributeName=Reference-Security-Policy,AttributeValue=pre
defined-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
```

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": 443,
            "SSLCertificateId":
"arn:aws:iam::123456789012:server-certificate/my-server-certificate",
            "LoadBalancerPort": 443,
            "Protocol": "HTTPS",
            "InstanceProtocol": "HTTPS"
          },
          "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.

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 SSLNegotiation
PolicyType
--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
```

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": 443,
            "SSLCertificateId":
"arn:aws:iam::123456789012:server-certificate/my-server-certificate",
            "LoadBalancerPort": 443,
            "Protocol": "HTTPS",
            "InstanceProtocol": "HTTPS"
          },
          "PolicyNames": [
            "my-SSLNegotiation-policy"
          ]
        }
      ]
    }
  ],
```

```
{
  "Listener": {
    "InstancePort": 80,
    "LoadBalancerPort": 80,
    "Protocol": "HTTP",
    "InstanceProtocol": "HTTP"
  },
  "PolicyNames": [ ]
},
...
]
}
```

Step 3: Configure Back-end Server Authentication (Optional)

If you have HTTPS/SSL on the back-end connection, you can choose to enable authentication on your back-end instance. This authentication can be used to ensure that back-end instances accept only encrypted communication and that they have the correct certificates.

You enable the back-end server authentication by creating a public key policy that uses a public key for authentication. You use this public key policy to create a back-end server authentication policy. Finally, you enable the back-end server authentication by setting the back-end server authentication policy with the back-end server port. In this example, the back-end server is listening for the HTTPS protocol on port 443.

The value of the public key policy is the public key of the certificate that the back-end servers present to the load balancer. You can retrieve the public key using OpenSSL.

To configure back-end server authentication

1. Use the 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=MIICiTCCAfIC
CQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBIDELMAKGA1UEBhMC
VVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBASTC0lBTSBDb25zb2x1MRIwEAYDVQQDEw1UZXR0Q21sYWMxHZAAd
BgkqhkiG9w0BCQEWEG5vb251QGFTYXpvc251b20wHhcnMTFwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBIDELMAKGA1UEBhMCVVMxCzAJBgNVBAGTAldBMRAwDgYD
VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBASTC0lBTSBDb25z
b2x1MRIwEAYDVQQDEw1UZXR0Q21sYWMxHZAAdBgkqhkiG9w0BCQEWEG5vb251QGFT
YXpvc251b20wZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
21uUSfwfEvySWtC2XADZ4nB+BLygVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
rDHudUzg3qX4waLG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpe
Ibb30hjZnzcvcQAARHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAatCu4
nUhVVxYUntned9+h8Mg9q6q+auNKyExzyLwax1Aoo7TJHidbtS4J5iNmZgXL0Fkb
```



```
FFBjvsfPjI1J00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvJx79LjSTb  
NYiyTVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
```

Note

To specify a public key value for the *policy-attributes* argument, 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 inside the value for the *policy-attributes* argument.

3. Use the following [create-load-balancer-policy](#) command to create a back-end server authentication policy by referring to my-PublicKey-policy. You can refer to multiple public key policies. When multiple public key policies are used, the load balancer tries all the keys, one at a time, for authentication. If one of the public keys matches the server certificate, authentication passes.

```
aws elb create-load-balancer-policy --load-balancer-name my-loadbalancer -  
-policy-name my-authentication-policy --policy-type-name BackendServerAu  
thenticationPolicyType --policy-attributes AttributeName=PublicKeyPolicy  
Name,AttributeValue=my-PublicKey-policy
```

4. Use the following [set-load-balancer-policies-for-backend-server](#) command to set my-authentication-policy to the instance (back-end server) port.

```
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 [Configure Health Checks](#) (p. 54).

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 back-end instances

Use the following [configure-health-check](#) command:

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

The following is an example response:

```
{
  "HealthCheck": {
    "HealthyThreshold": 2,
    "Interval": 30,
    "Target": "HTTP:80/png",
    "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 [Back-end Instances for Your Load Balancer \(p. 53\)](#).

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 [Troubleshooting Elastic Load Balancing: Registering Instances \(p. 134\)](#).

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 Load Balancer

If you have a load balancer, you can add a new listener that accepts HTTPS requests on port 443 for both the front-end and back-end connections.

For information about creating a new HTTPS listener, see [Create an HTTPS Load Balancer \(p. 31\)](#).

Contents

- [Prerequisites \(p. 46\)](#)
- [Add an HTTPS Listener Using the Console \(p. 46\)](#)
- [Add an HTTPS Listener Using the AWS CLI \(p. 47\)](#)

Prerequisites

If you do not have an SSL certificate, you can create and upload it. For more information, see [SSL Certificates for Elastic Load Balancing \(p. 20\)](#).

If you are using a certificate that has not been uploaded yet, ensure that it meets the criteria described in [Upload the Signed Certificate \(p. 22\)](#). If your certificate does not meet the criteria, you might get an error when you upload it. Create a new SSL certificate and upload it.

Add an HTTPS Listener Using the Console

To enable HTTPS support for your listeners, you must install 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 back-end instances.

To add an HTTPS listener to your load balancer

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.
3. Select your load balancer.
4. In the bottom pane, select the **Listeners** tab.
5. Click **Edit**.
6. In the **Edit Listeners** dialog box, click **Add**.
7. In the **Load Balancer Protocol** column, select **HTTPS (Secure HTTP)**. This updates the **Load Balancer Port**, **Instance Protocol**, and **Instance Port** columns. In the **Instance Protocol** column, select **HTTPS (Secure HTTP)**. This also updates the **Instance Port** column.
8. By default, Elastic Load Balancing selects the current predefined security policy, **ELBSecurityPolicy-2015-05**, for your HTTPS/SSL listener. This is the recommended setting. This policy uses server order preference (see [Predefined SSL Security Policies \(p. 27\)](#)) to negotiate SSL connections.

In the **Cipher** column, click **Change**, and then do one of the following:

- (Recommended) Ensure that **Predefined Security Policy** is selected and set to **ELBSecurityPolicy-2015-05**, and then click **Save**.
- Click **Predefined Security Policy**, select a policy, and then click **Save**.
- Click **Custom** and enable at least one protocol and one cipher. Under **SSL Protocols**, select one or more protocols to enable or disable. Under **SSL Options**, leave **Server Order Preference** selected, unless you do not want to use server order preference for SSL negotiation. Under **SSL Ciphers**, select one or more ciphers to enable or disable. Click **Save**.

Tip

The DSA and RSA ciphers are specific to the signing algorithm. If you already have an SSL certificate, you must enable the cipher that was used to create the certificate.

9. If you already have certificate installed on your load balancer and want to continue using it, you can skip this step.

In the **SSL Certificate** column, click **Change**, and then do one of the following:

- Select **Choose from an existing SSL Certificate**. Select your certificate from **Certificate Name**, and then click **Save**.
- Select **Upload a new SSL Certificate**. Enter the name of the certificate. In **Private Key**, copy and paste the contents of the private key file (PEM-encoded). In **Public Key Certificate**, 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.

Select Certificate

An SSL Certificate allows you to configure the HTTPS/SSL listeners of your load balancer. You may select a previously uploaded certificate below, or define a new SSL Certificate. [Learn more](#) about setting up HTTPS load balancers and certificate management.

Certificate Type:

Choose an existing SSL Certificate

Upload a new SSL Certificate

Certificate Name:

my-server-certificate

Private Key:

-----BEGIN RSA PRIVATE KEY-----
MIIC1TCCA+ICQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCB1DELMAkGA1UE
BhMCVVMxZzA5BjBjNVBAgTA1dBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQK
BzA5BjBjNVBAgTA1dBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQK
(pem encoded)

Public Key Certificate:

-----BEGIN CERTIFICATE-----
MIIE+TCCA+GgAwIBAgIQUS06HIX4Ks1oTW1s2A2krTANBgkqhkiG9w0BAQUF
cTELMAkGA1UEBhMCVVMxZzA5BjBjNVBAgTA1dBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQK
(pem encoded)

Certificate Chain:

Optional

(pem encoded)

Cancel

Save

10. (Optional) Click **Add** to add additional listeners.
11. Click **Save** to add the listeners you just configured.

Add an HTTPS Listener Using the AWS CLI

To enable HTTPS support for your listeners, you must install 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 back-end instances.

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

1. Get the Amazon Resource Name (ARN) of your SSL certificate. For example, `arn:aws:iam::123456789012:server-certificate/my-server-certificate`.
2. Use the following [create-load-balancer-listeners](#) command to add the listener to your load balancer:

```
aws elb create-load-balancer-listeners --load-balancer-name my-loadbalancer  
--listeners Protocol=HTTPS,LoadBalancerPort=443,InstanceProtocol=HTTPS,In  
stancePort=443,SSLCertificateId=arn:aws:iam::123456789012:server-certific  
ate/my-server-certificate
```

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-loadbalancer
```

The following is an example response:

```
{  
  "LoadBalancerDescriptions": [  
    {
```

```
...
"ListenerDescriptions": [
  {
    "Listener": {
      "InstancePort": 443,
      "SSLCertificateId":
"arn:aws:iam::123456789012:server-certificate/my-server-certificate",
      "LoadBalancerPort": 443,
      "Protocol": "HTTPS",
      "InstanceProtocol": "HTTPS"
    },
    "PolicyNames": [
      "ELBSecurityPolicy-2015-05"
    ]
  },
  {
    "Listener": {
      "InstancePort": 80,
      "LoadBalancerPort": 80,
      "Protocol": "HTTP",
      "InstanceProtocol": "HTTP"
    },
    "PolicyNames": []
  }
],
...
]
```

Update the SSL Certificate for Your Load Balancer

If you are using the HTTPS/SSL protocol for your listeners, you might have an SSL server certificate installed on your load balancer. Your SSL certificate comes with a validity period. You must replace the certificate before its validity period ends. To replace a certificate, you must first create a new certificate by following the same steps you used when you created the current certificate. For more information about creating an SSL certificate and uploading it, see [SSL Certificates for Elastic Load Balancing \(p. 20\)](#).

The following examples show you how to update an SSL certificate.

Contents

- [Prerequisites \(p. 48\)](#)
- [Updating an SSL Certificate Using the Console \(p. 49\)](#)
- [Updating an SSL Certificate Using the AWS CLI \(p. 49\)](#)

Prerequisites

Verify that your certificate meets the [prerequisites \(p. 23\)](#).

Updating an SSL Certificate Using the Console

To update an SSL certificate for an HTTPS load balancer

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.
3. Select your load balancer.
4. In the **Listeners** tab, click **Change** in the **SSL Certificate** column for the certificate.
5. In the **Select Certificate** dialog box, do one of the following:
 - If you have already uploaded an SSL certificate using IAM, select **Choose an existing SSL Certificates**, select the certificate from **Certificate Name**, and then click **Save**.
 - If you have an SSL certificate to upload, select **Upload a new SSL Certificate**. Enter a name for the certificate, copy the required information to the form, and then click **Save**. Note that the certificate chain is not required if the certificate is a self-signed certificate.

Select Certificate [X]

An SSL Certificate allows you to configure the HTTPS/SSL listeners of your load balancer. You may select a previously uploaded certificate below, or define a new SSL Certificate. [Learn more](#) about setting up HTTPS load balancers and certificate management.

Certificate Type: ☐ Choose an **existing** SSL Certificate
☒ Upload a **new** SSL Certificate

Certificate Name*: my-server-certificate

Private Key*: -----BEGIN RSA PRIVATE KEY-----
MIICATCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UE
BhMCVVMxZzA5BGNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQK
(pem encoded)

Public Key Certificate*: -----BEGIN CERTIFICATE-----
MIIE+TCCA+GgAwIBAgIQU306HIX4Ks1oTW1s2A2krTANBgkqhkiG9w0BAQUF
tTELMAkGA1UEBhMCVVMxZzA5BGNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQK
(pem encoded)

Certificate Chain: Optional
(pem encoded)

Cancel Save

Updating an SSL Certificate Using the AWS CLI

To update an SSL certificate for an HTTPS load balancer

1. If you have an SSL certificate but have not uploaded it, complete the instructions described in [Upload the Signed Certificate](#) (p. 22).
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-loadbalancer --load-balancer-port 443 --ssl-certificate-id arn:aws:iam::123456789012:server-certificate/my-new-certificate
```

Update the SSL Negotiation Configuration of Your 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 Elastic Load Balancing](#) (p. 26). For information about the configurations of the security policies provided by Elastic Load Balancing, see [Predefined SSL Security Policies](#) (p. 27).

If you create an HTTPS/SSL listener without associating a security policy, Elastic Load Balancing associate the current predefined security policy, ELBSecurityPolicy-2015-05, with your load balancer.

If you have an existing load balancer with an SSL negotiation configuration that does not use the latest protocols and ciphers, we recommend that you update your load balancer to use ELBSecurityPolicy-2015-05. If you prefer, you can create a custom configuration. We strongly recommend that you test the new security policies before you upgrade your load balancer configuration.

The following examples show you how to update the SSL negotiation configuration for an HTTPS/SSL listener.

Contents

- [Update the SSL Negotiation Configuration Using the Console](#) (p. 50)
- [Update the SSL Negotiation Configuration Using the AWS CLI](#) (p. 51)

Update the SSL Negotiation Configuration Using the Console

You can use a predefined security policy or a custom policy.

To update SSL negotiation configuration for an HTTPS/SSL load balancer

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.
3. Select your load balancer.
4. In the bottom pane, select the **Listeners** tab.
5. In the **Cipher** column of the listener to be updated, click **Change**.
6. In the **Select a Cipher** dialog box, select a security policy using one of the following options:
 - Click **Predefined Security Policy**, select one of the predefined policies from the list, and then click **Save**.
 - Click **Custom Security Policy**, select one or more protocols to enable, select **Server Order Preference** to use the order listed in the [Predefined SSL Security Policies](#) (p. 27) for SSL

negotiation, select one or more ciphers to enable (if you have an SSL certificate, you must enable the cipher that was used to create the certificate), and then click **Save**.

Update the SSL Negotiation Configuration Using the AWS CLI

You can use the current predefined security policy, **ELBSecurityPolicy-2015-05**, 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:

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

For information about the configuration for the predefined security policies, see [Predefined SSL Security Policies](#) (p. 27).

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 current predefined security policy:

```
aws elb create-load-balancer-policy --load-balancer-name my-loadbalancer  
--policy-name my-SSLNegotiation-policy --policy-type-name SSLNegotiation  
PolicyType  
--policy-attributes AttributeName=Reference-Security-Policy,AttributeValue=ELB  
SecurityPolicy-2015-05
```

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
```

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.

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 SSLNegotiation
PolicyType
--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
```

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.

Back-end Instances for Your Load Balancer

After you've created 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. 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 Back-end Instances \(p. 53\)](#)
- [Configure Health Checks \(p. 54\)](#)
- [Configure Security Groups for Your Load Balancer \(p. 57\)](#)
- [Add or Remove Availability Zones for Your Load Balancer in EC2-Classic \(p. 61\)](#)
- [Add or Remove Subnets for Your Load Balancer in a VPC \(p. 65\)](#)
- [De-register and Register EC2 Instances with Your Load Balancer \(p. 67\)](#)

Best Practices for Your Back-end Instances

- Install a web server, such as Apache or Internet Information Services (IIS), on all instances that you plan to register with your load balancer.
- [EC2-Classic] The load balancer communicates with its registered instances using the IP addresses associated with the instances. When an instance in EC2-Classic is stopped and then started, the IP address associated with the instance changes. This prevents the load balancer from routing traffic to your restarted instance. Therefore, we recommend that you de-register a stopped instance from your load balancer, and then register it again after you start it. For more information, see [De-register and Register EC2 Instances with Your Load Balancer \(p. 67\)](#).
- 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 adjust your instance's security group rules. For

more information, see [Security Group Rules for Path MTU Discovery](#) in the *Amazon EC2 User Guide for Linux Instances*.

- The security group for your back-end instances should have inbound rules for the listener port and the health check port for your load balancer, with the source as the security group for your load balancer.

Configure Health Checks

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

The load balancer routes traffic only to the healthy instances. When the load balancer determines that an instance is unhealthy, it stops routing traffic to that instance. The load balancer resumes routing traffic 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 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 an Elastic Load Balancing Health Check to Your Auto Scaling Group](#) in the *Auto Scaling Developer Guide*.

Contents

- [Health Check Configuration](#) (p. 54)
- [Update the Health Check Configuration](#) (p. 56)
- [Check the Health of Your Instances](#) (p. 56)
- [Troubleshoot Health Checks](#) (p. 56)

Health Check Configuration

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

Field	Description
Ping Protocol	The protocol to use to connect with the instance. Valid values: TCP, HTTP, HTTPS, and SSL Console default: HTTP CLI/API default: TCP

Field	Description
Ping Port	<p>The port to use to connect with the instance, as a <code>protocol:port</code> 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.</p> <p>Ping protocols: TCP, HTTP, HTTPS, and SSL</p> <p>Ping port range: 1 to 65535</p> <p>Console default: <code>HTTP:80</code></p> <p>CLI/API default: <code>TCP:80</code></p>
Ping Path	<p>The destination for the HTTP or HTTPS request.</p> <p>An HTTP or HTTPS GET request is issued to the instance on the ping port and the ping path. If the load balancer receives 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'.</p> <p>Default: <code>/index.html</code></p>
Response Timeout	<p>The amount of time to wait when receiving a response from the health check, in seconds.</p> <p>Valid values: 2 to 60</p> <p>Default: 5</p>
HealthCheck Interval	<p>The amount of time between health checks of an individual instance, in seconds.</p> <p>Valid values: 5 to 300</p> <p>Default: 30</p>
Unhealthy Threshold	<p>The number of consecutive failed health checks that must occur before declaring an EC2 instance unhealthy.</p> <p>Valid values: 2 to 10</p> <p>Default: 2</p>
Healthy Threshold	<p>The number of consecutive successful health checks that must occur before declaring an EC2 instance healthy.</p> <p>Valid values: 2 to 10</p> <p>Default: 10</p>

The load balancer sends a request to each registered instance at the ping port and ping path every `Interval` seconds. It waits for the instance to respond within the response timeout period. If the health checks exceed the threshold for consecutive failed responses, the load balancer takes the instance out

of service. When the health checks exceed the threshold for consecutive successful responses, the load balancer puts the instance back in service.

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. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.
3. Select your load balancer.
4. On the **Health Check** tab, click **Edit Health Check**.
5. In the **Configure Health Check** dialog box, update the settings as needed.
6. Click **Save**.

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/png,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. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.
3. Select your load balancer.
4. On the **Description** tab, find the **Status** row. The status message indicates how many instances are in service.
5. On the **Instances** tab, the **Status** column indicates the status of the 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 [Troubleshooting Elastic Load Balancing: Health Check Configuration](#) (p. 130).

Configure Security Groups for Your 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 for Linux Instances*.

There is a significant difference between the way Elastic Load Balancing supports security groups in EC2-Classic and in a VPC. In EC2-Classic, Elastic Load Balancing provides a special source security group that you can use to ensure that back-end instances receives traffic only from your load balancer. You can't modify this source security group. In a VPC, you provide the security group for your load balancer, which enables you to choose the ports and protocols to allow. For example, you can open Internet Control Message Protocol (ICMP) connections for the load balancer to respond to ping requests (however, ping requests are not forwarded to any back-end instances).

Contents

- [Security Groups for Load Balancers in EC2-Classic \(p. 57\)](#)
- [Security Groups for Load Balancers in a VPC \(p. 60\)](#)

Security Groups for Load Balancers in EC2-Classic

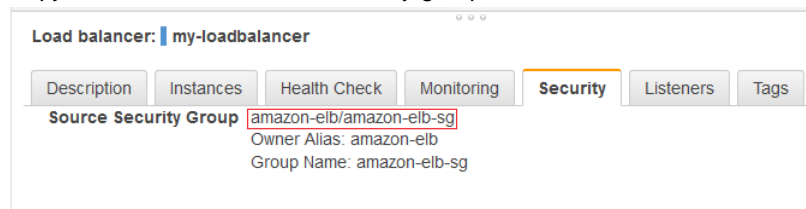
To allow communication between your load balancer and your back-end instances launched in EC2-Classic, create an inbound rule for the security group for your back-end instances that allows inbound traffic from either all IP addresses (using the `0.0.0.0/0` CIDR block) or only from the load balancer (using the source security group provided by Elastic Load Balancing).

Lock Down Traffic Between Your Load Balancer and Back-end Instances Using the Console

Use the following procedure to lock down traffic between your load balancer and your back-end instances.

To lock down traffic between our load balancer and instances

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.
3. Select your load balancer.
4. Select the **Security** tab.
5. Copy the name of the source security group.



6. Add a rule to the security group for your back-end instances as follows:

- a. On the **Instances** tab, click the instance ID of one of the instances registered with your load balancer.
- b. On the **Description** tab, find **Security groups**, and then click the name of the security group.
- c. On the **Inbound** tab, click **Edit**, and then click **Add Rule**.
- d. From the **Type** column, select the protocol type. The **Protocol** and **Port Range** columns are populated.
- e. From the **Source** column, select `Custom IP`. In the box, paste the name of the source security group that you copied earlier (for example, `amazon-elb/amazon-elb-sg`).
- f. (Optional) If your security group has rules that are less restrictive than the rule you just added, remove the less restrictive rule by clicking its delete icon.

Important

If you need to connect to your back-end instances, do not delete the inbound rules that allow traffic on port 22 (SSH) or port 3389 (RDP).

- g. Click **Save**.

Lock Down Traffic Between Your Load Balancer and Back-end Instances Using the AWS CLI

Use the following procedure to lock down traffic between your load balancer and your back-end instances.

To lock down traffic between our load balancer and instances

1. Use the following `describe-load-balancers` command to display the name and owner of the source security group for your load balancer:

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

The response includes the name and owner in the `SourceSecurityGroup` field. For example:

```
{
  "LoadBalancerDescriptions": [
    {
      ...
      "SourceSecurityGroup": {
        "OwnerAlias": "amazon-elb",
        "GroupName": "amazon-elb-sg"
      }
    }
  ]
}
```

2. Add a rule to the security group for your back-end instances as follows:
 - a. If you do not know the name of the security group for your back-end instances, use the following `describe-instances` command to get the name and ID of the security group for the specified back-end instance:

```
aws ec2 describe-instances --instance-ids i-315b7e51
```


The response includes the name and ID of the security group in the `SecurityGroups` field. Make a note of the name of the security group; you'll use it in the next step.

- b. Use the following [authorize-security-group-ingress](#) command to add a rule to the security group for your back-end instance to allow traffic from your load balancer:

```
aws ec2 authorize-security-group-ingress --group-name my-security-group
--source-security-group-name amazon-elb-sg --source-security-group-
owner-id amazon-elb
```

3. (Optional) Use the following [describe-security-groups](#) command to verify that the security group has the new rule:

```
aws ec2 describe-security-groups --group-names my-security-group
```

The response includes a `UserIdGroupPairs` data structure that lists the security groups that are granted permissions to access the instance.

```
{
  "SecurityGroups": [
    {
      ...
      "IpPermissions": [
        {
          "IpRanges": [],
          "FromPort": -1,
          "IpProtocol": "icmp",
          "ToPort": -1,
          "UserIdGroupPairs": [
            {
              "GroupName": "amazon-elb-sg",
              "GroupId": "sg-5a9c116a",
              "UserId": "amazon-elb"
            }
          ]
        },
        {
          "IpRanges": [],
          "FromPort": 1,
          "IpProtocol": "tcp",
          "ToPort": 65535,
          "UserIdGroupPairs": [
            {
              "GroupName": "amazon-elb-sg",
              "GroupId": "sg-5a9c116a",
              "UserId": "amazon-elb"
            }
          ]
        }
      ],
      "IpRanges": [],
      "FromPort": 1,
      "IpProtocol": "udp",
      "ToPort": 65535,
      "UserIdGroupPairs": [
```

```
{
  {
    "GroupName": "amazon-elb-sg",
    "GroupId": "sg-5a9c116a",
    "UserId": "amazon-elb"
  }
},
. . .
}
```

4. (Optional) If your security group has rules that are less restrictive than the rule you just added, use the [revoke-security-group-ingress](#) command to remove the less restrictive rules. For example, the following command removes a rule that allows TCP traffic from everyone (CIDR range 0.0.0.0/0):

```
aws ec2 revoke-security-group-ingress --group-name my-security-group --protocol tcp --port 80 --cidr 0.0.0.0/0
```

Important

If you need to connect to your back-end instances, do not delete the inbound rules that allow traffic on port 22 (SSH) or port 3389 (RDP).

Security Groups for Load Balancers in a VPC

You must ensure that your load balancer can communicate with your back-end instances on both the listener port and the health check port. Your security groups must allow traffic in both directions on these ports for each subnet attached to your load balancer.

When you use the AWS Management Console to create a load balancer in a VPC, you can choose an existing security group for the VPC or create a new security group for the VPC. 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 to 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.

The security group for your back-end instances should have inbound rules for the listener port and the health check port with the source as the security group for your load balancer.

Contents

- [Manage Security Groups Using the Console \(p. 61\)](#)

- [Manage Security Groups Using the AWS CLI \(p. 61\)](#)

Manage Security Groups Using the Console

Use the following procedure to update or reset the security group assigned to your load balancer in a VPC.

To update a security group assigned to your load balancer

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.
3. Select your load balancer.
4. Select the **Security** tab.
5. In the **Security Groups** pane, click **Edit**.
6. On the **Edit security groups** page, select your new security group and then click **Save**.

Manage Security Groups Using the AWS CLI

Use the following `apply-security-groups-to-load-balancer` command to assign an additional security group to a load balancer in a VPC:

```
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-fc448899"
    "sg-53fae93f"
  ]
}
```

Add or Remove Availability Zones for Your Load Balancer in EC2-Classic

You can set up your load balancer in EC2-Classic to distribute incoming requests across EC2 instances in a single Availability Zone or multiple Availability Zones within the same region as the load balancer. The load balancer routes traffic equally across the enabled Availability Zones. First, launch EC2 instances in all the Availability Zones that you plan to use, and then register the instances with your load balancer. After you add these Availability Zones, the load balancer starts routing traffic to the registered instances in the Availability Zones.

You can modify the list of Availability Zones for your load balancer at any time. To expand the availability of your application, launch instances in an additional Availability Zone, register the new instances with your load balancer, and then add the new Availability Zone to your load balancer. After you add an Availability Zone, the load balancer starts routing traffic to the instances in the enabled Availability Zone.

You might want to remove an Availability Zone from your load balancer temporarily when it has no healthy instances, when it experiences a decrease in traffic, or when you want to troubleshoot or update the

instances. After you've removed an Availability Zone, the load balancer stops routing traffic to the instances in the disabled Availability Zone and continues to route traffic to the instances in the enabled Availability Zones.

If your load balancer is in a VPC, see [Add or Remove Subnets for Your Load Balancer in a VPC](#) (p. 65).

Contents

- [Add or Remove an Availability Zone Using the AWS Management Console](#) (p. 62)
- [Add or Remove an Availability Zone Using the AWS CLI](#) (p. 63)

Add or Remove an Availability Zone Using the AWS Management Console

You can expand the availability of your application to an additional Availability Zone. First, register the EC2 instances in the Availability Zone with the load balancer so that they are ready to receive traffic when the load balancer begins routing traffic to the Availability Zone. Then, enable the Availability Zone.

To add an Availability Zone to your load balancer

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.
3. Select your load balancer.
4. In the bottom pane, select the **Instances** tab.
5. Click **Edit Instances**.
6. In the **Add and Remove Instances** dialog box, select instances in the new Availability Zone, and then click **Save**.
7. Click **Edit Availability Zones**.
8. In the **Add and Remove Availability Zones** dialog box, verify that the new Availability Zone is selected, select the new Availability Zone otherwise, and then click **Save**.

After you disable an Availability Zone, the instances in the Availability Zone remain registered with the load balancer. You can optionally de-register these instances from your load balancer.

To remove an Availability Zone from your load balancer

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.
3. Select your load balancer.
4. In the bottom pane, select the **Instances** tab.
5. Click **Edit Availability Zones**.
6. In the **Add and Remove Availability Zones** dialog box, identify the row that lists the Availability Zone, and then click **Save**.
7. (Optional) To de-register the instances in the removed Availability Zone from your load balancer, follow the instructions in [De-register and Register Your Instances Using the Console](#) (p. 68).

Add or Remove an Availability Zone Using the AWS CLI

You can expand your EC2 application to run in an additional Availability Zone. To do so, you first register the EC2 instances in the Availability Zone with the load balancer so that they are ready to receive traffic when the load balancer begins routing traffic.

To add an Availability Zone to your load balancer

1. Use the following [register-instances-with-load-balancer](#) command to register the instances:

```
aws elb register-instances-with-load-balancer --load-balancer-name my-loadbalancer --instances i-3a8cf324 i-2603ca33
```

The following is an example response:

```
{
  "Instances": [
    {
      "InstanceId": "i-4f8cf126"
    },
    {
      "InstanceId": "i-0bb7ca62"
    },
    {
      "InstanceId": "i-3a8cf324"
    },
    {
      "InstanceId": "i-2603ca33"
    }
  ]
}
```

2. (Optional) Use the following [describe-instance-health](#) command to verify that the instances are registered with the load balancer:

```
aws elb describe-instance-health --load-balancer-name my-loadbalancer --instances i-3a8cf324 i-2603ca33
```

The following is an example response:

```
{
  "InstanceStates": [
    {
      "InstanceId": "i-3a8cf324",
      "ReasonCode": "N/A",
      "State": "InService",
      "Description": "N/A"
    },
    {
      "InstanceId": "i-2603ca33",
      "ReasonCode": "Instance",
      "State": "OutOfService",
      "Description": "N/A"
    }
  ]
}
```

```
        "Description": "Instance registration is still in progress"
      }
    ]
  }
}
```

If the instance state description shows `Instance registration is still in progress`, Elastic Load Balancing is checking the health state of the instances. You can proceed to add the new Availability Zone while the health check is underway.

3. Use the following [enable-availability-zones-for-load-balancer](#) command to enable the Availability Zone:

```
aws elb enable-availability-zones-for-load-balancer --load-balancer-name
my-loadbalancer --availability-zones us-west-2b
```

The following is an example response:

```
{
  "AvailabilityZones": [
    "us-west-2a",
    "us-west-2b"
  ]
}
```

After you disable an Availability Zone, the instances in the Availability Zone remain registered with the load balancer. You can optionally de-register these instances from your load balancer.

To remove an Availability Zone from your load balancer

1. Use the following [disable-availability-zones-for-load-balancer](#) command:

```
aws elb disable-availability-zones-for-load-balancer --load-balancer-name
my-loadbalancer --availability-zones us-west-2a
```

The following is an example response:

```
{
  "AvailabilityZones": [
    "us-west-2b"
  ]
}
```

2. (Optional) To de-register the instances in the removed Availability Zone from the load balancer, follow the instructions in [De-register and Register Your Instances Using the AWS CLI](#) (p. 68).

Add or Remove Subnets for Your Load Balancer in a VPC

For load balancers in a VPC, we recommend that you attach one subnet per Availability Zone for at least two Availability Zones. The load balancer routes traffic equally across the subnets. First, launch EC2 instances in all the subnets that you plan to use, and then register the instances with your load balancer. After you attach these subnets, the load balancer starts routing traffic to the registered instances in the subnets.

You can modify the list of subnets for your load balancer at any time. To expand the availability of your application, launch instances in an additional subnet, register the new instances with your load balancer, and then add the new subnet to your load balancer. After you add a subnet, the load balancer starts routing traffic to the instances in the subnet.

You might want to remove a subnet from your load balancer temporarily when it has no healthy instances, when it experiences a decrease in traffic, or when you want to troubleshoot or update the instances. After you've removed a subnet, the load balancer stops routing traffic to the instances in the disabled subnet and continues to route traffic to the instances in the enabled subnets.

If your load balancer is in EC2-Classic, see [Add or Remove Availability Zones for Your Load Balancer in EC2-Classic \(p. 61\)](#).

Contents

- [Add or Remove Subnets Using the Console \(p. 65\)](#)
- [Attach or Detach Subnets Using the AWS CLI \(p. 66\)](#)

Add or Remove Subnets Using the Console

First, register the EC2 instances in the new subnet with the load balancer so that they are ready to receive traffic when the load balancer begins routing traffic to the subnet. Then, use the following procedure to attach the subnet to your load balancer.

To attach a subnet to your load balancer

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.
3. Select your load balancer.
4. In the bottom pane, select the **Instances** tab.
5. Click **Edit Instances**.
6. In the **Add and Remove Instances** dialog box, select instances in the new subnet, and then click **Save**.
7. Click **Edit Availability Zones**.
8. In the **Add and Remove Subnets** dialog box, under **Available Subnets**, click the icon in the **Action** column for the subnet to attach. The subnets are moved under **Selected Subnets**.

Note that you can select at most one subnet per Availability Zone. If you select a second subnet in an Availability Zone, it replaces the previously selected subnet for that Availability Zone.

9. Click **Save**.

Use the following procedure to detach your load balancer from a subnet. After you disable a subnet, the instances in the subnet remain registered with the load balancer. You can optionally de-register these instances from your load balancer.

To detach a subnet from your load balancer

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.
3. Select your load balancer.
4. In the bottom pane, select the **Instances** tab.
5. Click **Edit Availability Zones**.
6. In the **Add and Remove Subnets** dialog box, under **Selected Subnets**, click the icon in the **Action** column for the subnet to detach. The subnets are moved under **Available Subnets**.
7. Click **Save**.
8. (Optional) To de-register the instances in the removed subnet from the load balancer, follow the instructions in [De-register and Register Your Instances Using the Console](#) (p. 68).

Attach or Detach Subnets Using the AWS CLI

First, register the EC2 instances in the new subnets with the load balancer so that they are ready to receive traffic when the load balancer begins routing traffic to the subnets. Then, attach the subnet to your load balancer.

To add a subnet to your load balancer

1. Use the following [register-instances-with-load-balancer](#) command to register the instances:

```
aws elb register-instances-with-load-balancer --load-balancer-name my-load-balancer --instances i-3a8cf324 i-2603ca33
```

The following is an example response:

```
{
  "Instances": [
    {
      "InstanceId": "i-4f8cf126"
    },
    {
      "InstanceId": "i-0bb7ca62"
    },
    {
      "InstanceId": "i-3a8cf324"
    },
    {
      "InstanceId": "i-2603ca33"
    }
  ]
}
```

2. Use the following [attach-load-balancer-to-subnets](#) command to attach the 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 attached subnets for the load balancer. For example:


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

After you disable a subnet, the instances in the subnet remain registered with the load balancer. You can optionally de-register these instances from your load balancer.

To remove a subnet from your load balancer

1. Use the following [detach-load-balancer-from-subnets](#) command as to detach 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 attached subnets for the load balancer. For example:

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

2. (Optional) To de-register the instances in the removed subnet from your load balancer, follow the instructions in [De-register and Register Your Instances Using the AWS CLI](#) (p. 68).

De-register and Register EC2 Instances with Your Load Balancer

Elastic Load Balancing registers your EC2 instance with your load balancer using its IP address. The load balancer continuously monitors the health of the registered instances, and routes traffic to the healthy instances. The load balancer stops routing traffic to an instance if it detects that it is unhealthy, or when it is de-registered. Instead, it routes traffic to the remaining healthy instances.

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

[EC2-VPC] If you stop and start an EC2 instance in a VPC, the IP address associated with your instance does not change. However, your load balancer might not immediately recognize that the stopped instance has started and is ready to receive traffic. Therefore, we recommend that you register an instance with the load balancer after you stop and start it.

[EC2-Classic] If you stop and start an EC2 instance launched in EC2-Classic, the IP address associated with your instance changes, and your load balancer doesn't recognize the new IP address and can't route traffic to the instance. To avoid this problem, you must de-register the instance from your load balancer, stop and start the instance, and then register the instance with the load balancer.

Contents

- [De-register and Register Your Instances Using the Console \(p. 68\)](#)
- [De-register and Register Your Instances Using the AWS CLI \(p. 68\)](#)

De-register and Register Your Instances Using the Console

You can de-register your stopped back-end instance from load balancer, and then register your restarted instance with the load balancer.

To de-register your back-end instances from your load balancer

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.
3. Select your load balancer.
4. In the bottom pane, select the **Instances** tab.
5. In **Actions** column for the instance to de-register, click **Remove from Load Balancer**.
6. When prompted for confirmation, click **Yes, Remove**.
7. Stop the instance. For more information, see [Stop and Start Your Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

When you are ready, you can register the instance with your load balancer as follows.

To register your back-end instances with your load balancer

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. Start the instance again. For more information, see [Stop and Start Your Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.
3. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.
4. Select your load balancer.
5. In the bottom pane, select the **Instances** tab.
6. Click **Edit Instances**.
7. Select the instance to register with your load balancer.
8. Click **Save**.

De-register and Register Your Instances Using the AWS CLI

You can de-register your back-end instance from load balancer, and then register your restarted instance with the load balancer.

To de-register your back-end instances from your load balancer

1. De-register your back-end instance from your load balancer using 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"
    }
  ]
}
```

2. Stop the instance using the following [stop-instances](#) command:

```
aws ec2 stop-instances --instance-ids i-4e05f721
```

When you are ready, you can register the instance with your load balancer as follows.

To register your back-end instances with your load balancer

1. Start the instance again using the following [start-instances](#) command:

```
aws ec2 start-instances --instance-ids i-4e05f721
```

2. Register your back-end instance with your load balancer using the following [register-instances-with-load-balancer](#) command:

```
aws elb register-instances-with-load-balancer --load-balancer-name my-load  
balancer --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"
    }
  ]
}
```

Listeners for Your Load Balancer

Before you start using Elastic Load Balancing, you must configure one or more *listeners* for your 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 all ports in the ephemeral port range (1-65535).

Load balancers can listen on the following ports: 25, 80, 443, 465, 587, and 1024-65535.

Contents

- [Protocols \(p. 70\)](#)
- [HTTPS/SSL Listeners \(p. 72\)](#)
- [Listener Configurations Quick Reference \(p. 72\)](#)
- [X-Forwarded Headers \(p. 74\)](#)

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 health check configuration. 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.

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 Support for Your Load Balancer](#) (p. 83).

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 and terminates the connection before re-sending the request to the back-end instances.

To connect with the back-end instances, an HTTP GET request or an HTTPS GET request is issued to the instance on the ping port and the ping path specified in the health check configuration. If the load balancer receives any response other than "200 OK" within the response timeout period, the instance is considered unhealthy.

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](#) (p. 74).

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 Load Balancer](#) (p. 87).

Not all HTTP extensions are supported in the load balancer. In some cases, you will 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 an HTTPS load balancer with the following security features.

SSL Server Certificates

If you use HTTPS or SSL for your front-end listener, you must install an X.509 certificate (SSL server 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 back-end instances. You must create the certificate, get the certificate signed by a Certificate Authority (CA), and upload the certificate using AWS Identity and Access Management (IAM). For more information, see [SSL Certificates for Elastic Load Balancing](#) (p. 20).

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 Elastic Load Balancing](#) (p. 26).

Back-End Server Authentication

If want to use SSL, but don't want to terminate the connection on your load balancer, you can use TCP for connections from the client to your load balancer, use the SSL protocol for connections from the load balancer to your back-end application, and install certificates on all the back-end instances handling requests.

If you use an HTTPS/SSL connection for your back end, you can enable authentication on your back-end instance. This authentication can be used to ensure that back-end instances accept only encrypted communication, and to ensure that the back-end instance has the correct certificates.

You can install any certificate on your back-end instances, including a self-signed certificate.

For more information, see [Configure Back-end Server Authentication](#) (p. 42).

Listener Configurations Quick Reference

The following tables summarizes the listener settings that you can use to configure your load balancer.

HTTP/HTTPS 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	<ul style="list-style-type: none"> Supports the X-Forward-For (p. 74) header
Secure website or application using Elastic Load Balancing to offload SSL decryption	HTTPS	SSL negotiation (p. 26)	HTTP	NA	<ul style="list-style-type: none"> Supports the X-Forward-For (p. 74) header Requires an SSL certificate (p. 20) installed on the load balancer
Secure website or application using end-to-end encryption	HTTPS	SSL negotiation (p. 26)	HTTPS	Back-end authentication	<ul style="list-style-type: none"> Supports the X-Forward-For (p. 74) header Requires SSL certificates (p. 20) installed on the load balancer and the registered instances

TCP/SSL 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	<ul style="list-style-type: none"> Supports the Proxy Protocol (p. 83) header

Use Case	Front-End Protocol	Front-End Options	Back-End Protocol	Back-End Options	Notes
Secure website or application using Elastic Load Balancing to offload SSL decryption	SSL	SSL negotiation (p. 26)	TCP	NA	<ul style="list-style-type: none">• Supports the Proxy Protocol (p. 83) header• Requires an SSL certificate (p. 20) installed on the load balancer
Secure website or application using end-to-end encryption with Elastic Load Balancing	SSL	SSL negotiation (p. 26)	SSL	Back-end authentication	<ul style="list-style-type: none">• Supports the Proxy Protocol (p. 83) header• Requires SSL certificates (p. 20) installed on the load balancer and the registered instances

X-Forwarded Headers

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 that are widely used by the applications. Some of the non-standard HTTP headers have a `X-Forwarded` prefix. Elastic Load Balancing supports the following `X-Forwarded` headers.

Headers

- [X-Forwarded-For \(p. 74\)](#)
- [X-Forwarded-Proto \(p. 75\)](#)
- [X-Forwarded-Port \(p. 75\)](#)

X-Forwarded-For

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

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


```
X-Forwarded-For: clientIPAddress
```

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
```

If a request goes through multiple proxies, the *clientIPAddress* in the X-Forwarded-For request header is followed by the IP addresses of each successive proxy that the request goes through before it reaches your load balancer. Therefore, the right-most IP address is the IP address of the most recent proxy and the left-most IP address is the IP address of the originating client. In such cases, the X-Forwarded-For request header takes the following form:

```
X-Forwarded-For: OriginatingClientIPAddress, proxy1-IPAddress, proxy2-IPAddress
```

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 server. 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 port that an HTTP or HTTPS load balancer uses to connect to the client.

Configure Your Load Balancer

Contents

- [Configure the Idle Connection Timeout for Your Load Balancer \(p. 76\)](#)
- [Configure Cross-Zone Load Balancing for Your Load Balancer \(p. 77\)](#)
- [Configure Connection Draining for Your Load Balancer \(p. 80\)](#)
- [Configure Proxy Protocol Support for Your Load Balancer \(p. 83\)](#)
- [Configure Sticky Sessions for Your Load Balancer \(p. 87\)](#)
- [Tag Your Load Balancer \(p. 91\)](#)
- [Configure a Custom Domain Name for Your Load Balancer \(p. 94\)](#)

Configure the Idle Connection Timeout for Your Load Balancer

For each request that a client makes through a load balancer, the load balancer maintains two connections. One connection is with the client and the other connection is to the back-end instance. For each connection, the load balancer manages an idle timeout that is triggered when no data is sent over the connection for a specified time period. After the idle timeout period has elapsed, if no data has been sent or received, the load balancer closes the connection.

By default, Elastic Load Balancing sets the idle timeout to 60 seconds for both connections. If an HTTP request doesn't complete within the idle timeout period, the load balancer closes the connection, even if data is still being transferred. You can change the idle timeout setting for the connections to ensure that lengthy operations, such as file uploads, have time to complete.

If you use HTTP and HTTPS listeners, we recommend that you enable the keep-alive option for your EC2 instances. You can enable keep-alive in your web server settings or in the kernel settings for your EC2 instances. Keep-alive, when enabled, enables the load balancer to re-use connections to your back-end instance, which reduces the CPU utilization. To ensure that the load balancer is responsible for closing the connections to your back-end instance, make sure that the value you set for the keep-alive time is greater than the idle timeout setting on your load balancer.

Contents

- [Configure the Idle Timeout Using the Console \(p. 77\)](#)
- [Configure the Idle Timeout Using the AWS CLI \(p. 77\)](#)

Configure the Idle Timeout Using the Console

Use the following procedure to set the idle timeout for your load balancer.

To configure the idle timeout setting for your load balancer

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.
3. Select your load balancer.
4. In the bottom pane, select the **Description** tab.
5. Find **Connection Settings**, and then click **(Edit)**.
6. In the **Configure Connection Settings** dialog box, enter a value for **Idle Timeout**. The range for the idle timeout is 1 to 3,600 seconds.
7. Click **Save**.

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 Load Balancer

To ensure that request traffic is routed evenly across all back-end instances for your load balancer, regardless of the Availability Zone that they are in, enable cross-zone load balancing on your load balancer. Cross-zone load balancing reduces the need to maintain equivalent numbers of back-end instances in each Availability Zone, and improves your application's ability to handle the loss of one or more back-end instances. However, we still recommend that you maintain approximately equivalent numbers of instances in each 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 back-end instances in the region, reducing the impact of misbehaving clients.

Contents

- [Enable Cross-Zone Load Balancing \(p. 78\)](#)

- [Disable Cross-Zone Load Balancing \(p. 79\)](#)

Enable Cross-Zone Load Balancing

You can enable cross-zone load balancing for your 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. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.
3. Select your load balancer.
4. In the bottom pane, click **(Edit)** in the **Cross-Zone Load Balancing: Disabled** row.
5. In the **Configure Cross-Zone Load Balancing** dialog box, select **Enable**.
6. Click **Save**.

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

To enable 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\":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. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.
3. Select your load balancer.
4. In the bottom pane, click **(Edit)** in the **Cross-Zone Load Balancing: Enabled** row.
5. In the **Configure Cross-Zone Load Balancing** dialog box, select **Disable**.
6. Click **Save**.

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 Load Balancer

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

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

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

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

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

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 and Disable Connection Draining Using the Console \(p. 81\)](#)
- [Enable and Disable Connection Draining Using the AWS CLI \(p. 81\)](#)

Enable and Disable Connection Draining Using the Console

You can enable or disable 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. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.
3. Select your load balancer.
4. In the bottom pane, select the **Instances** tab.
5. Click **(Edit)** next to **Connection Draining**.
6. In the **Configure Connection Draining** dialog box, select **Enable Connection Draining**.
7. (Optional) In the **Timeout** field, enter the maximum time limit. This is a value between 1 and 3,600 seconds.
8. Click **Save**.

To disable connection draining using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.
3. Select your load balancer.
4. In the bottom pane, select the **Instances** tab.
5. Click **(Edit)** next to **Connection Draining**.
6. In the **Configure Connection Draining** dialog box, deselect **Enable Connection Draining**.
7. Click **Save**.

Enable and Disable Connection Draining Using the AWS CLI

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

To enable connection draining, set the `ConnectionDraining` attribute of your load balancer to `true`. You can optionally specify a timeout value.

To enable connection draining 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 "{\"ConnectionDraining\":{\"Enabled\":true,\"Timeout\":300}}"
```

The following is an example response:

```
{
  "LoadBalancerAttributes": {
    "ConnectionDraining": {
```

Elastic Load Balancing Developer Guide

Enable and Disable Connection Draining Using the AWS CLI

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

2. (Optional) Use the following [describe-load-balancer-attributes](#) command to verify that connection draining 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": true,  
      "Timeout": 300  
    },  
    "CrossZoneLoadBalancing": {  
      "Enabled": false  
    },  
    "ConnectionSettings": {  
      "IdleTimeout": 60  
    },  
    "AccessLog": {  
      "Enabled": false  
    }  
  },  
}
```

To disable connection draining, set the `ConnectionDraining` attribute of your load balancer to `false`.

To disable connection draining 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 "{\"ConnectionDraining\":{\"Enabled\":false}}"
```

The following is an example response:

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


2. Use the following [describe-load-balancer-attributes](#) command to verify that connection draining 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": false
    },
    "ConnectionSettings": {
      "IdleTimeout": 60
    },
    "AccessLog": {
      "Enabled": false
    }
  }
}
```

Configure Proxy Protocol Support for Your 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) or Secure Sockets Layer (SSL) for both front-end and back-end connections, your load balancer forwards requests to the back-end 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 back-end instance as part of the request.

You can enable Proxy Protocol on ports that use either the SSL and TCP protocols. You can use Proxy Protocol to capture the source IP of your client when you are using a non-HTTP protocol, or when you are using HTTPS and not terminating the SSL connection on your load balancer.

Note

The AWS Management Console does not support enabling Proxy Protocol.

Contents

- [Proxy Protocol Header \(p. 84\)](#)
- [Prerequisites for Enabling Proxy Protocol \(p. 84\)](#)
- [Enable Proxy Protocol Using the AWS CLI \(p. 84\)](#)
- [Disable Proxy Protocol Using the AWS CLI \(p. 86\)](#)

Proxy Protocol Header

The Proxy Protocol header helps you identify the IP address of a client when you use a load balancer configured for TCP/SSL connections. Because load balancers intercept traffic between clients and your back-end instances, the access logs from your back-end 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 outside of the EC2-Classical network.

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
```

Example: IPv6 (EC2-Classical only)

The following is an example of the IPv6 Proxy Protocol line for IPv6.

```
PROXY TCP6 2001:DB8::21f:5bff:febf:ce22:8a2e 2001:DB8::12f:8baa:eaef:ce29:6b2e  
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 back-end instance is configured, this duplication might result in errors.
- Confirm that your back-end instances can process the Proxy Protocol information.

Enable Proxy Protocol Using the AWS CLI

To enable Proxy Protocol, you need to create a policy of type `ProxyProtocolPolicyType` and then enable the policy on the back-end 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 back-end 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 ProxyProtocolPoli
cyType --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 and any policies that are currently enabled.

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

```
aws elb set-lb-policies-for-backend-server my-loadbalancer --instance-port 80 --policy-names my-SSLNegotiation-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": [],
      ...
    }
  ]
}
```

```
}  
]
```

Configure Sticky Sessions for Your Load Balancer

By default, a 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`.

Contents

- [Duration-Based Session Stickiness \(p. 87\)](#)
- [Application-Controlled Session Stickiness \(p. 89\)](#)

Duration-Based Session Stickiness

The load balancer uses a special cookie to track the instance for each request to each listener. When the load balancer receives a request, it first checks to see if this cookie is present in the request. If so, the request is sent to the instance specified in the cookie. If there is no cookie, the load balancer chooses an instance based on the existing load balancing algorithm. A cookie is inserted into the response for binding subsequent requests from the same user to that instance. The stickiness policy configuration defines a cookie expiration, which establishes the duration of validity for each cookie. The cookie is automatically updated after its duration expires.

If an instance fails or becomes unhealthy, the load balancer stops routing request to that instance, and chooses a new 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 different listener, 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. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.
3. Select your load balancer.
4. In the **Description** tab, find **Port Configuration**, and then click **(Edit)**, which is next to the port.
5. In the **Edit stickiness** dialog box, click **Enable Load Balancer Generated Cookie Stickiness**.
6. In **Expiration Period**, enter the cookie expiration period, in seconds.
7. Click **Save**.

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 my-ProxyProtocol-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 to enable the policies, use the `--policy-names` option to list all policies that you want to enable for the port.

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-2015-05"
  ]
},
...
],
...
"Policies": {
  "LBCookieStickinessPolicies": [
    {
      "PolicyName": "my-duration-cookie-policy",
      "CookieExpirationPeriod": 60
    }
  ],
  "AppCookieStickinessPolicies": [],
  "OtherPolicies": [
    "ELBSecurityPolicy-2015-05"
  ]
},
...
}
]
```

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.

If an instance fails or becomes unhealthy, the load balancer stops routing request to that instance, instead 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. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.
3. Select your load balancer.
4. In the **Description** tab, find **Port Configuration**, and then click **(Edit)**, which is next to the port.
5. In the **Edit stickiness** dialog box, click **Enable Application Generated Cookie Stickiness**.
6. In **Cookie Name**, enter the name of your application cookie.
7. Click **Save**.

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 to enable the policies, use the `--policy-names` option to list all the policies you want to enable for the port.

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-2015-05"
          ]
        },
        {
          "Listener": {
            "InstancePort": 80,
            "LoadBalancerPort": 80,
            "Protocol": "TCP",
            "InstanceProtocol": "TCP"
          },
          "PolicyNames": []
        }
      ]
    }
  ]
}
```



```
    },
    ],
    ...
    "Policies": {
        "LBCookieStickinessPolicies": [],
        "AppCookieStickinessPolicies": [
            {
                "PolicyName": "my-app-cookie-policy",
                "CookieName": "my-app-cookie"
            }
        ],
        "OtherPolicies": [
            "ELBSecurityPolicy-2015-05"
        ]
    },
    ...
}
]
```

Tag Your Load Balancer

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

You can add up to ten tags to each 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 \(p. 91\)](#)
- [Add and Remove Tags Using the Console \(p. 92\)](#)
- [Add and Remove Tags Using the AWS CLI \(p. 93\)](#)

Tag Restrictions

The following basic restrictions apply to tags:

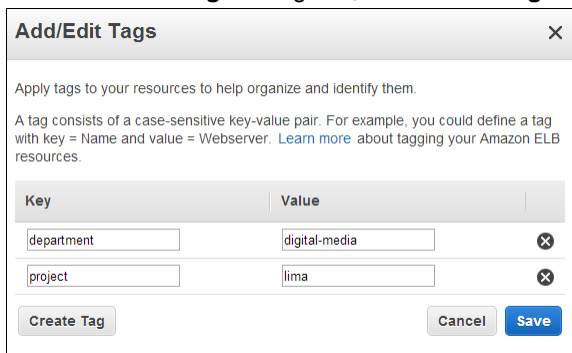
- Maximum number of tags per resource—10
- Maximum key length—127 Unicode characters
- Maximum value length—255 Unicode characters
- Tag keys and values are case sensitive. Allowed characters are letters, spaces, and numbers representable in UTF-8, plus the following special characters: + - . _ : / @. 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 and Remove Tags Using the Console

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

To add a tag to your load balancer

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.
3. Select your load balancer.
4. In the bottom pane, select the **Tags** tab.
5. Click **Add/Edit Tags**.
6. In the **Add/Edit Tags** dialog box, click **Create Tag** and specify a key and a value for each tag.



Add/Edit Tags [X]

Apply tags to your resources to help organize and identify them.

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. [Learn more](#) about tagging your Amazon ELB resources.

Key	Value	
department	digital-media	[X]
project	lima	[X]

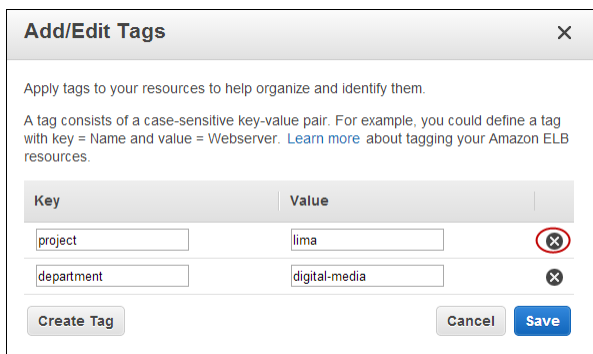
Create Tag Cancel Save

7. After you are done adding tags, click **Save**.

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

To remove a tag from your load balancer

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.
3. Select your load balancer.
4. In the bottom pane, select the **Tags** tab.
5. Click **Add/Edit Tags**.
6. In the **Add/Edit Tags** dialog box, click the remove icon of the tag you want to remove.



Add/Edit Tags [X]

Apply tags to your resources to help organize and identify them.

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. [Learn more](#) about tagging your Amazon ELB resources.

Key	Value	
project	lima	[X]
department	digital-media	[X]

Create Tag Cancel Save

7. After you are done removing the tags, click **Save**.

Add and Remove Tags Using the AWS CLI

To add a tag, use the following [add-tags](#) command:

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

Use the following [describe-tags](#) command to verify that the new tags are set:

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

The following is an example response:

```
{
  "TagDescriptions": [
    {
      "Tags": [
        {
          "Value": "lima",
          "Key": "project"
        },
        {
          "Value": "digital-media",
          "Key": "department"
        }
      ],
      "LoadBalancerName": "my-loadbalancer"
    }
  ]
}
```

To remove a tag, use the following [remove-tags](#) command:

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

Use the following [describe-tags](#) command to verify that the tag is removed:

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

The following is an example response:

```
{
  "TagDescriptions": [
    {
      "Tags": [
        {
          "Value": "digital-media",
          "Key": "department"
        }
      ],
      "LoadBalancerName": "my-loadbalancer"
    }
  ]
}
```

```
}  
}
```

Configure a Custom Domain Name for Your Load Balancer

Each 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 back-end 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 request is placed to your load balancer using this custom domain name, the DNS server resolves to the DNS name for your load balancer.

Contents

- [Associating Your Custom Domain Name with Your Load Balancer Name \(p. 94\)](#)
- [Configure DNS Failover for Your Load Balancer \(p. 95\)](#)
- [Disassociating Your Custom Domain Name from Your Load Balancer \(p. 96\)](#)

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 Amazon Route 53](#) in the *Amazon Route 53 Developer Guide*.
- For a list of accredited registrars, see the [Accredited Registrar Directory](#).

If you have a domain name but are using another DNS service, such as your domain registrar, consider the option to use Amazon Route 53 as your DNS service. When you use Amazon Route 53 as your DNS service, you can create an alias resource record set, which has the following advantages over other DNS services for routing DNS queries to your load balancer:

- Amazon Route 53 doesn't charge for DNS queries for alias resource record sets.
- 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`). If you're using a different DNS service, you need to create a CNAME resource record set to route queries to your load balancer, but DNS doesn't allow you to create a CNAME resource record set for the zone apex. (Note that some DNS services provide a workaround.)

For information about transferring DNS services for existing domains to Amazon Route 53, see [Configuring Amazon Route 53 as Your DNS Service](#) in the *Amazon Route 53 Developer Guide*.

For information about using Amazon Route 53 with your load balancer, see [Routing Queries to an Elastic Load Balancing Load Balancer](#) in the *Amazon Route 53 Developer Guide*.

Configure DNS Failover for Your Load Balancer

When you use Amazon Route 53 to route DNS queries to your load balancer, you can configure DNS failover for your load balancer. In a failover configuration, Amazon 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, Amazon 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. Amazon Route 53 directs traffic to the primary load balancer if it is available, or to the secondary load balancer otherwise.

To configure DNS failover for two load balancers using Amazon Route 53

1. Open the Amazon Route 53 console at <https://console.aws.amazon.com/route53/>.
2. In the navigation pane, choose **Hosted Zones**.
3. Select the name of your hosted zone.
4. Click **Create Record Set**.
5. In **Name**, the default value is the name of your domain (for example, `example.com`). To route DNS queries for a subdomain (for example, `apex.example.com`) to your load balancer, specify the name of the subdomain.
6. In **Type**, select **A - Ipv4 address**.
7. For **Alias**, click **Yes**.
8. Click **Alias Target**, and choose your primary load balancer from the list.

The value of **Alias Hosted Zone ID** appears automatically based on the load balancer that you chose for **Alias Target**.
9. For **Routing Policy**, select **Failover**.
10. For **Failover Record Type**, select **Primary**.
11. For **Set ID**, enter an ID for the record set or use the default value.
12. For **Evaluate Target Health**, select **Yes**.
13. For **Associate with Health Check**, select **No**.
14. Click **Create**.
15. Repeat the same steps to create an alias record set for your secondary load balancer, with the following exceptions:
 - In **Alias Target**, choose your secondary load balancer.
 - For **Failover Record Type**, select **Secondary**.
 - For **Evaluate Target Health**, select **Yes** to evaluate the health of the secondary load balancer. If the secondary load balancer is unhealthy, Amazon Route 53 routes traffic to the primary load balancer. If you select **No**, Amazon Route 53 assumes that the secondary load balancer is healthy and routes traffic to it whenever the primary load balancer is unhealthy.

For information about setting up advanced DNS failover configurations, see [Configuring Amazon Route 53 Active-Active and Active-Passive 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 [Creating, Changing, and Deleting Resource Record Sets](#) and [Deleting a Hosted Zone](#) in the *Amazon Route 53 Developer Guide*.

Monitor Your 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 [Monitor Your Load Balancer Using Amazon CloudWatch \(p. 97\)](#).

Elastic Load Balancing access logs

The access logs for Elastic Load Balancing capture detailed information for all 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 [Monitor Your Load Balancer Using Elastic Load Balancing Access Logs \(p. 107\)](#).

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 Elastic Load Balancing API Calls Using AWS CloudTrail \(p. 116\)](#).

Monitor Your Load Balancer Using Amazon CloudWatch

Elastic Load Balancing publishes data points to Amazon CloudWatch about 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. Each data point has an associated time stamp and (optionally) a unit of measurement. For example, total number of healthy EC2 instances behind a load balancer over a specified time period can be a metric.

CloudWatch provides statistics based on the metric data points published by Elastic Load Balancing. Statistics are metric data aggregations over specified periods of time. The following statistics are available: Minimum (min), Maximum (max), Sum, Average, and Count. When you request statistics, the returned data stream is identified by the metric name and a 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.

One purpose for monitoring metrics is to verify that your system is performing as expected. If a metric goes outside what you consider an acceptable range, you can create a CloudWatch alarm to watch over a specified metric and initiate an action (for example, send a notification in email) if the metric goes outside of the specified range.

For more information about Amazon CloudWatch, see the [Amazon CloudWatch Developer Guide](#).

Contents

- [CloudWatch Metrics for Elastic Load Balancing \(p. 98\)](#)
- [Statistics for Elastic Load Balancing Metrics \(p. 100\)](#)
- [View CloudWatch Metrics for Your Load Balancer \(p. 105\)](#)
- [Create CloudWatch Alarms for Your Load Balancer \(p. 106\)](#)

CloudWatch Metrics for Elastic Load Balancing

Elastic Load Balancing sends metrics for all load balancers associated with your AWS account to Amazon CloudWatch. By default, CloudWatch uses these metrics to provide detailed monitoring for your load balancers; you do not need to explicitly enable detailed monitoring.

Elastic Load Balancing Metrics

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.

Note that not every statistic available through CloudWatch applies to every metric for Elastic Load Balancing, though they are all available. For each metric, be aware of its preferred statistic so that you can track the most useful information.

The following CloudWatch metrics are available for Elastic Load Balancing.

Metric	Description
HealthyHostCount	<p>The number of healthy instances in each Availability Zone. An instance is considered healthy if it meets the healthy threshold configured for the health checks. If cross-zone load balancing is enabled, the number of healthy instances is calculated across all Availability Zones.</p> <p>Reporting criteria: There are registered instances</p> <p>Preferred statistic: average</p>

Metric	Description
UnHealthyHostCount	<p>The number of unhealthy instances in each Availability Zone. An instance is considered unhealthy if it exceeds the unhealthy threshold configured for the health checks. If cross-zone load balancing is enabled, the number of unhealthy instances is calculated across all Availability Zones.</p> <p>Reporting criteria: There are registered instances</p> <p>Preferred statistic: <code>average</code></p>
RequestCount	<p>The number of completed requests that were received and routed to the registered instances.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Preferred statistic: <code>sum</code></p>
Latency	<p>The time elapsed, in seconds, after the request leaves the load balancer until a response is received.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Preferred statistic: <code>average</code></p>
HTTPCode_ELB_4XX	<p>The number of HTTP 4XX client error codes generated by the load balancer when the listener is configured to use the HTTP or HTTPS protocol. Client errors are generated when a request is malformed or incomplete.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Preferred statistic: <code>sum</code></p>
HTTPCode_ELB_5XX	<p>The number of HTTP 5XX server error codes generated by the load balancer when the listener is configured to use the HTTP or HTTPS protocol. This does not include any response codes generated by registered instances.</p> <p>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 or the load balancer.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Preferred statistic: <code>sum</code></p>
HTTPCode_Backend_2XX HTTPCode_Backend_3XX HTTPCode_Backend_4XX HTTPCode_Backend_5XX	<p>The number of HTTP response codes generated by registered instances. This does not include any response codes generated by the load balancer.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Preferred statistic: <code>sum</code></p>
BackendConnectionErrors	<p>The number of connections that were not successfully established 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.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Preferred statistic: <code>sum</code></p>
SurgeQueueLength	<p>The total number of requests that are pending submission to a registered instance.</p> <p>Reporting criteria: There is a nonzero value</p> <p>Preferred statistic: <code>max</code></p>

Metric	Description
SpilloverCount	The total number of requests that were rejected because the queue was full. Reporting criteria: There is a nonzero value Preferred statistic: sum

Dimensions for Elastic Load Balancing Metrics

To refine the metrics returned by a request, you can use the following dimensions supported by Elastic Load Balancing.

Dimension	Description
LoadBalancerName	Limits the metric data to instances that are registered to the specified load balancer.
AvailabilityZone	Limits the metric data to load balancers in the specified Availability Zone.

For example, with the `HealthyHostCount` metric, you can use the `LoadBalancerName` and `AvailabilityZone` dimensions to get the average number of healthy instances for the specified load balancer in the specified Availability Zone for a specific period of time. Alternatively, you can track the minimum number of healthy hosts or the maximum number of unhealthy hosts in order to better understand how the health and the count of your back-end instances change over time.

Statistics for Elastic Load Balancing Metrics

Your load balancer is made up of load balancer nodes that forward traffic to your back-end instances. Each load balancer node routes traffic to instances within a single Availability Zone and reports metrics for the Availability Zone.

For all CloudWatch metrics, `min` and `max` represent the `min` and `max` as reported by individual load balancer nodes. For example, suppose there are 2 load balancer nodes. Node 1 reports a `HealthyHostCount` with a `min` of 2, a `max` of 10, and an `average` of 6. Load balancer node 2 reports a `HealthyHostCount` with a `min` of 1, `max` of 5, and an `average` of 3. This means that the `min` in CloudWatch is 1, the `max` is 10, and the `average` is around 4.

The `sum` is the aggregate value reported across all load balancer nodes during the given time period. Because the 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_4XX`, `HTTPCode_ELB_5XX`, `HTTPCode_Backend_XXX`, `BackendConnectionErrors`, and `SpilloverCount`.

The `count` is the number of samples measured. Because the metrics are gathered based on sampling intervals and events, the `count` is typically not useful. For example, in the healthy/unhealthy host metrics, `count` is based on the number of health check samples collected by the load balancer nodes, not the number of healthy/unhealthy hosts. For latency metrics, `count` is the number of samples that each load balancer node reports, not the actual value of latency reported.

The following table describes how to evaluate the statistics for the CloudWatch metrics sent by Elastic Load Balancing.

Statistics for Elastic Load Balancing Metrics

Metric Name	Statistics Details
HealthyHostCount UnHealthyHostCount	<p>This metric should be used with <code>AvailabilityZone</code> dimension.</p> <p>Preferred statistic: <code>average</code></p> <p>The <code>average</code> statistic indicates the average number of healthy or unhealthy instances seen by the load balancer nodes. To get the total healthy or unhealthy instances, calculate the average value for each Availability Zone.</p> <p>The <code>min</code> and <code>max</code> statistics represent the smallest and largest number of instances that were in the healthy or unhealthy state during the specified interval. Note that some load balancer nodes might see an instance as unhealthy for a brief period while other nodes see it as healthy, so <code>min</code> and <code>max</code> can be misleading.</p> <p>The <code>sum</code> statistic is not meaningful.</p> <p>The <code>count</code> statistic is the number of samples reported by all load balancer nodes, and is not a useful measure for troubleshooting issues.</p> <p>Example: Suppose that your load balancer has 4 back-end instances with 2 instances in <code>us-west-2a</code> and 2 instances in <code>us-west-2b</code>, that <code>us-west-2a</code> has 1 unhealthy instance and 1 healthy instance, and that <code>us-west-2b</code> has 2 healthy and 0 unhealthy instances. The AZ dimension reports an average of 1 healthy and 1 unhealthy instance in <code>us-west-2a</code>, and an average of 2 healthy and 0 unhealthy instances in <code>us-west-2b</code>. The regional dimension reports an average of 1.5 healthy instances and 0.5 unhealthy instances.</p>
RequestCount	<p>This metric is typically used with the <code>LoadBalancerName</code> dimension to view the total requests for a load balancer. It can also be used to measure the number of requests that were routed to an Availability Zone (which might not be the same Availability Zone that serviced the request for the back end).</p> <p>Preferred statistic: <code>sum</code></p> <p>The <code>sum</code> statistic is the only meaningful statistic for this measure.</p> <p>The <code>min</code>, <code>max</code>, and <code>average</code> statistics are not meaningful and all return a value of 1.</p> <p>The <code>count</code> statistic is the number of samples reported by all load balancer nodes and typically equals the <code>sum</code> for the period.</p> <p>Example: Suppose that your load balancer has 4 back-end instances with 2 instances in <code>us-west-2a</code> and 2 instances in <code>us-west-2b</code>, and that 100 requests are sent to the load balancer. There are 60 requests sent to <code>us-west-2a</code>, with each instance receiving 30 requests, and 40 requests sent to <code>us-west-2b</code>, with each instance receiving 20 requests. The AZ dimension reports a sum of 60 requests in <code>us-west-2a</code> and 40 requests in <code>us-west-2b</code>. The regional dimension reports a sum of 100 requests.</p>

Metric Name	Statistics Details
Latency	<p>Latency can be viewed for all requests or for requests routed to a single Availability Zone.</p> <p>Preferred statistic: <code>average</code></p> <p>The <code>average</code> statistic is the most useful diagnostic because it is the average of all requests sent to the back end.</p> <p>The <code>max</code> statistic can be useful to determine whether some requests are taking substantially longer than the average.</p> <p>The <code>min</code> statistic is typically not a useful measure, because it is the request/response with the lowest total time elapsed.</p> <p>The <code>count</code> statistic is approximately equal to the <code>sum</code> for the <code>Request-Count</code> metric, because it is the number of samples taken.</p> <p>Example: Suppose that your load balancer has 4 back-end instances with 2 instances in us-west-2a and 2 instances in us-west-2b. Requests sent to 1 instance in us-west-2a have a higher latency. The latency metric reported for us-west-2a has a higher value than the latency metric for us-west-2b.</p>
HTTPCode_ELB_4XX	<p>This metric is typically used with the <code>LoadBalancerName</code> dimension to view the total number of HTTP 4XX errors generated by the load balancer nodes. It can also be used to measure the number of errors for requests that were routed to an Availability Zone (which might not be the same Availability Zone that serviced the request for the back-end).</p> <p>Preferred statistic: <code>sum</code></p> <p>The <code>sum</code> statistic is the only meaningful statistic for this measure.</p> <p>The <code>min</code>, <code>max</code>, and <code>average</code> statistics are not meaningful and all return a value of 1.</p> <p>The <code>count</code> statistic is the number of samples reported by all load balancer nodes and typically equals the <code>sum</code> for the period.</p> <p>Example: Suppose that your load balancer has us-west-2a and us-west-2b enabled. Client requests include a malformed request URL. Client HTTP errors would likely increase in all Availability Zones. The regional metric would be the sum of the values for each Availability Zone.</p>

Metric Name	Statistics Details
HTTPCode_ELB_5XX	<p>This metric is typically used with the <code>LoadBalancerName</code> dimension to view the total number of HTTP 5XX errors generated by all the load balancer nodes. It can also be used to measure the number of errors for requests that were routed to an Availability Zone (which might not be the same Availability Zone that serviced the request for the back end).</p> <p>Preferred statistic: <code>sum</code></p> <p>The <code>sum</code> statistic is the only meaningful statistic for this measure.</p> <p>The <code>min</code>, <code>max</code>, and <code>average</code> statistics are not meaningful and all return a value of 1.</p> <p>The <code>count</code> statistic is the number of samples reported by all load balancer nodes and typically equals the <code>sum</code> for the period.</p> <p>Example: Suppose that your load balancer has <code>us-west-2a</code> and <code>us-west-2b</code> enabled. Instances in <code>us-west-2a</code> are experiencing high latency and are slow to respond to requests. As a result, the load balancer nodes' surge queue in <code>us-west-2a</code> fills up, resulting in spillovers and returns a 503 error to clients (see the <code>SpilloverCount</code> metric). If <code>us-west-2b</code> continues to respond normally, the regional <code>sum</code> of <code>HTTPCode_ELB_5XX</code> equals the <code>sum</code> for <code>us-west-2a</code>.</p>
HTTPCode_Backend_2XX HTTPCode_Backend_3XX HTTPCode_Backend_4XX HTTPCode_Backend_5XX	<p>This metric is typically used with the <code>LoadBalancerName</code> dimension to view the total number of HTTP response codes generated by back-end instances registered to a load balancer. It can also be used to measure the number of errors for requests that were routed to an Availability Zone (which might not be the same Availability Zone that serviced the request for the back end).</p> <p>Preferred statistic: <code>sum</code></p> <p>The <code>sum</code> statistic is the only meaningful statistic for this measure.</p> <p>The <code>min</code>, <code>max</code>, and <code>average</code> statistics are not meaningful and all return a value of 1.</p> <p>The <code>Count</code> statistic is the number of samples reported by all load balancer nodes and typically equals the <code>sum</code> for the period.</p> <p>Example: Suppose that your load balancer has 4 back-end instances with 2 instances in <code>us-west-2a</code> and 2 instances in <code>us-west-2b</code>. Requests sent to 1 instance in <code>us-west-2a</code> result in an HTTP 500 response. The metric reported for <code>us-west-2a</code> includes these error responses, while the metric in <code>us-west-2b</code> does not include these error responses. The regional total is equal to the total for <code>us-west-2a</code>.</p>

Metric Name	Statistics Details
BackendConnectionErrors	<p>This metric can be viewed for all back-end instances or the back-end instances in a single Availability Zone.</p> <p>Preferred statistic: <code>sum</code></p> <p>The <code>sum</code> statistic represents the total connection errors seen by all load balancer nodes for the given period of time.</p> <p>The <code>sum</code> statistic is the only meaningful statistic for this measure.</p> <p>The <code>average</code>, <code>min</code>, and <code>max</code> statistics are reported per load balancer node and are not typically useful. The difference between <code>min</code> and <code>max</code> or peak to average or average to trough might be useful to determine whether a single load balancer node is an outlier.</p> <p>The <code>count</code> statistic is the number of samples reported by all load balancer nodes and typically equals the sum for the period.</p> <p>Example: Suppose that your load balancer has 4 back-end instances with 2 instances in us-west-2a and 2 instances in us-west-2b. Attempts to connect to 1 instance in us-west-2a result in an increase in the back-end connection errors. The metric reported for us-west-2a includes these connection errors for the failed attempts, while the metric for us-west-2b does not include these errors. The regional total is equal to the total for us-west-2a.</p>
SurgeQueueLength	<p>This metric can be used to monitor the surge queue size for a single Availability Zone or for the region (the overall load balancer).</p> <p>Preferred statistic: <code>max</code></p> <p>The <code>max</code> statistic is the most useful statistic because it represents the peak of requests that were queued. The maximum value is 1,024. If any load balancer node has a full queue of 1,024 requests, there will likely be spillovers (see the <code>SpilloverCount</code> metric).</p> <p>The <code>average</code> statistic represents the number of requests that were in the queue on average across all load balancer nodes for a given period of time. It can be useful in combination with <code>min</code> and <code>max</code> to determine the range of queuing by load balancer nodes.</p> <p>The <code>sum</code> statistic is not a useful statistic, because it is the total of all recorded samples across all load balancer nodes.</p> <p>The <code>min</code> statistic is not typically useful because it represents the lowest value observed for any load balancer node for any sample.</p> <p>The <code>count</code> statistic is not a meaningful statistic for this measure.</p> <p>Example: Suppose that your load balancer has us-west-2a and us-west-2b enabled. Instances in us-west-2a are experiencing high latency and are slow to respond to requests. As a result, the load balancers' surge queue in us-west-2a fills up, with clients likely experiencing increased response times. If this continues, the load balancer will likely have spillovers (see the <code>SpilloverCount</code> metric). If us-west-2b continues to respond normally, the regional <code>max</code> will be the same as the <code>max</code> for us-west-2a, while us-west-2b will have a small value (or no metric) for the same period.</p>

Metric Name	Statistics Details
SpilloverCount	<p>This metric is typically used with the <code>LoadBalancerName</code> dimension to view the total spillovers for the load balancer. It can also be used to measure the number of rejected requests that were routed to an Availability Zone.</p> <p>Preferred statistic: <code>sum</code></p> <p>The <code>sum</code> statistic represents the total of all load balancer nodes reports for the given period of time.</p> <p>The <code>sum</code> statistic is the only meaningful statistic for this measure.</p> <p>The <code>average</code>, <code>min</code>, and <code>max</code> statistics are reported per load balancer node and are not typically useful for the spillover metric.</p> <p>The <code>min</code> statistic is not typically useful because it represents the lowest value observed for any load balancer node for any sample.</p> <p>The <code>count</code> statistic is not a meaningful statistic for this measure.</p> <p>Example: Suppose that your load balancer has <code>us-west-2a</code> and <code>us-west-2b</code> enabled. Instances in <code>us-west-2a</code> are experiencing high latency and are slow to respond to requests. As a result, the load balancers' surge queue in <code>us-west-2a</code> fills up, resulting in load balancer spillovers. The spillover metric will be incremented. If <code>us-west-2b</code> continues to respond normally, the regional <code>sum</code> will be the same as the sum for <code>us-west-2a</code>, while <code>us-west-2b</code> will have a small value (or no metric) for the same period.</p>

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.

Note

Alternatively, you can view metrics for your load balancer using the CloudWatch console. For more information, see [Viewing Your AWS Metrics with Amazon CloudWatch](#) in the *Amazon CloudWatch Developer Guide*.

To view the metrics from your load balancer

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.
3. Select your load balancer.
4. In the bottom pane, click the **Monitoring** tab.
5. To filter the results by time, select a time range from **Showing data for**.
6. Select an individual graph to get a larger view of the individual metric. The following metrics are available:
 - Sum HTTP 2XXs
 - Sum HTTP 4XXs
 - Sum HTTP 5XXs
 - Sum ELB HTTP 4XXs
 - Sum ELB HTTP 5XXs
 - Healthy Hosts
 - Unhealthy Hosts

- Average Latency
- Sum Requests
- Backend Connection Errors
- Surge Queue Length
- Spillover Count

Create CloudWatch Alarms for Your Load Balancer

An alarm watches a single metric over the time period that you specify. Depending on the value of the metric relative to a threshold that you define, the alarm can send one or more notifications using Amazon SNS, a service that enables applications, end users, and devices to instantly send and receive notifications. For more information, see [Get Started with Amazon SNS](#).

An alarm sends notifications to Amazon SNS when the specified metric reaches the defined range and remains in that range for a specified period of time. An alarm has three possible states:

- **OK**—The value of the metric is within the range you've specified.
- **ALARM**—The value of the metric is outside the range that you've specified for the specified period of time.
- **INSUFFICIENT_DATA**—Either the metric is not yet available or there is not enough data is available for the metric to determine the alarm state.

Whenever the state of an alarm changes, CloudWatch uses Amazon SNS to send a notification to the email addresses that you specified.

Use the following procedure to create an alarm for your load balancer using the Amazon EC2 console. The alarm sends notifications to an SNS topic whenever the load balancer's latency is above 120 seconds for 1 consecutive period of 5 minutes. Note that a short period creates a more sensitive alarm, while a longer period can mitigate brief spikes in a metric.

Note

Alternately, you can create an alarm for your load balancer using the CloudWatch console. For more information, see [Send Email Based on Load Balancer Alarm](#) in the *Amazon CloudWatch Developer Guide*.

To create an alarm for your load balancer

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.
3. Select your load balancer.
4. From the **Monitoring** tab, click **Create Alarm**.
5. If you have an SNS topic that you want to use, select it from **Send a notification to**. Otherwise, create an SNS topic as follows:
 - a. Click **create topic**.
 - b. In **Send a notification to**, enter a name for your topic.
 - c. In **With these recipients**, enter the email addresses of the recipients to notify, separated by commas. You can enter up to 10 email addresses. Each recipient receives an email from Amazon SNS with a link to subscribe to the SNS topic in order to receive notifications.

6. Define the threshold for your alarm as follows. For **Whenever**, select **Average** and **Average Latency**. For **Is**, select **>** and enter 120. For **For at least**, specify 1 consecutive period of **5 minutes**.
7. In **Name of alarm**, a name is automatically generated for you. If you prefer, you can specify a different name.
8. Click **Create Alarm**.

Monitor Your Load Balancer Using Elastic Load Balancing Access Logs

Elastic Load Balancing provides access logs that capture detailed information about all 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 logging is an optional feature of Elastic Load Balancing that is disabled by default. You must explicitly enable access logging for your load balancer, and then 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.

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](#) (p. 107)
- [Access Log Entries](#) (p. 108)
- [Processing Access Logs](#) (p. 111)
- [Enable Access Logs for Your Load Balancer](#) (p. 111)
- [Disable Access Logs for Your Load Balancer](#) (p. 115)

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:

```
bucket[/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
```

bucket

The name of the S3 bucket.

prefix

The prefix (logical hierarchy) in the bucket. If you don't specify a prefix, the logs are placed at the root level of the bucket.

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:

```
s3://my-loadbalancer-logs/my-app/AWSLogs/123456789012/elasticloadbalancing/us-west-2/2014/02/15/123456789012_elasticloadbalancing_us-west-2_my-loadbalancer_20140215T2340Z_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 Simple Storage Service Developer Guide*.

Access Log Entries

Elastic Load Balancing logs all 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.

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
timestamp	The time when the load balancer received the request from the client, in ISO 8601 format.
elb	The name of the load balancer

Elastic Load Balancing Developer Guide
Access Log Entries

Field	Description
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. If the client didn't send a full request, the load balancer can't dispatch the request to a registered instance, and this value is set to -.
request_processing_time	The total time elapsed, in seconds, from the time the load balancer received the request and sent it to a registered instance. 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.
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. [TCP listener] The total time elapsed, in seconds, from the time the load balancer sent the first byte of the request to a registered instance until the instance sent back the first byte. 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.
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 back end. [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. 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.
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). [HTTP listener] The value includes the request body but not the headers. [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). [HTTP listener] The value includes the response body but not the headers. [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. [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 -.

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/ HT
TP/1.1" "curl/7.38.0" DHE-RSA-AES128-SHA TLSv1.2
```

Example TCP Entry

The following is an example log entry for a 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 EMR enables you to quickly and efficiently process vast amounts of data. For more information, see the [Amazon Elastic MapReduce Developer Guide](#).
- [Splunk](#)
- [Sumo Logic](#)

Enable Access Logs for Your 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.

Important

The bucket and your load balancer must be in the same region. The bucket can be owned by a different account than the account that owns the load balancer.

Tasks

- [Step 1: Create an S3 Bucket \(p. 111\)](#)
- [Step 2: Attach a Policy to Your S3 Bucket \(p. 112\)](#)
- [Step 3: Enable Access Logs \(p. 113\)](#)
- [Step 4: Verify that the Load Balancer Created a Test File in the S3 Bucket \(p. 115\)](#)

Step 1: Create an S3 Bucket

You can create an S3 bucket using the Amazon S3 console. If you already have a bucket and want to use it to store the access logs, skip this step and go to [Step 2: Attach a Policy to Your S3 Bucket \(p. 112\)](#) to grant Elastic Load Balancing permission to write logs to your bucket.

Tip

If you will use the console to enable access logs, you can skip this step and have Elastic Load Balancing create a bucket with the required permissions for you. If you will use the AWS CLI to enable access logs, you must create the bucket and grant the required permissions yourself.

To create an Amazon S3 bucket

1. Open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
2. Click **Create Bucket**.
3. In the **Create a Bucket** dialog box, do the following:
 - a. In the **Bucket Name** box, enter a name for your bucket (for example, `my-loadbalancer-logs`). 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 Restrictions and Limitations](#) in the *Amazon Simple Storage Service Developer Guide*.
 - b. In **Region**, select the region where you created your load balancer.
 - c. Click **Create**.

Step 2: Attach a Policy to Your S3 Bucket

After you've created or identified your S3 bucket, you must attach a policy 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 your bucket already has an attached policy, you can add the statements for the Elastic Load Balancing access log 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.

Tip

If you will use the console to enable access logs, you can skip this step and have Elastic Load Balancing create a bucket with the required permissions for you.

To attach a policy statement to your bucket

1. Open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
2. Select the bucket, and then click **Properties**.
3. Under **Permissions**, click **Add bucket policy**.
4. On the **Bucket Policy Editor** page, click **AWS Policy Generator**.
5. On the **AWS Policy Generator** page, do the following:
 - a. For **Select Type of Policy**, select **S3 Bucket Policy**.
 - b. For **Effect**, select **Allow** to allow access to the S3 bucket.
 - c. In **Principal**, specify the account ID for Elastic Load Balancing to grant Elastic Load Balancing access to the S3 bucket. Select the account ID that corresponds to the region for your load balancer and bucket.

Region	Region Name	Elastic Load Balancing Account ID
us-east-1	US East (N. Virginia)	127311923021
us-west-1	US West (N. California)	027434742980
us-west-2	US West (Oregon)	797873946194
eu-west-1	EU (Ireland)	156460612806
eu-central-1	EU (Frankfurt)	054676820928
ap-northeast-1	Asia Pacific (Tokyo)	582318560864
ap-southeast-1	Asia Pacific (Singapore)	114774131450
ap-southeast-2	Asia Pacific (Sydney)	783225319266
sa-east-1	South America (Sao Paulo)	507241528517
us-gov-west-1*	AWS GovCloud (US)	048591011584
cn-north-1**	China (Beijing)	638102146993

* This region requires a separate account. For more information, see [AWS GovCloud \(US\)](#).

** This region requires a separate account. For more information, see [China \(Beijing\)](#).

- d. For **Actions**, select `PutObject` to allow Elastic Load Balancing to store objects in the S3 bucket.
- e. For **Amazon Resource Name (ARN)**, enter the ARN of the S3 bucket in the following format:

```
arn:aws:s3:::bucket/prefix/AWSLogs/aws-account-id/*
```

You must specify the ID of the AWS account that owns the load balancer, and you should not include the hyphens. For example:

```
arn:aws:s3:::my-loadbalancer-logs/my-app/AWSLogs/123456789012/*
```

Note that if you are using `us-gov-west-1` region, use `arn:aws-us-gov:` instead of `arn:aws:` in the ARN.

- f. Click **Add Statement**, and then click **Generate Policy**.
- g. Copy the policy that is displayed in the **Policy JSON Document** page, and then click **Close**. The policy document should look similar to the following:

```
{
  "Id": "Policy1429136655940",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1429136633762",
      "Action": [
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::my-loadbalancer-logs/my-app/AWS
Logs/123456789012/*",
      "Principal": {
        "AWS": [
          "797873946194"
        ]
      }
    }
  ]
}
```

6. Go back to the **Bucket Policy Editor** page and paste the policy into the text area.
7. Click **Save** to save the policy. If **Save** is not enabled, press Enter.
8. Under **Permissions**, click **Save** to attach the policy to your bucket.

Step 3: Enable Access Logs

You can enable access logs using the AWS Management Console or the AWS CLI. Note that when you enable access logs using the console, you can have Elastic Load Balancing create the bucket for you with necessary permissions for the load balancer to write to your bucket.

Use the following example to capture and deliver logs to your S3 bucket every 60 minutes (the default interval).

To enable access logs for your load balancer using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.
3. Select your load balancer.
4. In the **Description** tab, find **Access Logs**, and then click **(Edit)**.
5. In the **Configure Access Logs** dialog box, do the following:
 - a. Select **Enable Access Logs**.
 - b. Leave **Interval** set to the default setting, 60 minutes.
 - c. In **S3 Location**, enter the name of your S3 bucket, including the prefix (for example, `my-loadbalancer-logs/my-app`).

Tip

If you want Elastic Load Balancing to create the bucket, you must specify a name that is 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 Restrictions and Limitations](#) in the *Amazon Simple Storage Service Developer Guide*.

- d. (Optional) If you want Elastic Load Balancing to create the bucket, select **Create the location for me**.
- e. Click **Save**.

To enable 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": "my-loadbalancer-logs",
    "EmitInterval": 60,
    "S3BucketPrefix": "my-app"
  }
}
```

To enable access logs, 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": "my-loadbalancer-logs",
      "S3BucketPrefix": "my-app"
    }
  },
}
```



```
}
  "LoadBalancerName": "my-loadbalancer"
}
```

Step 4: Verify that the Load Balancer Created a Test File in the S3 Bucket

After the access log is enabled for your load balancer, Elastic Load Balancing validates the S3 bucket and creates a test file. You can use the S3 console to verify that the test file was created.

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. From the **All Buckets** list, select your S3 bucket.
3. Navigate to the test log file. The path should be as follows:

```
my-bucket/prefix/AWSLogs/123456789012/ELBAccessLogTestFile
```

Disable Access Logs for Your Load Balancer

You can disable access logs for your load balancer at any time. After you disable access logging, your access logs remain in your Amazon S3 until you delete the them. For information about managing your S3 bucket, see [Working with Buckets](#) in the *Amazon Simple Storage Service Console User Guide*.

To disable access logging using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **LOAD BALANCING**, click **Load Balancers**.
3. Select your load balancer.
4. In the **Description** tab, find **Access Logs**, and then click **(Edit)**.
5. In the **Configure Access Logs** dialog box, select **Disable Access Logs**.
6. Click **Save**.

To disable access logging using the AWS CLI

Use the following [modify-load-balancer-attributes](#) command to disable access logging:

```
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": "my-loadbalancer-logs",
      "EmitInterval": 60,
      "Enabled": false,
      "S3BucketPrefix": "my-app"
    }
  }
}
```

```
}  
  }  
}
```

Log Elastic Load Balancing API Calls Using AWS CloudTrail

You can use AWS CloudTrail to capture all calls to the Elastic Load Balancing API made by or on behalf of your AWS account. CloudTrail stores the information as log files in an Amazon S3 bucket that you specify. CloudTrail logs all calls to the Elastic Load Balancing API, whenever you use the Elastic Load Balancing API directly, or indirectly through the AWS Management Console or AWS CLI. You can use the logs collected by CloudTrail to monitor the activity of your load balancers and determine what API call was made, what source IP address was used, who made the call, when it was made, and so on. For the complete list of Elastic Load Balancing API actions, see the [Elastic Load Balancing API Reference](#).

AWS CloudTrail only logs calls to the API. To monitor other actions for your load balancer, such as when a client makes a request to your load balancer, use access logs. For more information, see [Monitor Your Load Balancer Using Elastic Load Balancing Access Logs](#) (p. 107).

Contents

- [Configure CloudTrail Event Logging](#) (p. 116)
- [Elastic Load Balancing Event Entries in CloudTrail Log Files](#) (p. 116)

Configure CloudTrail Event Logging

CloudTrail creates log entries in each supported region separately and stores them in the Amazon S3 bucket for that region. For information about the regions supported by CloudTrail, see [Regions and Endpoints – CloudTrail](#) in the *Amazon Web Services General Reference*.

When you enable CloudTrail logging, the CloudTrail service can create an Amazon S3 bucket for you to store your log files. If you prefer one log file for all supported regions, you can aggregate the logs created across the regions to a single bucket.

For more information, see [Creating and Updating Your Trail](#) and [Aggregating CloudTrail Log Files to a Single Amazon S3 Bucket](#) in the *AWS CloudTrail User Guide*.

You can optionally configure CloudTrail to use [Amazon SNS](#) to notify you when a log file is created. CloudTrail can send notifications to you frequently, so we recommend that you use Amazon SNS with an [Amazon SQS](#) queue and handle the notifications programmatically.

There is no cost to use CloudTrail. However, the standard rates for Amazon S3 apply, as well as the standard rates for Amazon SNS and Amazon SQS, if you use them.

Elastic Load Balancing Event Entries in CloudTrail Log Files

The log files from CloudTrail contain event information in JSON format. An event record represents a single AWS API call and includes information about the requested action, such as the user that requested the action, and the date and the time of the request. The log files include events for all AWS API calls for your AWS account, not just Elastic Load Balancing API calls. However, you can read the log files and scan for calls to the Elastic Load Balancing API using the `eventSource` element with the value

Elastic Load Balancing Developer Guide

Elastic Load Balancing Event Entries in CloudTrail Log Files

elasticloadbalancing.amazonaws.com. To view information about a specific ELB API, such as `CreateLoadBalancer`, scan for the API name in the `eventName` element.

The following example shows a CloudTrail log file for a user who created a load balancer and then deleted that load balancer using the ELB CLI. The CLI is identified by the `userAgent` element. The requested API calls ([CreateLoadBalancer](#) and [DeleteLoadBalancer](#)) are found in the `eventName` element for each record. Information about the user (Alice) can be found in the `userIdentity` element. For more information about the different elements and values in CloudTrail log files, see [CloudTrail Event Reference](#) in the *AWS CloudTrail User Guide*.

```
{
  Records: [
    {
      eventVersion: "1.01",
      userIdentity: {
        type: "IAMUser",
        principalId: "60505EXAMPLE",
        arn: "arn:aws:iam::123456789012:user/Alice",
        accountId: "123456789012",
        accessKeyId: "AKIAIOSFODNN7EXAMPLE"
      },
      eventTime: "2014-04-01T15:31:48Z",
      eventSource: "elasticloadbalancing.amazonaws.com",
      eventName: "CreateLoadBalancer",
      awsRegion: "us-west-2",
      sourceIPAddress: "127.0.0.01",
      userAgent: "Amazon CLI/ElasticLoadBalancing API 2012-06-01",
      requestParameters: {
        loadBalancerName: "my-loadbalancer",
        listeners: [
          {
            loadBalancerPort: 80,
            protocol: "http",
            instanceProtocol: "http",
            instancePort: 80
          }
        ],
        availabilityZones: [
          "us-west-2a"
        ]
      },
      responseElements: {
        {
          dnsName: "my-loadbalancer-1234567890.elb.amazonaws.com"
        },
        {
          requestID: "b9960276-b9b2-11e3-8a13-f1ef1EXAMPLE",
          eventID: "6f4ab5bd-2daa-4d00-be14-d92efEXAMPLE"
        }
      }
    }
  ]
}
```

```
{
  eventVersion: "1.01",
  userIdentity: {
    type: "IAMUser",
    principalId: "60505EXAMPLE",
```

Elastic Load Balancing Developer Guide

Elastic Load Balancing Event Entries in CloudTrail Log Files

```
arn: "arn:aws:iam::123456789012:user/Alice",
accountId: "123456789012",
accessKeyId: "AKIAIOSFODNN7EXAMPLE"
},
eventTime: "2014-04-01T16:30:30Z",
eventSource: "elasticloadbalancing.amazonaws.com",
eventName: "DeleteLoadBalancer",
awsRegion: "us-west-2",
sourceIPAddress: "127.0.0.02",
userAgent: "Amazon CLI/ElasticLoadBalancing API 2012-06-01",
requestParameters: {
  loadBalancerName: "my-loadbalancer"
},
responseElements: null,
requestID: "f0f17bb6-b9ba-11e3-9b20-999fdEXAMPLE",
eventID: "4f99f0e8-5cf8-4c30-b6da-3b69fEXAMPLE"
},
. . .
]
}
```

You can also use one of the Amazon partner solutions that integrate with CloudTrail to read and analyze your CloudTrail log files. For more information, see the [AWS partners](#) page.

Control Access to Your Load Balancer

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

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

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

For general information about IAM, see, [What is IAM?](#). For information about creating and managing users and groups, see [IAM Users and Groups](#).

Contents

- [Using an IAM Policy to Grant Permissions \(p. 119\)](#)
- [Example IAM Policies for Elastic Load Balancing \(p. 122\)](#)

Using an IAM Policy to Grant Permissions

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

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

```
    "Resource": "resource-arn",
    "Condition": {
      "condition": {
        "key": "value"
      }
    }
  }
}
```

- **Effect**— The *effect* can be `Allow` or `Deny`. By default, IAM users don't have permission to use resources and API actions, so all requests are denied. An explicit allow overrides the default. An explicit deny overrides any allows.
- **Action**— The *action* is the specific API action for which you are granting or denying permission. To learn about specifying *action*, see [Specifying Actions in an IAM Policy \(p. 120\)](#).
- **Resource**— The resource that's affected by the action. With some Elastic Load Balancing API actions, you can restrict the permissions granted or denied to a specific load balancer. To specify a specific load balancer in this statement, you must use the Amazon Resource Name (ARN) of the load balancer. For more information, see [Specifying Resources in an IAM Policy \(p. 121\)](#). If the API action does not support specifying a specific load balancer, use the `*` wildcard to specify all load balancers.
- **Condition**— You can optionally use conditions to control when your policy is in effect. For more information, see [Specifying Condition Keys in an IAM Policy \(p. 121\)](#).

Specifying Actions in an IAM Policy

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

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

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

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

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

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

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

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

For a list of the API actions for Elastic Load Balancing, see [Actions](#) in the *Elastic Load Balancing API Reference*.

Specifying Resources in an IAM Policy

Resource-level permissions refers to the ability to specify which resources users are allowed to perform actions on. Elastic Load Balancing has partial support for resource-level permissions. This means that for certain API actions, you can control which load balancers users are allowed to use those actions on.

To specify a load balancer in the **Resource** element of the policy statement, you must use its Amazon Resource Name (ARN). The ARN for a load balancer has the following syntax:

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

- *region*— The region for the load balancer. For list of regions supported by Elastic Load Balancing, see [Regions and Endpoints](#).
- *account-id*— Your AWS account ID, with no hyphens (for example, 0123456789012).
- *load-balancer-name*— The name of your load balancer. You can use the * wildcard to specify all of your load balancers.

You control user access to a specific load balancer using a **Resource** statement with the ARN of the load balancer. For example:

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

However, you can't specify a specific load balancer with the following API actions:

- DescribeInstanceHealth
- DescribeLoadBalancerAttributes
- DescribeLoadBalancerPolicyTypes
- DescribeLoadBalancers
- DescribeLoadBalancerPolicies
- DescribeTags

If you specify a resource with an API action that does not support resource-level permissions, the API call fails. Instead, you must use the following resource statement with API actions that don't support resource-level permissions:

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

Specifying Condition Keys in an IAM Policy

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

Note

Key names are case sensitive.

- `aws:CurrentTime`— Use with date/time conditions to restrict access based on request time (see [Date Conditions](#)).

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

After you have decided how you want to control access to your load balancers, open the [IAM Console](#) and follow the directions in [Managing IAM Policies](#) to create an IAM policy for Elastic Load Balancing. For information about testing your IAM policies, see [Testing IAM Policies](#).

Example IAM Policies for Elastic Load Balancing

The following examples show policy statements that you can use to control the permissions that IAM users have to access your load balancer.

Example 1: Allows users to use specific API actions with a specific load balancer

The following policy grants users permission to use all Describe APIs with all load balancers associated with AWS account, and allows users to use the listed API actions only on the specified load balancer:

```
{
  "Version": "2012-10-17",
  "Statement": [ {
    "Effect": "Allow",
    "Action": "elasticloadbalancing:Describe*",
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "elasticloadbalancing:ApplySecurityGroupsToLoadBalancer",
      "elasticloadbalancing:AttachLoadBalancerToSubnets",
      "elasticloadbalancing:ConfigureHealthCheck",
      "elasticloadbalancing:Create*",
      "elasticloadbalancing>Delete*",
      "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
      "elasticloadbalancing:DetachLoadBalancerFromSubnets",
      "elasticloadbalancing:DisableAvailabilityZonesForLoadBalancer",
      "elasticloadbalancing:EnableAvailabilityZonesForLoadBalancer",
      "elasticloadbalancing:ModifyLoadBalancerAttributes",

```



```
        "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
        "elasticloadbalancing:Set*"
    ],
    "Resource": "arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/my-load-balancer"
  }]
}
```

Example 2: Allow users to use specific API actions with two specific load balancers

The following policy grants users permission to use all Describe APIs with all load balancers associated with the AWS account, and allows users to use the listed API actions only on the specified load balancers:

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "elasticloadbalancing:Describe*",
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "elasticloadbalancing:ApplySecurityGroupsToLoadBalancer",
      "elasticloadbalancing:AttachLoadBalancerToSubnets",
      "elasticloadbalancing:ConfigureHealthCheck",
      "elasticloadbalancing:Create*",
      "elasticloadbalancing>Delete*",
      "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
      "elasticloadbalancing:DetachLoadBalancerFromSubnets",
      "elasticloadbalancing:DisableAvailabilityZonesForLoadBalancer",
      "elasticloadbalancing:EnableAvailabilityZonesForLoadBalancer",
      "elasticloadbalancing:ModifyLoadBalancerAttributes",
      "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
      "elasticloadbalancing:Set*"
    ],
    "Resource": [
      "arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/my-load-balancer-1",
      "arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/my-load-balancer-2"
    ]
  }]
}
```

Example 3: Allow users to use specific API actions with a specific load balancer

The following policy grants users permission to use the RegisterInstancesWithLoadBalancer and DeregisterInstancesFromLoadBalancer API actions on the specified load balancer:

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
```

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

Example 4: Allow users to use specific API actions on a specific load balancer using SSL

The following policy grants users permission to use the `RegisterInstancesWithLoadBalancer` and `DeregisterInstancesFromLoadBalancer` API actions on the specified load balancer with an SSL request.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
      "elasticloadbalancing:RegisterInstancesWithLoadBalancer"
    ],
    "Condition": { "Bool": { "aws:SecureTransport": "true" } },
    "Resource": "arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/my-load-balancer"
  }]
}
```

Troubleshoot Your Load Balancer

The following tables list the troubleshooting resources that you'll find useful as you work with Elastic Load Balancing.

API Errors

Error
CertificateNotFound: undefined (p. 126)
OutOfService: A Transient Error Occurred (p. 127)

HTTP Errors

Error
HTTP 400: BAD_REQUEST (p. 127)
HTTP 405: METHOD_NOT_ALLOWED (p. 128)
HTTP 408: Request Timeout (p. 128)
HTTP 502: Bad Gateway (p. 128)
HTTP 503: Service Unavailable (p. 128)
HTTP 504: Gateway Timeout (p. 128)

Response Codes

Response Code
HTTPCode_ELB_4XX (p. 129)
HTTPCode_ELB_5XX (p. 129)
HTTPCode_Backend_2XX (p. 130)
HTTPCode_Backend_3XX (p. 130)

Response Code
HTTPCode_Backend_4XX (p. 130)
HTTPCode_Backend_5XX (p. 130)

Health Check Errors

Error
Connection to the instances has timed out (p. 131)
Health check target page error (p. 132)
Public key authentication is failing (p. 132)
Stopped and started instances failing load balancer health check (p. 132)
Instance is not receiving traffic from the load balancer (p. 133)
Ports on instance are not open (p. 133)
Instances in the Auto Scaling group are failing load balancer health check (p. 133)

Registering Instances

Error
Taking too long to register back-end instances. (p. 134)
Unable to register instance launched from a paid AMI. (p. 134)

Troubleshooting Elastic Load Balancing: API Errors

The following are error messages returned by Elastic Load Balancing API, potential causes, and steps that you can take to narrow down and resolve the issues.

Error Messages

- [CertificateNotFound: undefined](#) (p. 126)
- [OutofService: A Transient Error Occurred](#) (p. 127)

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, contact the [AWS Support Center](#).

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 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 associated back-end instances.

Solution: Retry the API call. If the problem persists, contact the [AWS Support Center](#).

Troubleshooting Elastic Load Balancing: 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, [Hypertext Transfer Protocol-HTTP/1.1](#). 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.

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 the checks, the load balancer sends the request to the back-end EC2 instance. If the method field in the request is malformed, the load balancer responds with an [HTTP 400: BAD_REQUEST](#) (p. 127) error. If the length of the method in the request exceeds 127 characters, the load balancer responds with an [HTTP 405: METHOD_NOT_ALLOWED](#) (p. 128) error.

The back-end EC2 instance processes a valid request by implementing the method in the request and sending a response back to the client. Your back-end instance must be configured to handle both supported and unsupported methods.

The following are error messages returned by your load balancer, potential causes, and steps that you can take to narrow down and resolve the issues.

Error Messages

- [HTTP 400: BAD_REQUEST](#) (p. 127)
- [HTTP 405: METHOD_NOT_ALLOWED](#) (p. 128)
- [HTTP 408: Request Timeout](#) (p. 128)
- [HTTP 502: Bad Gateway](#) (p. 128)
- [HTTP 503: Service Unavailable](#) (p. 128)
- [HTTP 504: Gateway Timeout](#) (p. 128)

HTTP 400: BAD_REQUEST

Description: Indicates that the client sent a bad request.

Cause: The client sent a malformed request that does not meet HTTP specifications.

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 the HTTP specifications.

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. Contact the AWS [Support Center](#) for further 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, contact the AWS [Support Center](#) for further assistance.

Cause 2: Registered instances closing the connection to Elastic Load Balancing.

Solution 2: Enable keep-alive settings on your EC2 instances and set the keep-alive timeout to greater than or equal to the idle timeout settings of your load balancer.

Cause 3: No registered instances.

Solution 3: 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 Load Balancer \(p. 77\)](#).

Cause 4: No healthy instances.

Solution 4: 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` in CloudWatch.

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: The application takes longer to respond than the configured idle timeout.

Solution: Monitor the `HTTPCode_ELB_5XX` and `Latency` CloudWatch 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 back-end capacity or increase the configured idle timeout so that lengthy operations (such as uploading a large file) can complete.

Troubleshooting Elastic Load Balancing: 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 back-end instances. You can use the metrics returned by CloudWatch for your load balancer to troubleshoot issues. For more information see [Monitor Your Load Balancer Using Amazon CloudWatch \(p. 97\)](#).

The following are response code metrics returned by CloudWatch for your load balancer, potential causes, and steps that you can take to narrow down and resolve the issues.

Response Codes

- [HTTPCode_ELB_4XX \(p. 129\)](#)
- [HTTPCode_ELB_5XX \(p. 129\)](#)
- [HTTPCode_Backend_2XX \(p. 130\)](#)
- [HTTPCode_Backend_3XX \(p. 130\)](#)
- [HTTPCode_Backend_4XX \(p. 130\)](#)
- [HTTPCode_Backend_5XX \(p. 130\)](#)

HTTPCode_ELB_4XX

Cause: A malformed or canceled request from the client.

Solutions

- See [HTTP 400: BAD_REQUEST \(p. 127\)](#).
- See [HTTP 405: METHOD_NOT_ALLOWED \(p. 128\)](#).
- See [HTTP 408: Request Timeout \(p. 128\)](#).

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 \(p. 128\)](#).
- See [HTTP 503: Service Unavailable \(p. 128\)](#).
- See [HTTP 504: Gateway Timeout \(p. 128\)](#).

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 (bypass 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.

Troubleshooting Elastic Load Balancing: Health Check Configuration

Your load balancer checks the health of the registered instances using either the default health check configuration provided by Elastic Load Balancing (ELB) or a health check configuration that you specify. The health check configuration contains information such as protocol, ping port, ping path, response timeout, and health check interval. For more information, see [Configure Health Checks \(p. 54\)](#).

There are several ways that you can check the current health state of EC2 instances registered with your load balancer. You can use the AWS Management Console, the `describe-instance-health` AWS CLI command, or the `DescribeInstanceHealth` action.

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 that you can take to narrow down and resolve the issues:

Issues

- [Connection to the instances has timed out \(p. 131\)](#)

- [Connection to your Internet-facing load balancer launched in a VPC has timed out \(p. 132\)](#)
- [Health check target page error \(p. 132\)](#)
- [Public key authentication is failing \(p. 132\)](#)
- [Stopped and started instances failing load balancer health check \(p. 132\)](#)
- [Instance is not receiving traffic from the load balancer \(p. 133\)](#)
- [Ports on instance are not open \(p. 133\)](#)
- [Instances in the Auto Scaling group are failing load balancer health check \(p. 133\)](#)

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 TCP connection:

```
telnet private-IP-address-of-the-instance port
```

Use the following command for an HTTP or an HTTPS connection:

```
curl -I private-IP-address-of-the-instance/health-check-target-page:port
```

If you are using HTTP/HTTPS connection and getting a non-200 response, see [Health check target page error \(p. 132\)](#). 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:

- Verify by checking the monitoring graph for over-utilization of CPU. For information, see [Get Statistics for a Specific EC2 Instance](#).
- Verify by checking 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 about using Auto Scaling, see [What is Auto Scaling?](#).

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 than your configured timeout to respond.

Solution 3: Use a simpler health check target page or adjust the health check interval settings.

Connection to your Internet-facing load balancer launched in a VPC has timed out

Problem: Health check requests are not reaching your instances launched in a VPC because the front-end connection (client to load balancer) has timed out.

Cause: Your Internet-facing load balancer is attached to a private subnet.

Solution: Verify that the VPC has an Internet gateway and that the route table has a route to the Internet Gateway.

Health check target page error

Problem: An HTTP GET request issued to the instance on the given ping port and the ping path (for example, HTTP:80/index.html) is receiving a non-200 response code. Or, some instances are failing the health check and some instances are healthy.

Cause 1: No target page is configured on some or all the instances.

Solution 1: Create a target page (for example, `index.html`) on all the registered instances.

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 on the instances is not configured to receive request from the load balancer.

Solution 3: Check the application on your instance to investigate the cause for non-200 response.

Public key authentication is failing

Problem: A load balancer configured to use the HTTPS or the SSL protocol with back-end authentication enabled is failing public key authentication. Use the `s_client` command to see the list of all the server certificates in the certificate chain. For more information on this command, see https://www.openssl.org/docs/apps/s_client.html.

Cause: The public key on the SSL certificate does not match the public key configured on the load balancer.

Solution: Your SSL certificate might need to be updated. If your SSL certificate is current, try re-installing the certificate on your load balancer. For more information, see [Update the SSL Certificate for Your Load Balancer](#) (p. 48).

Stopped and started instances failing load balancer health check

EC2-Classical Instance

Problem: The instances launched in EC2-Classical have been stopped and then started. The load balancer is not able to connect to the restarted instance.

Cause: Elastic Load Balancing registers your load balancer with your EC2 instance using the associated IP address. When your EC2 instance launched in EC2-Classical is stopped and then restarted, the IP address associated with your instance changes. Your load balancer does not recognize the new IP address which prevents it from connecting with the instance.

Solution: Deregister the instance from the load balancer after you stop the instance and then register the instance with the load balancer after you start the instance. For more information, see [De-register and Register EC2 Instances with Your Load Balancer](#) (p. 67).

EC2-VPC Instance

Problem: The instances launched in a VPC have been stopped and then started. The load balancer is not able to connect to the restarted instance.

Cause: When you stop and then start your VPC instance, it might take some time for the load balancer to recognize that the instance has restarted. During this time, the load balancer is not connected with the restarted instance.

Solution: Re-register the instance with the load balancer after the restart. For more information, see [De-register and Register EC2 Instances with Your Load Balancer \(p. 67\)](#).

Instance is not receiving traffic from the load balancer

Problem: Instance security group is blocking the traffic from ELB security group.
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 instances 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 of your EC2-VPC load balancer does not have egress rules set up to send traffic to the back-end instances.

Solution 2: Edit the security group of your EC2-VPC load balancer to allow traffic to the subnets and the back-end instances.

For information about managing the security group for the instances launched in EC2-Classic, see [Security Groups for Load Balancers in EC2-Classic \(p. 57\)](#).

For information about managing the security group of the instances launched in a VPC, see [Security Groups for Load Balancers in a VPC \(p. 60\)](#).

Ports on instance are not open

Problem: The health check sent by the load balancer is blocked by the back-end instance ports or firewalls. 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 the Auto Scaling group are failing load balancer health check

Problem: Instances in your Auto Scaling group are passing the default Amazon EC2 health check but failing the load balancer health check.

Cause: Amazon EC2 health checks are different from load balancer health checks. Amazon EC2 health checks include status checks on an instance to detect hardware and software issues, 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 load balancer health check because the application running within the instance might have issues that cause the load balancer to consider the instance out of service. This instance might pass the Amazon EC2 health check; it would not be replaced by the Auto Scaling policy because it is considered healthy based on the Amazon EC2 health check.

Solution: Use the load balancer health check for your Auto Scaling group. When you use the load balancer health check, Auto Scaling will determine the health status of your instances by checking the results of both the instance status check and the load balancer health check. For more information, see [Add an Elastic Load Balancing Health Check to your Auto Scaling Group](#) in the *Auto Scaling Developer Guide*.

Troubleshooting Elastic Load Balancing: Registering Instances

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 back-end instances, potential causes, and steps that you can take to narrow down and resolve the issues.

Issues

- [Taking too long to register back-end instances.](#) (p. 134)
- [Unable to register instance launched from a paid AMI.](#) (p. 134)

Taking too long to register back-end instances.

Problem: Registering instances taking longer than expected to be `In Service`.

Cause: Your back-end instances might be failing health check. After the initial instance registration steps are completed (it can take up to approximately 30 seconds), the back-end instances go through the health checks. Your load balancer will not treat your back-end instances as `In Service` until it has successfully met the healthy threshold defined in your health check configuration.

Solution: Try the steps recommended in the preceding section [Connection to the instances has timed out](#) (p. 131).

Unable to register instance launched from a paid AMI.

Problem: Elastic Load Balancing not registering instances launched using a paid AMI.

Cause: Your instances might have been launched using a paid AMI from [Amazon DevPay](#) site.

Solution: Elastic Load Balancing does not support registering instances launched using paid AMI from [Amazon DevPay](#) site. If you want to use paid AMI, depending on your use case, you might find [AWS Marketplace](#) to be a good alternative. If you are already using a paid AMI from AWS Marketplace and are unable to register the instance launched from that paid AMI, contact the AWS [Support Center](#) for further assistance.

For more information about paid AMIs, see [Paid AMIs](#) in the *Amazon Elastic Compute Cloud User Guide*.

Elastic Load Balancing Limits

Your AWS account comes with the following default limits for Elastic Load Balancing:

- Load balancers per region: 20
- Listeners per load balancer: 100

To view the default limits for other services, see [AWS Service Limits](#).

You can request a limit increase for your account using the following procedure.

To request a limit increase

1. Open the [AWS Support Center](#) page, sign in, if necessary, and then click **Open a new case**.
2. Under **Regarding**, select **Service Limit Increase**.
3. Under **Limit Type**, select `Elastic Load Balancers`, fill in all of the necessary fields in the form, and then click the button at the bottom of the page for your preferred method of contact.

Elastic Load Balancing Command Line Interface

The ELB CLI wraps the API actions to provide multi-function commands. The CLI commands are written in Java and include shell scripts for both Windows and Linux/Unix/Mac OSX. The shell scripts are available as a self-contained ZIP file.

Important

The Elastic Load Balancing CLI has been replaced by the AWS Command Line Interface (AWS CLI), a unified tool to manage multiple AWS services. New features released after ELB CLI version 1.0.35.0 (dated 7/24/14) are included only in the AWS CLI. We strongly recommend that you use the AWS CLI. To get started using the AWS CLI, see [Getting Set Up with the AWS Command Line Interface](#).

Process for Installing the ELB CLI

- [Task 1: Download the Command Line Interface](#) (p. 136)
- [Task 2: Set the JAVA_HOME Environment Variable](#) (p. 136)
- [Task 3: Set the AWS_ELB_HOME Environment Variable](#) (p. 138)
- [Task 4: Set the AWS_CREDENTIAL_FILE Environment Variable](#) (p. 139)
- [Task 6: Verify the Installation](#) (p. 140)

Task 1: Download the Command Line Interface

Download the command line interface tools from [Elastic Load Balancing API Tools](#) and unzip the ZIP file.

Task 2: Set the JAVA_HOME Environment Variable

The Elastic Load Balancing command line tool reads an environment variable (JAVA_HOME) on your computer to locate the Java runtime. The command line tool requires Java version 5 or later to run. Either a JRE or JDK installation is acceptable.

To set the JAVA_HOME Environment Variable

1. If you do not have Java 1.5 or later installed, download and install it now. To view and download JREs for a range of platforms, including Linux and Windows, see <http://java.oracle.com/>.
2. Set JAVA_HOME to the full path of the directory that contains a subdirectory named `bin` that in turn contains the Java executable. For example, if your Java executable is in the `/usr/jdk/bin` directory, set JAVA_HOME to `/usr/jdk`. If your Java executable is in `C:\jdk\bin`, set JAVA_HOME to `C:\jdk`.

Note

If you are using Cygwin, you must use Linux paths (for example, `/usr/bin` instead of `C:\usr\bin`) for `AWS_ELB_HOME` and `AWS_CREDENTIAL_FILE`. However, `JAVA_HOME` should have a Windows path. Additionally, the value cannot contain any spaces, even if the value is quoted or the spaces are escaped.

The following Linux example sets `JAVA_HOME` for a Java executable in the `/usr/local/jre/bin` directory.

```
$ export JAVA_HOME=/usr/local/jre
```

The following Windows example uses `set` and `setx` to set `JAVA_HOME` for a Java executable in the `C:\java\jdk1.6.0_6\bin` directory. The `set` command defines `JAVA_HOME` for the current session and `setx` makes the change permanent.

```
C:\> set JAVA_HOME=C:\java\jdk1.6.0_6
C:\> setx JAVA_HOME C:\java\jdk1.6.0_6
```

Note

- The `setx` command does not use the `=` sign.
- Don't include the `bin` directory in `JAVA_HOME`; that's a common mistake some users make. The command line tool won't work if you do.
- The value for `JAVA_HOME` cannot contain any spaces, even if the value is quoted or the spaces are escaped. If the value contains a space character, the command line tool will return an error when you add `JAVA_HOME` to your path in the next step.

3. Add your Java directory to your path before other versions of Java.

On Linux, you can update your `PATH` as follows:

```
$ export PATH=$JAVA_HOME/bin:$PATH
```

On Windows the syntax is slightly different:

```
C:\> set PATH=%JAVA_HOME%\bin;%PATH%
C:\> setx PATH %JAVA_HOME%\bin;%PATH%
```

4. On Linux, verify your `JAVA_HOME` setting with the command `$JAVA_HOME/bin/java -version`.

```
$ $JAVA_HOME/bin/java -version
java version "1.5.0_09"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_09-b03)
Java HotSpot(TM) Client VM (build 1.5.0_09-b03, mixed mode, sharing)
```

The syntax is different on Windows, but the output is similar.

```
C:\> %JAVA_HOME%\bin\java -version
java version "1.5.0_09"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_09-b03)
Java HotSpot(TM) Client VM (build 1.5.0_09-b03, mixed mode, sharing)
```

Task 3: Set the AWS_ELB_HOME Environment Variable

The command line tool depends on an environment variable (AWS_ELB_HOME) to locate supporting libraries. You'll need to set this environment variable before you can use the tool.

To set the AWS_ELB_HOME Environment Variable

1. Set AWS_ELB_HOME to the path of the directory into which you unzipped the command line tool. This directory is named ElasticLoadBalancing-w.x.y.z (w, x, y, and z are version/release numbers) and contains sub-directories named bin and lib.

The following Linux example sets AWS_ELB_HOME for a directory named ElasticLoadBalancing-1.0.12.0 in the /usr/local directory.

```
$ export AWS_ELB_HOME=/usr/local/ElasticLoadBalancing-1.0.12.0
```

The following Windows example sets AWS_ELB_HOME for a directory named ElasticLoadBalancing-1.0.12.0 in the C:\CLIs directory.

```
C:\> set AWS_ELB_HOME=C:\CLIs\ElasticLoadBalancing-1.0.12.0
C:\> setx AWS_ELB_HOME C:\CLIs\ElasticLoadBalancing-1.0.12.0
```

2. Add the tool's bin directory to your system PATH. The rest of this guide assumes that you've done this.

On Linux, you can update your PATH as follows:

```
$ export PATH=$PATH:$AWS_ELB_HOME/bin
```

On Windows the syntax is slightly different:

```
C:\> set PATH=%PATH%;%AWS_ELB_HOME%\bin
C:\> setx PATH %PATH%;%AWS_ELB_HOME%\bin
```


Task 4: Set the AWS_CREDENTIAL_FILE Environment Variable

After you sign up for AWS, you must create access keys for the account. Your access keys are your `AccessKeyId` and `SecretAccessKey`. You must provide your access keys to the command line tools so that they know that the commands that you issue come from your account. The command line tool reads your access keys from a credential file that you create on your local system.

You can either specify your access keys with the `--aws-credential-file` parameter every time you issue a command or you can create an environment variable that points to the credential file on your local system. If the environment variable is properly configured, you can omit the `--aws-credential-file` parameter when you issue a command. The following procedure describes how to create a credential file and a corresponding `AWS_CREDENTIAL_FILE` environment variable.

To create a credential file on your local system

1. Retrieve your access keys.

Although you can retrieve the access key ID from the **Security Credentials** page, you cannot retrieve the secret access key. You'll need to create new access keys if the secret access key was lost or forgotten. For more information, see [How Do I Get Security Credentials?](#) in the *Amazon Web Services General Reference*.

2. Write down your secret access key and access key ID, or save them.
3. Add your access key ID and secret access key to the file named `credential-file-path.template`:
 - a. Open the file `credential-file-path.template` included in your command line interface (CLI) archive.
 - b. Copy and paste your access key ID and secret access key into the file.
 - c. Rename the file and save it to a convenient location on your computer.
 - d. If you are using Linux, set the file permissions as follows:

```
$ chmod 600 credential-file-name
```

4. Set the `AWS_CREDENTIAL_FILE` environment variable to the fully qualified path of the file you just created.

The following Linux example sets `AWS_CREDENTIAL_FILE` for `myCredentialFile` in the `/usr/local` directory.

```
$ export AWS_CREDENTIAL_FILE=/usr/local/myCredentialFile
```

The following Windows example sets `AWS_CREDENTIAL_FILE` for `myCredentialFile.txt` in the `C:\aws` directory.

```
C:\> set AWS_CREDENTIAL_FILE=C:\aws\myCredentialFile.txt  
C:\> setx AWS_CREDENTIAL_FILE C:\aws\myCredentialFile.txt
```

Task 6: Verify the Installation

To verify the installation and configuration of Elastic Load Balancing CLI

1. On your Linux or Windows workstation, open a new command prompt.
2. Type the command `elb-cmd`.
3. You should see output similar to the following:

Command Name	Description
-----	-----
elb-apply-security-groups-to-lb Balancer.	Apply VPC security groups to Load Balancer.
elb-attach-lb-to-subnets Subnets in VPC.	Attach existing LoadBalancer to Subnets in VPC.
elb-configure-healthcheck with a LoadBalancer.	Configure the parameters f...tered with a LoadBalancer.
elb-create-app-cookie-stickiness-policy	Create an application-generated stickiness policy.
elb-create-lb	Create a new LoadBalancer.
elb-create-lb-cookie-stickiness-policy	Create an LB-generated stickiness policy.
elb-create-lb-listeners	Create a new LoadBalancer listener.
elb-create-lb-policy LoadBalancerPolicyType.	Create a LoadBalancer poli...ed
elb-delete-lb	Deletes an existing LoadBalancer.
elb-delete-lb-listeners LoadBalancer.	Deletes a listener on an existing LoadBalancer.
elb-delete-lb-policy	Delete a LoadBalancer policy.
elb-deregister-instances-from-lb Balancer.	Deregisters instances from a Load Balancer.
elb-describe-instance-health	Describes the state of instances.
elb-describe-lb-policies Policies.	Describes the details of LoadBalancer Policies.
elb-describe-lb-policy-types PolicyTypes.	Describes the details of LoadBalancer PolicyTypes.
elb-describe-lbs of LoadBalancers.	Describes the state and properties of LoadBalancers.

Elastic Load Balancing Developer Guide

Task 6: Verify the Installation

<code>elb-detach-lb-from-subnets</code>	Detach existing LoadBalancer from Subnets in VPC.
<code>elb-disable-zones-for-lb</code>	Remove availability zones from an LoadBalancer.
<code>elb-enable-zones-for-lb</code>	Add availability zones to existing LoadBalancer.
<code>elb-register-instances-with-lb</code>	Registers instances to a LoadBalancer.
<code>elb-set-lb-listener-ssl-cert</code>	Set the SSL Certificate fo...ecified LoadBalancer port.
<code>elb-set-lb-policies-for-backend-server</code>	Set LoadBalancer policies for backend servers.
<code>elb-set-lb-policies-of-listener</code>	Set LoadBalancer policies for a port.
<code>version</code>	Prints the version of the CLI tool and the API.
For help on a specific command, type '<commandname> --help'	

Document History

The following table describes the releases and important changes for Elastic Load Balancing.

Feature	WSDL	Description	Release Date
Additional fields for access log entries		Added the <code>user_agent</code> , <code>ssl_cipher</code> , and <code>ssl_protocol</code> fields. For more information, see Access Log Files (p. 107) .	May 18, 2015
Support for registering linked EC2-Classic instances	2014-08-11 WSDL	Added support for registering linked EC2-Classic instances with your load balancer.	January 19, 2015
Support for tagging your load balancer	2014-08-11 WSDL	You can use tags to organize and manage your load balancers. 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 released 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 Command Line Interface (AWS CLI) instead. For more information, see Access Elastic Load Balancing Using the AWS CLI (p. 7) .	August 11, 2014
idle connection timeout	2014-07-24 WSDL	You can configure the idle connection timeout for your load balancer.	July 24, 2014
Support for granting IAM users and groups access to specific load balancers or API actions	2014-03-20 WSDL	You can create an IAM policy to grant IAM users and groups access to specific load balancers or API actions. For more information, see Control Access to Your Load Balancer (p. 119) .	May 12, 2014

Feature	WSDL	Description	Release Date
Support for AWS CloudTrail	2014-03-20 WSDL	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. For more information, see Log Elastic Load Balancing API Calls Using AWS CloudTrail (p. 116).	April 04, 2014
connection draining	2014-03-20 WSDL	Added information about connection draining. With this support you can enable your load balancer to stop sending new requests to the registered 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 Load Balancer (p. 80).	March 20, 2014
access logs	2014-03-06 WSDL	You can enable your load balancer to capture detailed information about the requests sent to your load balancer and store it in an S3 bucket. For more information, see Monitor Your Load Balancer Using Elastic Load Balancing Access Logs (p. 107).	March 06, 2014
Support for TLSv1.1-1.2	2013-11-06 WSDL	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 Elastic Load Balancing (p. 26). For information about updating your current SSL negotiation configuration, see Update the SSL Negotiation Configuration of Your Load Balancer (p. 50).	February 19, 2014
cross-zone load balancing	2013-11-06 WSDL	Added information about enabling cross-zone load balancing for your load balancer. For information on cross-zone load balancing, see Request Routing (p. 4). For procedure on enabling or disabling the cross-zone load balancing, see Configure Cross-Zone Load Balancing for Your Load Balancer (p. 77)	November 06, 2013

Feature	WSDL	Description	Release Date
Additional CloudWatch Metrics	2012-06-01 WSDL	Added information about the additional Cloudwatch metrics reported by Elastic Load Balancing. For more information, see Monitor Your Load Balancer Using Amazon CloudWatch (p. 97) .	October 28, 2013
Support for Proxy Protocol	2012-06-01 WSDL	Added information about Proxy Protocol support for load balancers configured for TCP/SSL connections. For more information, see Proxy Protocol Header (p. 84) .	July 30, 2013
Support for DNS failover	2012-06-01 WSDL	Added information about configuring Route 53 DNS failover for load balancers. For more information, see Configure DNS Failover for Your Load Balancer (p. 95) .	June 03, 2013
Console support for viewing CloudWatch metrics and creating alarms	2012-06-01 WSDL	Added information about viewing CloudWatch metrics and creating alarms for a specified load balancer using the console. For more information, see Monitor Your Load Balancer Using Amazon CloudWatch (p. 97) .	March 28, 2013
Support for registering EC2 instances in a default VPC	2012-06-01 WSDL	Added support for EC2 instances launched in a default VPC.	March 11, 2013
internal load balancers	2012-06-01 WSDL	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 that 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 Load Balancer (p. 16) .	June 10, 2012
Console support for managing listeners, cipher settings, and SSL certificates	2011-11-15 WSDL	For information, see Configure an HTTPS Listener for Your Load Balancer (p. 45) and Update the SSL Certificate for Your Load Balancer (p. 48) .	May 18, 2012
Support for Elastic Load Balancing in Amazon VPC	2011-11-15 WSDL	Added support for creating a load balancer in a virtual private cloud (VPC).	November 21, 2011
Amazon CloudWatch	2011-08-15 WSDL	You can monitor your load balancer using CloudWatch. For more information, see Monitor Your Load Balancer Using Amazon CloudWatch (p. 97) .	October 17, 2011

Feature	WSDL	Description	Release Date
Additional security features	2011-08-15 WSDL	You can configure SSL ciphers, back-end SSL, and back-end server authentication. For more information, see Create an HTTPS Load Balancer (p. 31) .	August 30, 2011
zone apex domain name	2011-04-05 WSDL	For more information, see Configure a Custom Domain Name for Your Load Balancer (p. 94) .	May 24, 2011
instance lock-down	2011-04-05 WSDL	You can use the security group provided by Elastic Load Balancing to lock down your back-end instance. For more information, see Security Groups for Load Balancers in EC2-Classic (p. 57) .	May 24, 2011
Support for IPv6	2011-04-05 WSDL	You can use Internet Protocol version 6 (IPv6) with your load balancer in EC2-Classic.	May 24, 2011
Support for X-Forwarded-Proto and X-Forwarded-Port headers	2010-07-01 WSDL	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 X-Forwarded Headers (p. 74) .	October 27, 2010
Support for HTTPS	2010-07-01 WSDL	With this release, you can leverage the SSL/TLS protocol for encrypting traffic and offload SSL processing 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)	2010-07-01 WSDL	For more information, see Control Access to Your Load Balancer (p. 119) .	September 02, 2010
sticky sessions	2009-11-25 WSDL	For more information, see Configure Sticky Sessions for Your Load Balancer (p. 87) .	April 07, 2010
AWS SDK for Java	2009-11-25 WSDL	Added support for the SDK for Java.	March 22, 2010
AWS SDK for .NET	2009-05-15 WSDL	Added support for the AWS SDK for .NET.	November 11, 2009

Feature	WSDL	Description	Release Date
New service	2009-05-15 WSDL	Initial public beta release of Elastic Load Balancing.	May 18, 2009