Elastic Load Balancing

Developer Guide

API Version 2011-11-15

Elastic Load Balancing: Developer Guide

Elastic Load Balancing Developer Guide

Welcome to the *Elastic Load Balancing Developer Guide*. Amazon Web Services (AWS) provides Elastic Load Balancing to automatically distribute your incoming application traffic across multiple Amazon Elastic Compute Cloud (Amazon EC2) instances. It detects unhealthy instances and reroutes traffic to healthy instances until the unhealthy instances have been restored. Elastic Load Balancing automatically scales its request handling capacity in response to incoming traffic.

The *Elastic Load Balancing Developer Guide* gives you basic information about Elastic Load Balancing so you can make an informed decision about choosing to use it. This guide also helps you decide how to use Elastic Load Balancing in specific user scenarios. You can choose which one is right for you.

Before You Begin

How Do I...	Relevant Topics
Learn about the business case for Elastic Load Balancing	Elastic Load Balancing product information
Learn about how Elastic Load Balancing works and decide whether Elastic Load Balancing is the right choice for my use case	What is Elastic Load Balancing? (p. 3)

Where Do I Start?

We recommend that you read *What is Elastic load Balancing* to get answers to your question on basics. Then, to familiarize yourself with Elastic Load Balancing, you should walk through *Get Started with Elastic Load Balancing*. This tutorial will show you how to create a basic load balancer.

If you are a *new* AWS customer, you are eligible to use the free usage tier for twelve months following your AWS sign-up date. The free tier includes 750 hours per month of Amazon EC2 Micro Instance usage, and 750 hours per month of Elastic Load Balancing, plus 15 GB of data processing.

Where Do I...	Relevant Topics
Learn if I am eligible to use the free usage tier for twelve months following my AWS sign-up date	AWS Free Usage Tier
Learn how to create a basic load balancer and register my Amazon EC2 instances with the load balancer	Get Started With Elastic Load Balancing (p. 11)

If you've used load balancing in a physical hardware environment, you'll know how to evaluate the behavior of the load balancer in that context. If you are planning to use load balancing in a cloud environment, you need to be aware of some of the Elastic Load Balancing features that might affect your load balancing scenario.

Elastic Load Balancing supports the load balancing of applications using HTTP, HTTPS (secure HTTP), TCP, and SSL (secure TCP) listener protocols and allows you to choose the protocols for both the front-end (client to load balancer) and the back-end (load balancer to back-end instance) connections.

Elastic Load Balancing provides several different interfaces you can use to manage your load balancers. You can create, access, and manage your load balancers using the AWS Management Console, the command line interface (CLI), or the Query API. You will have to install command line interface and the Query API before you can use them.

Where Do I ...	Relevant Topics
Learn about the best practices you can use to evaluate and test Elastic Load Balancing for your use case	Best practices in Evaluating Elastic load Balancing
Learn about the different listener protocols supported by Elastic Load Balancing	Choosing Listeners for Your Load Balancer (p. 34)
Learn how to install the interfaces needed for accessing the Elastic Load Balancing	Get Set Up with Elastic Load Balancing Interfaces (p. 20)

How Do I Use Elastic Load Balancing?

Elastic Load Balancing provides several features that help you load balance your applications effectively. You can create and use your load balancer either for Amazon EC2 or within Amazon Virtual Private Cloud (Amazon VPC), depending on where you've launched your EC2 instances.

Where Do I...	Relevant Topics
Learn more about how to use the various features supported by Elastic Load Balancing	Using Elastic Load Balancing (p. 39)
Learn more about scenarios specific to my instances launched in Amazon EC2	Deploying Elastic Load Balancing in Amazon EC2 (p. 41)
Learn more about scenarios specific to my instances launched in Amazon VPC	Deploying Elastic Load Balancing in Amazon VPC (p. 67)

What Is Elastic Load Balancing?

If you have managed an enterprise network, you know how valuable load balancers can be. With a load balancer, you can add and remove servers as your needs change without disrupting the overall flow of information. If one server fails, the load balancer will stop sending traffic to it. The load balancer offers clients a single point of contact, and it can also serve as the first line of defense against attacks on your network. You can offload the work of encryption and decryption to the load balancer, so your servers can focus on their main task.

Load balancers are just as useful in cloud computing. Amazon Web Services (AWS) provides Elastic Load Balancing to manage traffic on a fleet of Amazon Elastic Compute Cloud (Amazon EC2) instances. Elastic Load Balancing has all the advantages of an on-premises load balancer, plus it offers the following benefits:

- Distribution of requests to Amazon EC2 instances (servers) in multiple Availability Zones in a way that minimizes the risk of overloading one single instance. And if an entire Availability Zone goes offline, Elastic Load Balancing routes traffic to instances in other Availability Zone.
- Continuous monitoring of the health of Amazon EC2 instances registered with the load balancer and send the request only to the *healthy* instances. If an instance becomes *unhealthy*, Elastic Load Balancing stops sending traffic to that instance and spreads the load across remaining healthy instances.
- Support for an end-to-end traffic encrytion on those networks that uses secure (HTTPS/SSL) connections.
- Takes over the encryption and decryption work from the Amazon EC2 instances and manages it centrally on the load balancer.
- Supports the sticky session feature, which is the ability to "stick" user sessions to specific Amazon EC2 instances.
- Association of the load balancer with your domain name. Because the load balancer is the only computer that is exposed to the Internet, you don’t have to create and manage public domain names for the instances that the load balancer manages. 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.
- When used in a Virtual Private Cloud (VPC), supports creation and management of security groups associated with your Elastic Load Balancing to provide additional networking and security options.
- Supports use of both the Internet Protocol version 4 (IPv4) and Internet Protocol version 6 (IPv6).

As with all 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 for Elastic Load Balancing, go to Elastic Load Balancing Pricing.

If you are a *new* AWS customer, you are eligible to use the free usage tier for twelve months following your AWS sign-up date. The free tier includes 750 hours per month of Amazon EC2 Micro Instance usage, and 750 hours per month of Elastic Load Balancing, plus 15 GB of data processing. For information about the free usage tier, go to AWS Free Usage Tier.

How Elastic Load Balancing Works

Elastic Load Balancing consists of two components: the load balancers and the controller service. The load balancers monitor the traffic and handle requests that come in through the Internet. The controller service monitors the load balancers, adding and removing capacity as needed and verifying that the load balancers are functioning properly.

You have to create your load balancer before you can start using it. Each load balancer you create must have a unique Domain Name System (DNS) name. For example, if you create a load balancer named myLB in the US East Region, your load balancer might have a DNS name such as myLB-1234567890.us-east-1.elb.amazonaws.com.

You have to register the instances that you want to load balance with the load balancer. Elastic Load Balancing registers your load balancer with your instances using the IP addresses. When the instance is stopped and then started, the IP address associated with your instance changes. This prevents the load balancer from routing traffic to your restarted instance. Elastic Load Balancing gives you the option to de-register your instance from the load balancer after you've stopped your instance, and then register the load balancer with your instance after you've restarted.

Your load balancer monitors and routes the incoming traffic to the registered instances. Your load balancer also monitors the health of the instances and ensures that the traffic goes to healthy instances. When the load balancer detects an unhealthy instance, it stops routing the traffic to that instance and resumes the routing when the instance has been restored to a healthy state. Elastic Load Balancing performs health checks on your instances using the configuration you provide regardless of whether the instance is in a healthy or unhealthy state.

Amazon EC2 provides the ability to launch your instances in multiple Availability Zones. You can configure your load balancer to load balance incoming application traffic across multiple instances in a single Availability Zone or across multiple instances in several Availability Zones in the same Region. For example, if you choose to load balance multiple instances across two Availability Zones and all the instances in the first Availability Zone become unhealthy, the load balancer will route traffic to the healthy instances in the other Availability Zone. When you use multiple Availability Zones, it is important to keep approximately the same capacity in each Availability Zone registered with the load balancer.

Using Auto Scaling with Elastic Load Balancing makes it easy to increase or decrease your back-end capacity to meet varying traffic levels. For example, you could set a condition declaring that when the number of healthy instances behind a load balancer goes down to two, two or more instances are launched. Or, you could set a condition to monitor the latency of the load balancer, and when the latency exceeds certain time period, such as three seconds, capacity is increased. You can also use AWS Management Console to register or deregister instances used by the load balancer as the capacity requirements of your application change over time.

Overview of Elastic Load Balancing

Elastic Load Balancing Concepts

This topic introduces you to Elastic Load Balancing basics you need to understand before you create your load balancer.

Load Balancer

A *load balancer* is represented by a DNS name and a set of ports. The load balancer is the destination to which all requests intended for your application should be directed. Each load balancer can distribute requests to multiple EC2 instances. Load Balancers can span multiple Availability Zones within an EC2 *region*, but they cannot span multiple regions.

To create or work with a load balancer in a specific region, use the corresponding regional service endpoint. For information about regions and endpoints supported by Elastic Load Balancing, go to Regions and Endpoints.

If no endpoint is explicitly specified, the US East (Northern Virginia) Region endpoint is used by default.

Elastic Load Balancing automatically generates a DNS name for each load balancer. You can map any other domain name (such as www.example.com) to the automatically generated DNS name using CNAME. Or you can use an Amazon Route 53 alias for the load balancer's DNS name. Amazon Route 53 provides secure and reliable routing to the infrastructure that uses AWS products, such as Amazon EC2, Amazon Simple Storage Service (Amazon S3), or Elastic Load Balancing. For more information on using Amazon Route 53 for your load balancer, see Using Domain Names with Elastic Load Balancing (p. 90). For information about CNAME records, see the CNAME Record Wikipedia article.

Registering EC2 Instances

Elastic Load Balancing registers your load balancer with your EC2 instances using the IP addresses that are associated with your instances.

When the instance is stopped and then restarted, the IP address associated with your instance changes. Your load balancer cannot recognize the new IP address, which prevents it from routing traffic to your instances. You can de-register your Amazon EC2 instances from your load balancer after you stop your instance, and then register the load balancer with your instance after you've restarted. For more information on deregistering and registering your EC2 instances, see De-Registering and Registering Amazon EC2 Instances (p. 85).

Availability Zones and Regions

A load balancer can distribute traffic to instances across all Availability Zones within a region. Elastic Load Balancing does not distribute traffic across regions.

For critical applications, we recommend that you distribute incoming traffic across multiple Availability Zones by registering multiple Availability Zones with your load balancer and registering your EC2 instances in each registered Availability Zone.

Incoming traffic is load balanced equally across all Availability Zones enabled for your load balancer, so it is important to have approximately *equivalent* numbers of instances in each zone. For example, if you have ten instances in Availability Zone us-east-1a and two instances in us-east-1b, the traffic will still be equally distributed between the two Availability Zones. As a result, the two instances in us-east-1b will have to serve the same amount of traffic as the ten instances in us-east-1a. As a best practice, we recommend you keep an equivalent or nearly equivalent number of instances in each of your Availability Zones. So in the example, rather than having ten instances in us-east-1a and two in us-east-1b, you could distribute your instances so that you have six instances in each Availability Zone.

For more information, see Expanding a Load Balanced Application to an Additional Availability Zone (p. 59).

Sticky Sessions

By default, a load balancer routes each request independently to the application 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 application instance. This ensures that all requests coming from the user during the session will be sent to the same application instance.

The key to managing the *sticky session* is determining how long should your load balancer consistently route the user's request to the same application instance. If your application has its own session cookie, then you can set Elastic Load Balancing to create the session cookie to follow the duration specified by the application's session cookie. If your application does not have its own session cookie, then you can set Elastic Load Balancing to create a session cookie by specifying your own stickiness duration. You can associate stickiness duration for only HTTP/HTTPS load balancer listeners.

- For more information about creating cookies that allow duration-based session stickiness, see Enabling Duration-Based Session Stickiness (p. 95).
- For more information about creating cookies that allow application-specific session stickiness, see Enabling Application-Controlled Session Stickiness (p. 97).

An application instance must always receive and send two cookies: A cookie that defines the stickiness duration and a special Elastic Load Balancing cookie named AWSELB, that has the mapping to the application instance.

HTTPS Support

HTTPS Support is a feature that allows you to use the SSL/TLS protocol for encrypted connections (also known as *SSL offload*). This feature enables traffic encryption between your load balancer and clients that initiate HTTPS sessions with your load balancer.

To use HTTPS Support you simply upload your certificate and key and then create a load balancer (or create or update a listener for an existing load balancer) that uses the HTTPS (Secure HTTP) or SSL (Secure TCT) protocol. To upload your certificate and key, you can use the AWS Management Console when you create your load balancer. Alternatively, you can upload the certificate and key using the tools provided by AWS Identity and Access Management (IAM). For more information on uploading SSL certificates, see Managing Server Certificates in the AWS Identity and Access Management documentation.

For more information about the advantages of using HTTPS Support, see Advantages of Using HTTPS/SSL with Elastic Load Balancing (p. 35). For information about creating a load balancer that uses HTTPS, see Creating a Load Balancer With SSL Cipher Settings and Back-end Server Authentication (p. 42).

X-Forwarded-For Support

The `X-Forwarded-For` request header helps you identify the IP address of a client. 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 along to your server.

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

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

The following example is an `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 example is an `X-Forwarded-For` request header for a client with an IPv6 address of `2001:DB8::/32:21f:5bff:febf:ce22`.

```
X-Forwarded-For: 2001:DB8::/32:21f:5bff:febf:ce22
```

If you have back-end application instances in multiple Availability Zones, the `X-Forwarded-For` request header can contain one or more load balancer IP addresses. Because Elastic Load Balancing uses a different load balancer for each Availability Zone, a client request can be passed from one load balancer to another before reaching a back-end application instance. For example, if you have back-end instances in Availability Zones US-east-1a and US-east-1b, a client request might be handled initially by the load balancer in US-east-1a. If Elastic Load Balancing determines that this request should be routed to US-east-1b, the load balancer in US-east-1a routes the request to the load balancer in US-east-1b. This rerouting may occur if there are no healthy instances in US-east-1a or if sticky sessions are used and the back-end instance is in US-east-1b. Each of the load balancers adds its IP address to the `X-Forwarded-For` request header.

If more than one load balancer is involved in a client request, the `X-Forwarded-For` request header takes the following form:

```
X-Forwarded-For: clientIPAddress, previousLoadBalancerIPAddress
```

The following example is an `X-Forwarded-For` request header that arrived at a back-end application instance in the US-east-1b Availability Zone. The client (`203.0.113.7`) made a request that arrived first at a load balancer in US-east-1a (`10.12.33.44`). Subsequently, the load balancer for US-east-1a routed the request to the load balancer in US-east-1b (`10.73.23.88`).

```
X-Forwarded-For: 203.0.113.7, 10.12.33.44, 10.73.23.88
```

X-Forwarded-Proto Support

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:

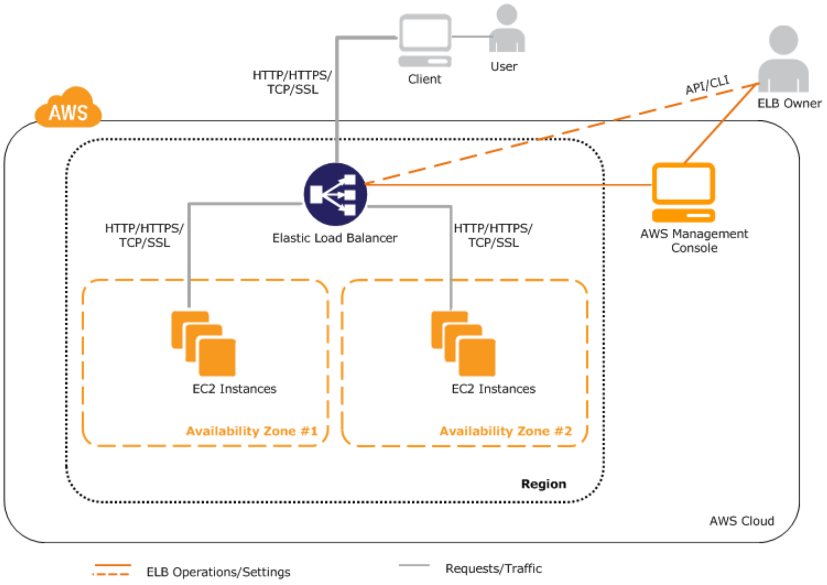
```
X-Forwarded-Proto: originatingProtocol
```

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
```

Architectural Overview

The following diagram shows how the various components of the Elastic Load Balancing work together. The remainder of this section provides a step-by-step view of the flow of events that takes place when a client requests a URL served by your applications.



This example assumes that you have created a load balancer, created a custom domain name and associated your load balancer with the domain name using a CNAME entry in DNS, and have registered your instances with it.

1. The client sends a URL request to DNS servers to access your application. The DNS server responds with a DNS name. For example, myLB-1234567890.us-east-1.elb.amazonaws.com.
2. The client looks for the resolution of the DNS name sent by the DNS server. The DNS entry is controlled by Amazon because your application instances are under the amazonaws.com domain. The Amazon DNS servers return one or more IP addresses.
3. The client then opens a connection to the machine at the provided IP address. The instance at this address is the load balancer you created.
4. The load balancer checks the health states of all the registered EC2 application instances within the selected Availability Zones and will begin routing trafic to instances that have met the healthy threshold defined in the health check configuration.
5. The load balancer routes the client request to the healthy EC2 application instance identified in the previous step. At this point, the client is communicating with one of your EC2 instances through your load balancer. The load balancer listeners can be configured to use either HTTP, HTTPS, TCP, or SSL protocols for both front-end connection (client to load balancer) and back-end connection (load balancer to back-end instance).

Note

Amazon Route 53 is AWS's highly available and cost-effective DNS service. Using Amazon 53's Alias records will provide performance improvements as the clients will then need to only make a single request to resolve the domain name. Also, queries to Alias records are free of charge.

Available Interfaces

You can access and work with your load balancer using one of the following interfaces:

- **AWS Management Console**—A simple web-browser interface that you can use to create and manage your load balancers without using additional software or tools. On the AWS site, you can open the console by clicking Sign in to the AWS Console.
- **Command Line Interfaces (CLI)**—A Java-based command-line client that wraps the SOAP API.
- **Programmatic Interface**— SDKs provided by AWS, third-party libraries, and Elastic Load Balancing Query API.

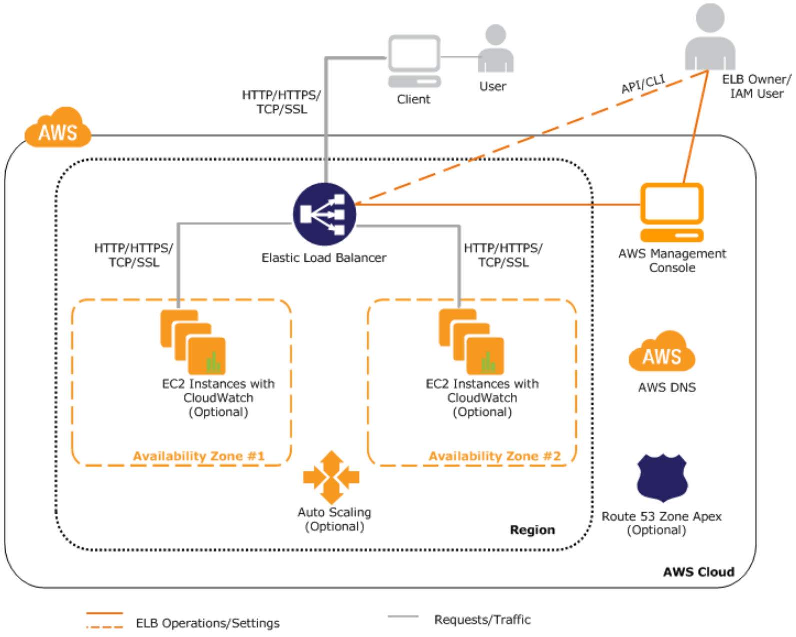
For information on installing and using the command line interfaces, Query APIs, and SDKs provided by AWS, see Get Set Up with Elastic Load Balancing Interfaces (p. 20).

Integration With AWS Services

Elastic Load Balancing integrates with the following AWS services to provide solutions to help your load balancer improve availability and scalability of your applications.

Amazon Web Services	Solutions
Amazon EC2	Runs your back-end application instances.
Auto Scaling	Creates capacity groups of instances that can grow or shrink on demand. For more information, go to the Auto Scaling Developer Guide.
Amazon CloudWatch	Collects the data provided by your load balancer and presents it as readable, near real-time metrics. These metrics can be used to monitor the health state of your instances. You can create an Amazon CloudWatch alarm to send notification to an Auto Scaling policy if an individual metric goes outside of what you consider an acceptable range. For more information on using Amazon CloudWatch with your load balancer, see Monitoring Your Load Balancer Using CloudWatch (p. 99). For information on Amazon CloudWatch go to the Amazon CloudWatch Developer Guide.
Amazon Route 53	Provides secure and reliable routing to your application instances. Route 53 automatically routes queries to the nearest DNS server in a global network of DNS servers, resulting in low latency. You can use Route 53 to translate friendly domain names like www.example.com into IP addresses like 192.0.2.1. For information on using Amazon Route 53 to create a custom domain name for your load balancer see Using Domain Names with Elastic Load Balancing (p. 90). For information on Amazon Route 53, go to the Amazon Route 53 Developer Guide.
AWS Identity and Access Management (IAM)	Manages users and user permissions in AWS. IAM provides central control over users and security credentials. Use IAM to create create multiple users who can use AWS products, each with individual security credentials, all controlled by a single AWS account. For information on specifying user permissions for Elastic Load Balancing resources, see Controlling User Access to Your AWS Account (p. 105). For information on AWS Identity and Access Management, go to Using IAM.

The following diagram shows how the various services in AWS integrate with Elastic Load Balancing.



Where Do I Go From Here

- You might want to test drive Elastic Load Balancing by creating a basic load balancer and registering your EC2 instances with the newly created load balancer. Get Started with Elastic Load Balancing (p. 11) provides information on creating a basic load balancer using the AWS Management Console.
- You might want to explore some common user scenarios for Elastic Load Balancing. Before you do this, you must install the tools and interfaces that you plan to use to access your load balancer. For information on installing the command line interfaces and using the Query API, go to Get Set Up with Elastic Load Balancing Interfaces.
- For detailed instructions on using Elastic Load Balancing in Amazon EC2, go to How Do I Use Elastic Load Balancing in Amazon EC2
- For detailed instructions on using Elastic Load Balancing in Amazon Virtual Private Cloud(VPC), go to How Do I Use Elastic Load Balancing in Amazon VPC.

Get Started with Elastic Load Balancing

Topics

After you read What Is Elastic Load Balancing? (p. 3), and you decide you want load balancing for your Amazon Elastic Compute Cloud instances, it's time to get started with basic load balancing tasks. Your first task will be to create a load balancer.

The following task list gives you a general overview of the steps you'll need to follow to create a basic load balancer. After this task list you'll step through detailed procedures for each part of the creation process.

Creating a Basic Load Balancer

1	Configure the listeners for your load balancer by specifying the ports and protocols to use for front-end connection (client to load balancer) and back-end connection (load balancer to back-end instance).
2	Configure a health check for your Amazon EC2 back-end instances.
3	Register your Amazon EC2 instances with the load balancer.
4	Review settings.
5	Create and test your load balancer.
6	[Optional] Delete your load balancer.

The following step-by-step instructions will help you create a basic load balancer using the AWS Management Console, a point-and-click web-based interface. Before you get started with the console, be sure you've done the following:

- Sign up for Amazon Web Services (AWS). If you haven't signed up for AWS yet, go to http://aws.amazon.com and click the **Sign Up Now** button.

  Note

  If you are a *new* AWS customer, you are eligible to use the free usage tier for twelve months following your AWS sign-up date. The free tier includes 750 hours per month of Amazon EC2 Micro Instance usage, and 750 hours per month Elastic Load Balancing plus 15GB of data processing. For more information on what is available on the free tier, go to AWS Free Usage Tier.

- Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
- Launch Amazon EC2 instances with HTTP access on port 80. You'll be registering these instances with your load balancer. For more information about launching Amazon EC2 instances, see Launching and Using Instances in the *Amazon Cloud Compute User Guide*.

Important

The load balancer you're about to create will be live (and not running in a sandbox). If you are not signed up for free usage tier, you will incur the standard Elastic Load Balancing usage fees for the load balancer until you terminate it. The total charges will be minimal (typically less than a dollar), if you complete the Getting Started in one sitting and delete your load balancer when you are finished. For more information about Elastic Load Balancing usage rates, go to the Elastic Load Balancing product page.

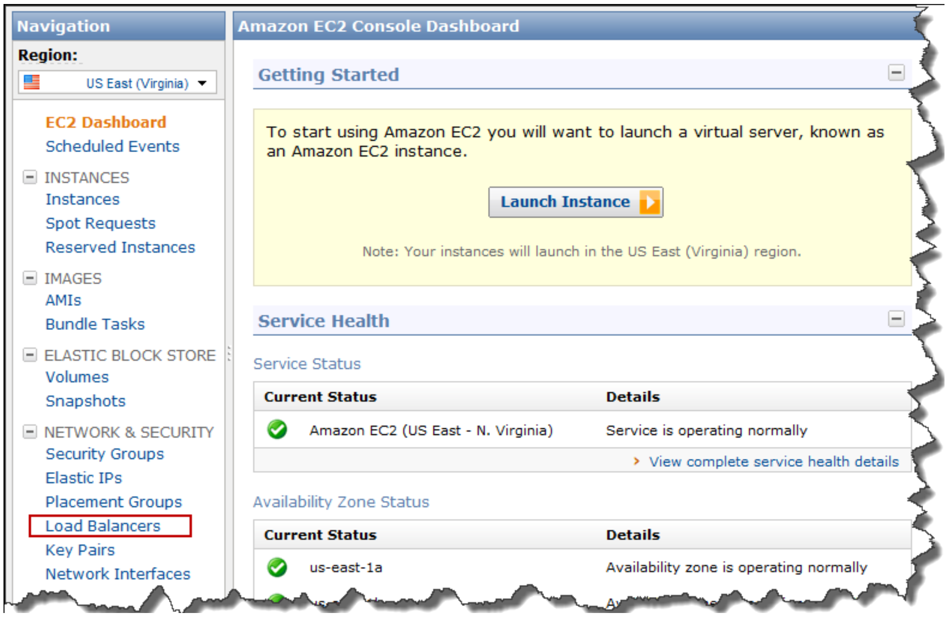
Click Task 1: Configure Listeners (p. 12) to get started.

Task 1: Configure Listeners

To configure listeners for your load balancer

1. Start the **Create Load Balancer** wizard:

   a. In the Amazon EC2 Console Dashboard page, click **Load Balancers** in the **Navigation** pane.



To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

Note: Your instances will launch in the US East (Virginia) region.

Service Status

Current Status	Details
✓ Amazon EC2 (US East - N. Virginia)	Service is operating normally

Availability Zone Status

Current Status	Details
✓ us-east-1a	Availability zone is operating normally

b. On the **Load Balancers** page, click **Create Load Balancers**.

2. On the **DEFINE LOAD BALANCER** page, make the following selections:

   a. Enter a name for your load balancer (e.g., `MyLoadBalancer`).

   b. Leave **CreateLB inside** set to `EC2` because in this example you'll create your load balancer in Amazon EC2.

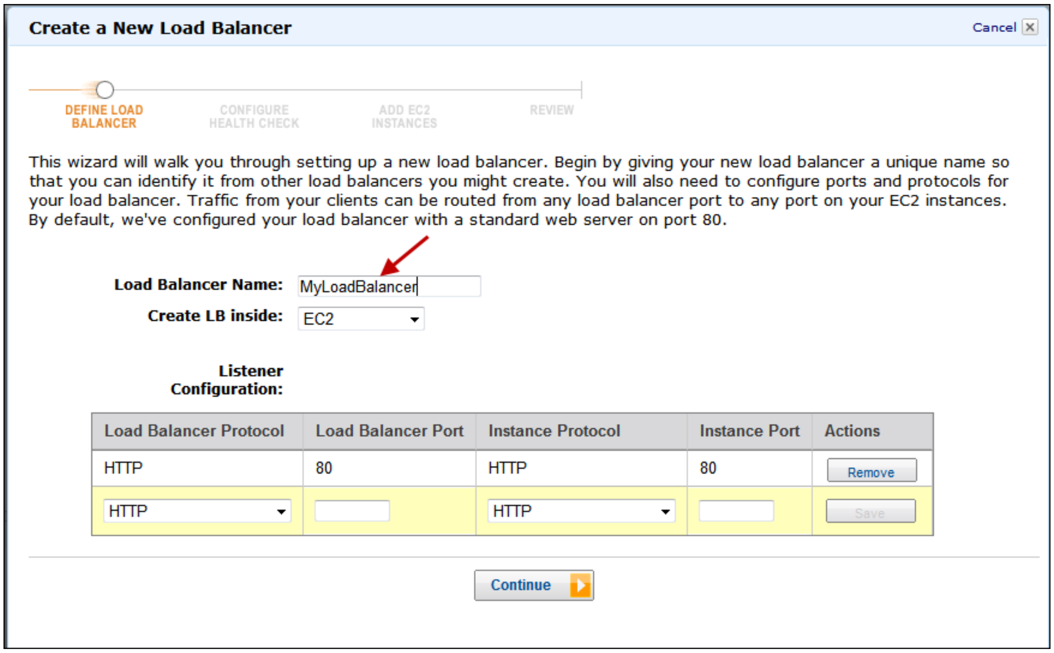
   ### Note

   If you want to create a load balancer for your EC2 instances inside Amazon Virtual Private Cloud (Amazon VPC), see Deploying Elastic Load Balancing in Amazon VPC (p. 67)

   c. Leave **Listener Configuration** set to the default value for this example.

   ### Important

   The default settings require that your Amazon EC2 HTTP servers are active and accepting requests on port 80.



This wizard will walk you through setting up a new load balancer. Begin by giving your new load balancer a unique name so that you can identify it from other load balancers you might create. You will also need to configure ports and protocols for your load balancer. Traffic from your clients can be routed from any load balancer port to any port on your EC2 instances. By default, we've configured your load balancer with a standard web server on port 80.

Load Balancer Name: MyLoadBalancer

Create LB inside: EC2

Listener Configuration:

Load Balancer Protocol	Load Balancer Port	Instance Protocol	Instance Port	Actions
HTTP	80	HTTP	80	Remove
HTTP		HTTP		Save

3. Click **Continue**.

Task 2: Configure Health Check

Now that you have defined the load balancer's name and have kept the default settings for the listener ports, you're ready to use the **CONFIGURE HEALTH CHECK** page of the **Create Load Balancer** wizard.

Elastic Load Balancing routinely checks the health of each load-balanced Amazon EC2 instance based on the configurations that you specify. If Elastic Load Balancing finds an unhealthy instance, it stops sending traffic to the instance and reroutes traffic to healthy instances.

To configure a health check for your Amazon EC2 instances

1. On the **CONFIGURE HEALTH CHECK** page of the **Create a New Load Balancer** wizard, set the following configurations:

   a. Leave **Ping Protocol** set to its default value of `HTTP`.

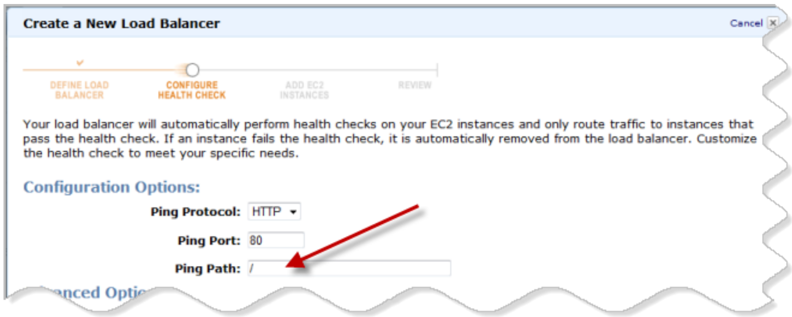
   b. Leave **Ping Port** set to its default value of `80`.

      Elastic Load Balancing pings the port you choose (in this example, port 80) to send health check queries to your Amazon EC2 instances.

      ⚠ **Important**

      Your Amazon EC2 instances must accept incoming traffic on the ping port. This example assumes that each of your instances has a working HTTP server that accepts incoming traffic on port 80.

   c. In the **Ping Path** field, replace the default value with a single forward slash ("/").



Your load balancer will automatically perform health checks on your EC2 instances and only route traffic to instances that pass the health check. If an instance fails the health check, it is automatically removed from the load balancer. Customize the health check to meet your specific needs.

Configuration Options:

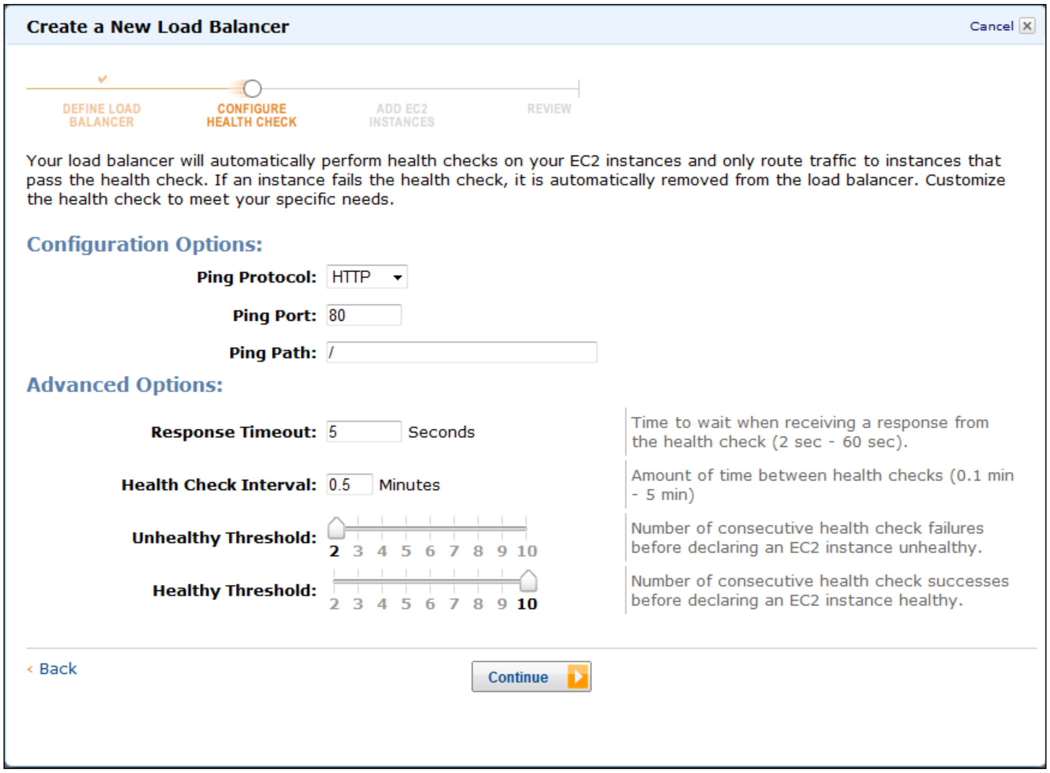
Ping Protocol: HTTP

Ping Port: 80

Ping Path: /

Elastic Load Balancing sends health check queries to the path you specify in **Ping Path**. This example uses a single forward slash so that Elastic Load Balancing sends the query to your HTTP server's default home page, whether that default page is named `index.html`, `default.html`, or a different name.

d. Leave the **Advanced Options** set to their default values.



Your load balancer will automatically perform health checks on your EC2 instances and only route traffic to instances that pass the health check. If an instance fails the health check, it is automatically removed from the load balancer. Customize the health check to meet your specific needs.

Configuration Options:

Ping Protocol: HTTP

Ping Port: 80

Ping Path: /

Advanced Options:

Response Timeout: 5 Seconds — Time to wait when receiving a response from the health check (2 sec - 60 sec).

Health Check Interval: 0.5 Minutes — Amount of time between health checks (0.1 min - 5 min)

Unhealthy Threshold: 2 (2 3 4 5 6 7 8 9 10) — Number of consecutive health check failures before declaring an EC2 instance unhealthy.

Healthy Threshold: 10 (2 3 4 5 6 7 8 9 10) — Number of consecutive health check successes before declaring an EC2 instance healthy.

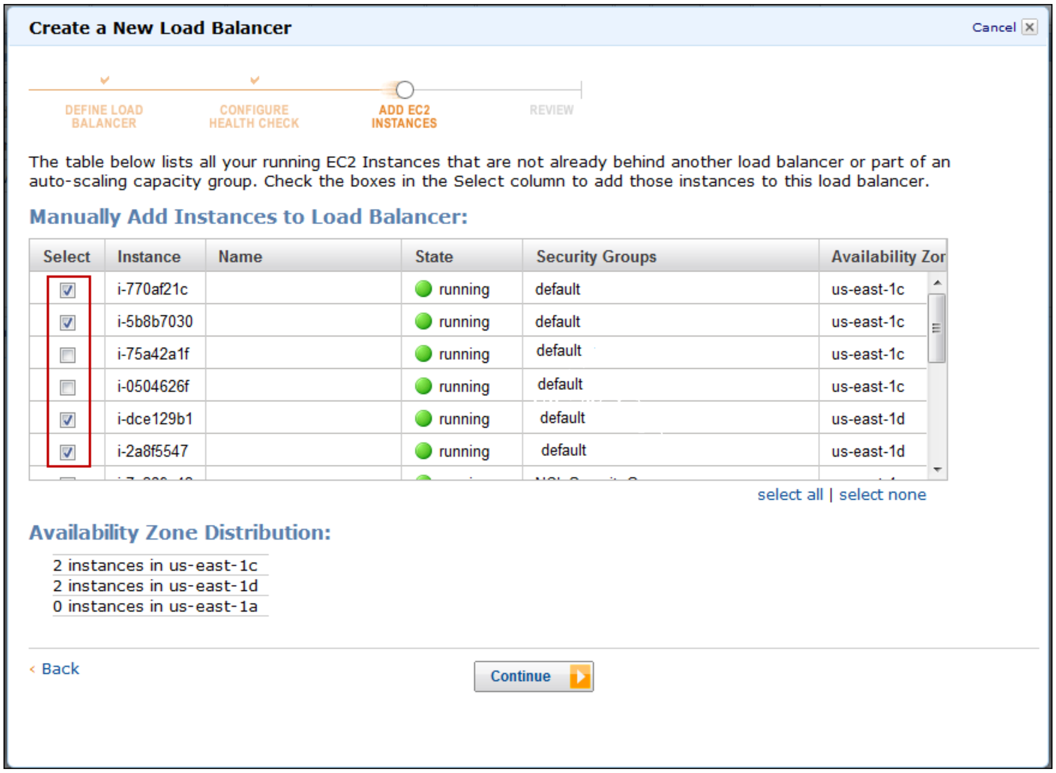
2. Click **Continue**.

Task 3: Register Amazon EC2 Instances

Now that you've made your configuration choices you're ready to add your EC2 instances using the **ADD INSTANCES** page of the **Create Load Balancer** wizard.

To register your Amazon EC2 instances

1. On the **ADD INSTANCES** page, check the boxes in the **Select** column to add instances to your load balancer.



Create a New Load Balancer

The table below lists all your running EC2 Instances that are not already behind another load balancer or part of an auto-scaling capacity group. Check the boxes in the Select column to add those instances to this load balancer.

Manually Add Instances to Load Balancer:

Select	Instance	Name	State	Security Groups	Availability Zor
☑	i-770af21c		● running	default	us-east-1c
☑	i-5b8b7030		● running	default	us-east-1c
□	i-75a42a1f		● running	default	us-east-1c
□	i-0504626f		● running	default	us-east-1c
☑	i-dce129b1		● running	default	us-east-1d
☑	i-2a8f5547		● running	default	us-east-1d

Availability Zone Distribution:

2 instances in us-east-1c
2 instances in us-east-1d
0 instances in us-east-1a

2. Click **Continue**.

Task 4: Review Settings

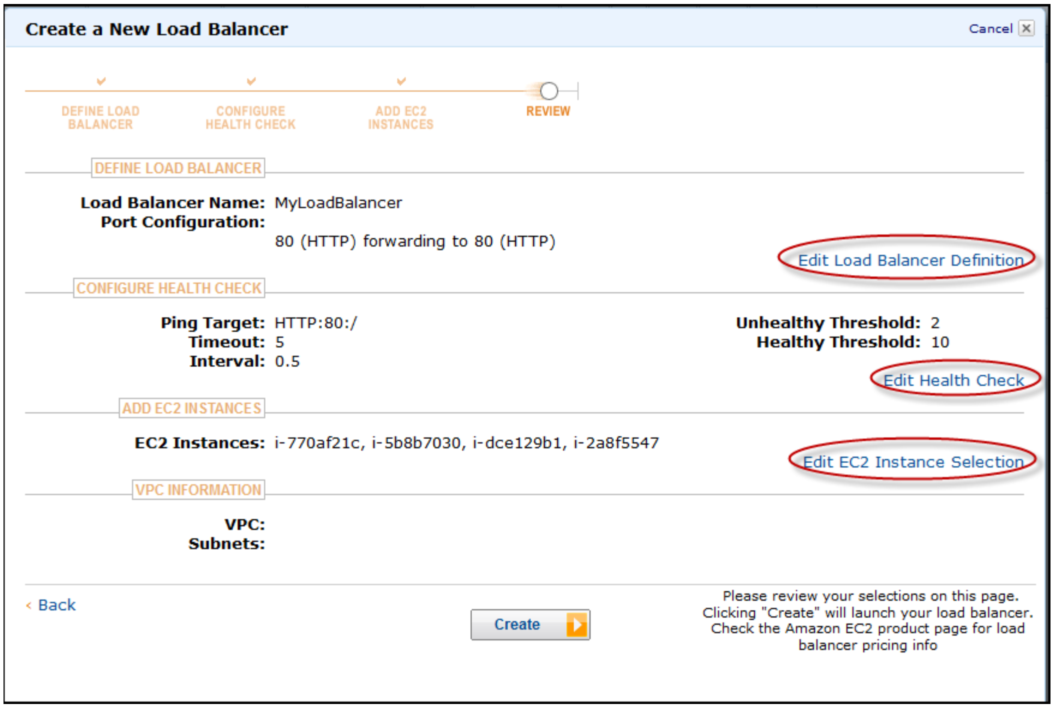
Now that you've added your EC2 instances to your load balancer it's time to use the **REVIEW** page of the **Create a New Load Balancer** wizard to review the settings you have selected.

To review your settings

1. On the **Review** page of the **Create a New Load Balancer** wizard, check your settings. You can make changes to the settings by clicking the edit link for each setting.

   Note

   The settings can be modified even after you've created your load balancer. Use command line interface (CLI) or the Query API to modify settings on your existing load balancer. For information on installing and using the tools, see Get Set Up with Elastic Load Balancing Interfaces (p. 20)



DEFINE LOAD BALANCER

Load Balancer Name: MyLoadBalancer
Port Configuration:
80 (HTTP) forwarding to 80 (HTTP)

CONFIGURE HEALTH CHECK

Ping Target: HTTP:80:/
Timeout: 5
Interval: 0.5
Unhealthy Threshold: 2
Healthy Threshold: 10

ADD EC2 INSTANCES

EC2 Instances: i-770af21c, i-5b8b7030, i-dce129b1, i-2a8f5547

VPC INFORMATION

VPC:
Subnets:

Please review your selections on this page. Clicking "Create" will launch your load balancer. Check the Amazon EC2 product page for load balancer pricing info

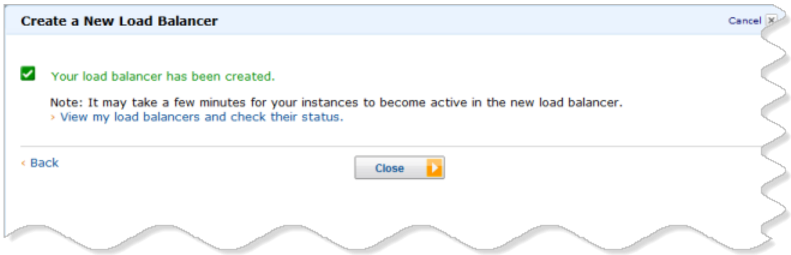
2. Click **Create**.

Task 5: Create a Load Balancer

Now that you've made your configuration choices, added your instances, reviewed your selections, and have clicked **Create**, you're ready to create your load balancer.

To create your load balancer

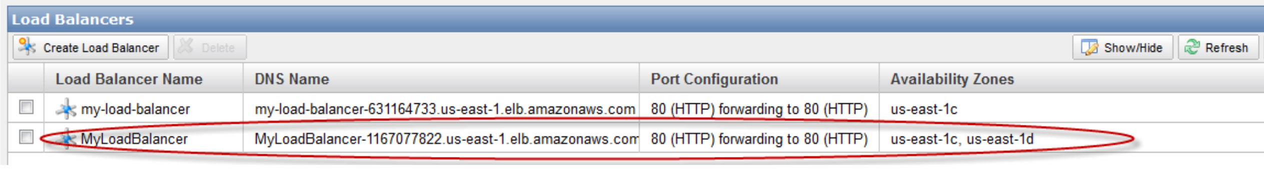
1. After you click **Create** button in the **REVIEW** page, a confirmation window opens. Click **Close**.



☑ Your load balancer has been created.

Note: It may take a few minutes for your instances to become active in the new load balancer.
› View my load balancers and check their status.

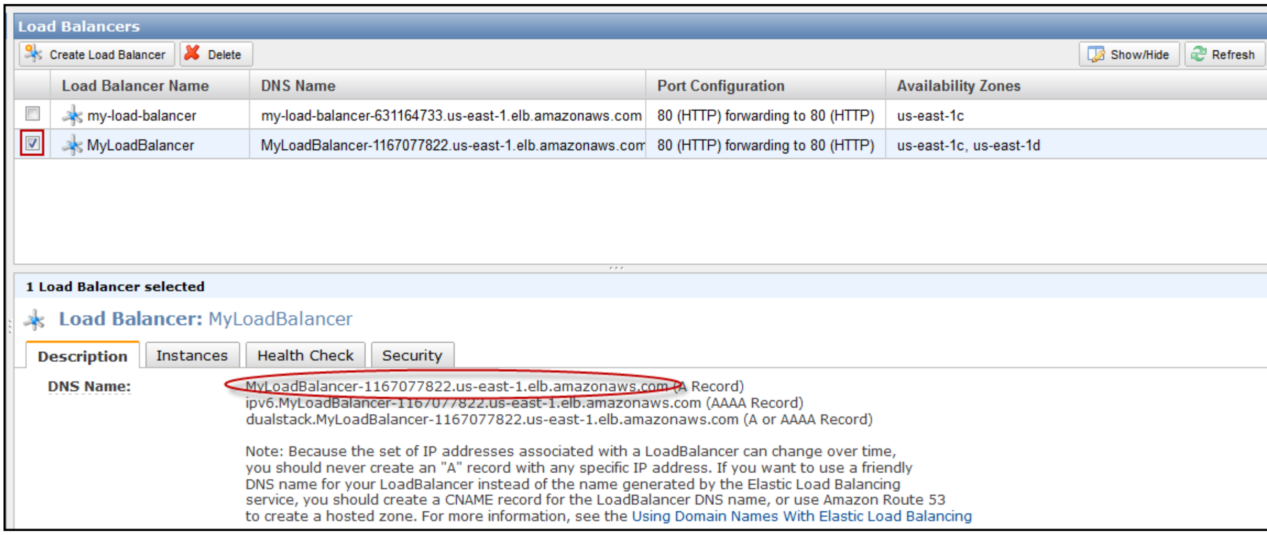
2. When the confirmation window closes, the Load Balancers page opens. Your new load balancer now appears in the list.



	Load Balancer Name	DNS Name	Port Configuration	Availability Zones
□	my-load-balancer	my-load-balancer-631164733.us-east-1.elb.amazonaws.com	80 (HTTP) forwarding to 80 (HTTP)	us-east-1c
□	MyLoadBalancer	MyLoadBalancer-1167077822.us-east-1.elb.amazonaws.com	80 (HTTP) forwarding to 80 (HTTP)	us-east-1c, us-east-1d

3. Select the check box next to your load balancer.

   A set of tabs opens with details about your new load balancer.



	Load Balancer Name	DNS Name	Port Configuration	Availability Zones
☐	my-load-balancer	my-load-balancer-631164733.us-east-1.elb.amazonaws.com	80 (HTTP) forwarding to 80 (HTTP)	us-east-1c
☑	MyLoadBalancer	MyLoadBalancer-1167077822.us-east-1.elb.amazonaws.com	80 (HTTP) forwarding to 80 (HTTP)	us-east-1c, us-east-1d

1 Load Balancer selected

Load Balancer: MyLoadBalancer

DNS Name: MyLoadBalancer-1167077822.us-east-1.elb.amazonaws.com (A Record)
ipv6.MyLoadBalancer-1167077822.us-east-1.elb.amazonaws.com (AAAA Record)
dualstack.MyLoadBalancer-1167077822.us-east-1.elb.amazonaws.com (A or AAAA Record)

Note: Because the set of IP addresses associated with a LoadBalancer can change over time, you should never create an "A" record with any specific IP address. If you want to use a friendly DNS name for your LoadBalancer instead of the name generated by the Elastic Load Balancing service, you should create a CNAME record for the LoadBalancer DNS name, or use Amazon Route 53 to create a hosted zone. For more information, see the Using Domain Names With Elastic Load Balancing

4. To test your load balancer, copy the **DNS Name** value that is listed in the **Description** tab and paste it into the address field of an Internet-connected web browser. If your load balancer is working, you will see the default page of your HTTP server.

Congratulations! You've successfully created a basic load balancer. The load balancer is live and has started incurring the standard Elastic Load Balancing usage fees. You can either delete the load balancer now and incur minimal hourly (typically less than a dollar) charges, or continue using the load balancer.

Note

If you are a *new* AWS customer and are using the free usage tier, you can continue using the load balancer. Go to AWS Free Usage Tier to check the number of hours available to you.

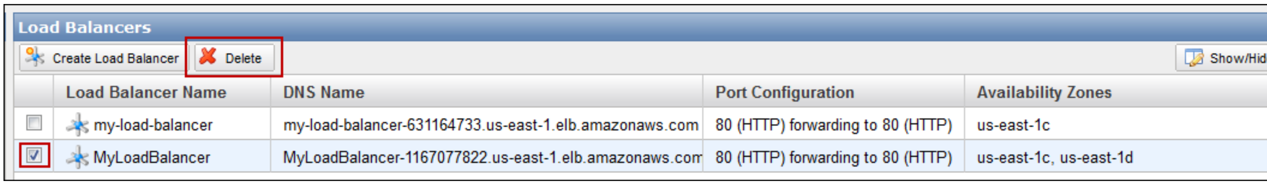
- For information on deleting your load balancer, see the following section.
- Elastic Load Balancing supports the load balancing of applications using HTTP, HTTPS (Secure HTTP), TCP, and SSL (Secure TCP) listener protocols. For a quick overview of the different configurations available for your load balancer, see Elastic Load Balancing Listener Configurations Quick Reference (p. 36).
- For information on some common Elastic Load Balancing user scenarios, and the tasks needed to accomplish efficient distribution of application loads among your Amazon EC2 instances, see Using Elastic Load Balancing (p. 39).
- For information about Elastic Load Balancing usage rates, go to the Elastic Load Balancing product page.

Task 6: Delete Your Load Balancer

As soon as your load balancer becomes available, you’re billed for each hour or partial hour that you keep the load balancer running. After you've decided that you no longer need the load balancer, you can delete it.

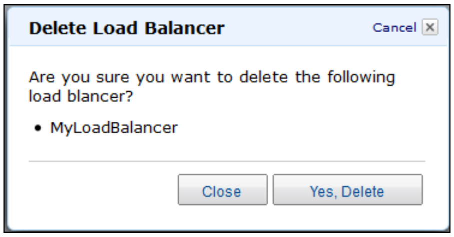
To delete your load balancer

1. On the Amazon EC2 Console Dashboard page, click **Load Balancers** in the **Navigation** pane.
2. On the **Load Balancers** page, select the check box next to the load balancer you want to delete, and then click **Delete**.



	Load Balancer Name	DNS Name	Port Configuration	Availability Zones
☐	my-load-balancer	my-load-balancer-631164733.us-east-1.elb.amazonaws.com	80 (HTTP) forwarding to 80 (HTTP)	us-east-1c
☑	MyLoadBalancer	MyLoadBalancer-1167077822.us-east-1.elb.amazonaws.com	80 (HTTP) forwarding to 80 (HTTP)	us-east-1c, us-east-1d

3. In the **Delete Load Balancer** windows, click **Yes, Delete**.



Delete Load Balancer

Are you sure you want to delete the following load blancer?

- MyLoadBalancer

Elastic Load Balancing deletes the load balancer. As soon as the load balancer is deleted, you stop incurring charges for that load balancer.

Note

Even after you delete a load balancer, the Amazon EC2 instances associated with the load balancer continue to run. You will continue to incur charges on the Amazon EC2 instances while they are running. For information on stopping your EC2 instances, go to Stopping and Starting Instances in the *Amazon EC2 User Guide*. For information on terminating your EC2 instances, go to Terminate Your Instance.

Get Set Up with Elastic Load Balancing Interfaces

Topics

You can create, access, and manage your load balancers using one of the Amazon Web Services (AWS) Elastic Load Balancing interfaces: the AWS Management Console, the command line interface (CLI), the Query API, the SOAP API, or the SDK for Elastic Load Balancing.

The AWS Management Console provides a web-based interface for many AWS products. You can use the console to make requests to an API, or, if you prefer, you can make requests programmatically directly to an API. Before you begin using the AWS Management Console, you will need to create an AWS account alias and sign in using that account. If you already have an AWS account, you do not need to create a new one. You will use the Amazon EC2 console for Elastic Load Balancing requests. If you haven't signed up for AWS yet,

- Go to http://aws.amazon.com and click the **Sign Up Now** button.
- Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

The command line interface, the Query API, and the SDKs have to be installed before you can use them.

Elastic Load Balancing provides a command line interface (CLI) to access Elastic load Balancing functionality without using the AWS Management Console, the APIs, or the SDKs. The 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. There is no installation required, simply download and unzip it. For more information on installing and using the Elastic Load Balancing command line interface, see Installing the Command Line Interface (p. 21).

Elastic Load Balancing provides a Query API you can use to programmatically access Elastic Load Balancing functionality. Query requests are HTTP or HTTPS requests that use the HTTP verbs GET or POST and a Query parameter named Action or Operation. Action is used throughout this documentation, although Operation is supported for backward compatibility with other AWS Query APIs. For information on using the Query API, see Using the Query API (p. 26).

You can access 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 for the particular service. The WSDL references an XML schema document, which strictly defines the data types that might appear in SOAP requests and responses. For information on using the SOAP API, see Using the SOAP API (p. 29)

AWS SDKs make it easier for developers to build applications that tap into cost-effective, scalable, and reliable AWS infrastructure services. With AWS SDKs, you can get started in minutes with a single, downloadable package that includes the library, code samples, and reference documentation. You can access Elastic Load Balancing programmatically using the SDKs in Java, .NET, PHP, or Ruby. For more information on downloading and using the AWS SDKs, see Using the SDKs (p. 32).

Installing the Command Line Interface

This section describes how to set up the Elastic Load Balancing command line tool.

Process for Installing the Command Line Tool

Task 1: Download the Command Line Interface (p. 21)
Task 2: Set the JAVA_HOME Environment Variable (p. 21)
Task 3: Set the AWS_ELB_HOME Environment Variable (p. 23)
Task 4: Set the AWS_CREDENTIAL_FILE Environment Variable (p. 24)
Task 5: Set the Region (p. 25)

Note

As a convention, command line text is prefixed with a generic **`PROMPT>`** command line prompt. The actual command line prompt on your computer is likely to be different. We also use **`$`** to indicate a Linux/UNIX–specific command and **`C:\>`** for a Windows–specific command. Although we don't provide explicit instructions, the tool also works on the Mac OS X. (Commands on the Mac OS X resemble the Linux and UNIX commands.) The example output resulting from the command is shown immediately thereafter without any prefix.

Task 1: Download the Command Line Interface

The command line tool is available as a ZIP file on the Elastic Load Balancing Developer Tools website. The tool is written in Java and includes shell scripts for both Windows and Linux/UNIX/Mac OSX. The ZIP file is self-contained; no installation is required. You just download it and unzip it.

Some additional setup is required before you can use the tool. These steps are discussed next.

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/UNIX and Windows, go to 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/UNIX paths (e.g., /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/UNIX 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

   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.

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

   On Linux and UNIX, you can update your PATH as follows:

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

   On Windows the syntax is slightly different:

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

   Note

   The **setx** command does not use the = sign.

4. On Linux or UNIX, 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/UNIX 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 and UNIX, 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

You must also provide your AWS credentials to the command line tool. The command line tool reads your credentials from a credential file that you create on your local system.

You can either specify your credentials 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 set up security credentials for your command line tool

1. Log in to the AWS security credentials web site.
2. Retrieve an access key and its corresponding secret key.

   a. Scroll down to the **Access Credentials** section and select the **Access Keys** tab.

   b. Locate an active Access Key in the **Your Access Keys** list.

   c. To display the Secret Access Key, click **Show** in the **Secret Access Key** column.

   d. Write down the keys or save them.

   e. If no Access Keys appear in the list, click **Create a New Access Key** and follow the on-screen prompts.

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 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/UNIX 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 5: Set the Region

By default, the Elastic Load Balancing tools use the Eastern United States Region (`us-east-1`) with the `elasticloadbalancing.us-east-1.amazonaws.com` service endpoint URL. If your instances are in a different region, you must specify the region where your instances reside. For example, if your instances are in Europe, you must specify the eu-west1 Region by using the `--region eu-west-1` parameter or by setting the `AWS_ELB_URL` environment variable.

This section describes how to specify a different Region by changing the service endpoint URL.

To specify a different region

1. To view available regions go to Regions and Endpoints in the *Amazon Web Services General Reference*.
2. If you want to change the service endpoint, set the `AWS_ELB_URL` environment variable.
   - The following Linux/UNIX example sets `AWS_ELB_URL` to the EU (Ireland) Region.

   ```
   $ export AWS_ELB_URL=https://elasticloadbalancing.eu-west-1.amazonaws.com
   ```

   - The following Windows example sets `AWS_ELB_URL` to the EU (Ireland) Region.

   ```
   C:\> set AWS_ELB_URL=https://elasticloadbalancing.eu-west-1.amazonaws.com

   C:\> setx AWS_ELB_URL https://elasticloadbalancing.eu-west-1.amazonaws.com
   ```

You're ready to start accessing Elastic Load Balancing using the command line interface (CLI). For descriptions of all the Elastic Load Balancing commands, see Elastic Load Balancing Quick Reference Card.

Using the Query API

Query requests are HTTP or HTTPS requests that use the HTTP verb GET or POST and a Query parameter named Action or Operation. Action is used throughout this documentation, although Operation is supported for backward compatibility with other AWS Query APIs.

Endpoints

For information about this product's regions and endpoints, go to Regions and Endpoints in the Amazon Web Services General Reference.

Query Parameters

Each Query request must include some common parameters to handle authentication and selection of an action. For more information, go to Common Query Parameters in the *Elastic Load Balancing API Reference*.

Note

Some API operations take lists of parameters. These lists are specified using the following notation: `param.member.n`. Values of n are integers starting from 1. All lists of parameters must follow this notation, including lists that only contain one parameter. For example, a Query parameter list looks like this:

```
&attribute.member.1=this
&attribute.member.2=that
```

The Request ID

In every response from Amazon Web Services (AWS), you will find `ResponseMetadata` that contains a string element called `RequestId`. This is simply a unique identifier AWS assigns to this request for tracking and troubleshooting purposes.

To improve readability of the API documentation and reduce redundancy, `RequestId` is not listed on the individual API documentation pages.

Query API Authentication

You can send Query requests over either HTTP or HTTPS. Regardless of which protocol you use, you must include a signature in every Query request. This section describes how to create the signature. The method described in the following procedure is known as *signature version 2*.

To create the signature

1. Create the canonicalized query string that you will need later in this procedure:

    a. Sort the UTF-8 query string components by parameter name with natural byte ordering. The parameters can come from the GET URI or from the POST body (when `Content-Type` is `application/x-www-form-urlencoded`).

    b. URL encode the parameter name and values according to the following rules:

- Do not URL encode any of the unreserved characters that RFC 3986 defines. These unreserved characters are A-Z, a-z, 0-9, hyphen (-), underscore (_), period (.), and tilde (~).
- Percent encode all other characters with %XY, where X and Y are hex characters 0-9 and uppercase A-F.
- Percent encode extended UTF-8 characters in the form %XY%ZA, and so forth.
- Percent encode the space character as %20. Don't use +.

Note

Currently all AWS service parameter names use unreserved characters, so you don't need to encode them. However, you might want to include code to handle parameter names that use reserved characters, for possible future use.

c. Separate the encoded parameter names from their encoded values with the equal sign (=) (ASCII character 61), even if the parameter value is empty.

d. Separate the name-value pairs with an ampersand (&) (ASCII code 38).

2. Create the string to sign according to the following pseudo-grammar (the `"\n"` represents an ASCII newline).

```
StringToSign = HTTPVerb + "\n" +
ValueOfHostHeaderInLowercase + "\n" +
HTTPRequestURI + "\n" +
CanonicalizedQueryString <from the preceding step>
```

The `HTTPRequestURI` component is the HTTP absolute path component of the URI up to, but not including, the query string. If the `HTTPRequestURI` is empty, use a forward slash (`/`).

3. Calculate an RFC 2104-compliant HMAC with the string you just created, your Secret Access Key (p. 115) as the key, and SHA256 or SHA1 as the hash algorithm.
For more information, go to http://www.ietf.org/rfc/rfc2104.txt.
4. Convert the resulting value to base64.
5. Use the resulting value as the value of the *`Signature`* request parameter.

Important

The final signature you send in the request must be URL encoded as specified in RFC 3986 (for more information, go to http://www.ietf.org/rfc/rfc3986.txt). If your toolkit URL encodes your final request, then it handles the required URL encoding of the signature. If your toolkit doesn't URL encode the final request, then make sure to URL encode the signature before you include it in the request. Most importantly, make sure the signature is URL encoded *only once*. A common mistake is to URL encode it manually during signature formation, and then again when the toolkit URL encodes the entire request.

For detailed descriptions of the Elastic Load Balancing API actions, see Elastic Load Balancing API Reference.

Query Example

Example EnableAvailabiltyZoneForLoadBalancer API Request

This example uses the Elastic Load Balancing action `EnableAvailabilityZonesForLoadBalancer`.

```
https://elasticloadbalancing.amazonaws.com/?AvailabilityZones.member.1=us-east-
1c
&LoadBalancerName=ReferenceLB1
&Action=EnableAvailabilityZonesForLoadBalancer
&Version=2009-05-15
&AWSAccessKeyId=<Your AWS Access Key ID>
&SignatureVersion=2
&SignatureMethod=HmacSHA1
&Timestamp=2009-02-17T05%3A13%3A00.000Z
```

The following is the string to sign.

```
GET\n
elasticloadbalancing.amazonaws.com\n
/\n
AWSAccessKeyId=<Your AWS Access Key ID>
&Action=EnableAvailabilityZonesForLoadBalancer
&AvailabilityZones.member.1=us-east-1c
&LoadBalancerName=ReferenceLB1
&SignatureMethod=HmacSHA1
&SignatureVersion=2
&Timestamp=2009-02-17T05%3A13%3A00.000Z
&Version=2009-05-15
```

The following is the signed request.

```

https://elasticloadbalancing.amazonaws.com/?Action=EnableAvailabilityZonesFor
LoadBalancer
&AvailabilityZones.member.1=us-east-1c
&AWSAccessKeyId=<Your AWS Access Key ID>
&LoadBalancerName=ReferenceLB1
&SignatureVersion=2
&SignatureMethod=HmacSHA1
&Timestamp=2009-10-17T05%3A13%3A00.000Z
&Signature=<URLEncode(Base64Encode(Signature))>
&Version=2009-05-15
```

Using the SOAP API

Topics

Endpoints

For information about this product's regions and endpoints, go to Regions and Endpoints in the Amazon Web Services General Reference.

WSDL and Schema Definitions

You can access 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, which defines the operations and security model for the particular service. The WSDL references an XML Schema document, which strictly defines the data types that might appear in SOAP requests and responses. For more information on WSDL and SOAP, see Web Services References (p. 32).

Note

Elastic Load Balancing supports SOAP only through HTTPS.

All schemas have a version number. The version number appears in the URL of a schema file and in a schema's target namespace. This makes upgrading easy by differentiating requests based on the version number.

Programming Language Support

Because the SOAP requests and responses in Elastic Load Balancing follow current standards, nearly any programming language can be used.

Note

AWS provides libraries, sample code, tutorials, and other resources for software developers who prefer to build applications using language-specific APIs instead of Elastic Load Balancing's SOAP and Query APIs. These libraries provide basic functions (not included in Elastic Load Balancing's SOAP and Query APIs), such as request authentication, request retries, and error handling so that it's easier to get started. Libraries and resources are available for the following languages:

- Java
- PHP
- Ruby
- Windows and .NET

For libraries and sample code in all languages, go to Sample Code & Libraries.

Request Authentication

Elastic Load Balancing complies with the current WS-Security standard, which requires you to hash and sign SOAP requests for integrity and non-repudiation. WS-Security defines profiles which are used to implement various levels of security. Secure SOAP messages use the BinarySecurityToken profile, consisting of an X.509 certificate with an RSA public key.

The following is the content of an insecure `RunInstances` operation (using EC2 as an example):

```
<Runinstances xmlns="http://ec2.amazonaws.com/doc/2009-05-05">
    <instancesSet>
        <item>
            <imageId>ami-60a54009</imageId>
            <minCount>1</minCount>
            <maxCount>3</maxCount>
        </item>
    </instancesSet>
    <groupSet/>
</RunInstances>
```

To secure the request, we add the BinarySecurityToken element.

The secure version of the request begins with the following:

```

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

  <SOAP-ENV:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd">
      <wsse:BinarySecurityToken
      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wsse
curity-utility-1.0.xsd"
      EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
soap-message-security-1.0#Base64Binary"
     ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-X.509-
token-profile-1.0#X.509v3"
      wsu:Id="CertId-1064304">....many, many lines of base64 encoded
      X.509 certificate...</wsse:BinarySecurityToken>
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:SignedInfo>
          <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-
exc-c14n#"></ds:CanonicalizationMethod>
         <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
sha1"></ds:SignatureMethod>
          <ds:Reference URI="#id-17984263">
            <ds:Transforms>
              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#"></ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmld
sig#sha1"></ds:DigestMethod>
            <ds:DigestValue>0pjZ1+TvgPf6uG7o+Yp3l2YdGZ4=</ds:DigestValue>
          </ds:Reference>
          <ds:Reference URI="#id-15778003">
            <ds:Transforms>
              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#"></ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmld
sig#sha1"></ds:DigestMethod>
            <ds:DigestValue>HhRbxBBmc2OO348f8nLNZyo4AOM=</ds:DigestValue>
          </ds:Reference>
        </ds:SignedInfo>
       <ds:SignatureValue>bmVx24Qom4kd9QQtclxWIlgLk4QsQBPaKESi79x479xgbO9PEStXMi
HZuBAi9luuKdNTcfQ8UE/d
       jjHKZKEQRCOlLVy0Dn5ZL1RlMHsv+OzJzzvIJFTq3LQKNrzJzsNe</ds:SignatureValue>

        <ds:KeyInfo Id="KeyId-17007273">
          <wsse:SecurityTokenReference
              xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd" wsu:Id="STRId-22438818">
            <wsse:Reference URI="#CertId-1064304"
                            ValueType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-X.509-token-profile-1.0#X.509v3">
            </wsse:Reference>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
      <wsu:Timestamp
          xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd" wsu:Id="id-17984263">
        <wsu:Created>2006-06-09T10:57:35Z</wsu:Created>
        <wsu:Expires>2006-06-09T11:02:35Z</wsu:Expires>
      </wsu:Timestamp>
    </wsse:Security>
  </SOAP-ENV:Header>
```

If you are matching this against requests generated by Elastic Load Balancing supplied libraries, or those of another vendor, the following are the most important elements.

Elements

- **BinarySecurityToken—**Contains the X.509 certificate in base64 encoded PEM format
- **Signature—**Contains an XML digital signature created using the canonicalization, signature algorithm, and digest method
- **Timestamp—**Requests to Elastic Load Balancing are valid within 5 minutes of this value to help prevent replay attacks

The Response Structure

In response to a request, the Elastic Load Balancing service returns an XML data structure that conforms to an XML schema defined as part of the Elastic Load Balancing WSDL. The structure of a XML response is specific to the associated request.

The following is an example response (using EC2 as an example):

```
 <RuninstancesResponse xmlns="http://ec2.amazonaws.com/doc/2009-05-05">
  <reservationId>r-47a5402e</reservationId>
  <ownerId>UYY3TLBUXIEON5NQVUUX6OMPWBZIQNFM</ownerId>
  <groupSet>
    <item>
      <groupId>default</groupId>
    </item>
  </groupSet>
  <instancesSet>
    <item>
       <InstanceId>i-2ba64342</InstanceId>
       <imageId>ami-60a54009</imageId>
       <InstanceState>
        <code>0</code>
    <name>pending</name>
      </InstanceState>
      <DNSName></DNSName>
    </item>
    <item>
      <InstanceId>i-2bc64242</InstanceId>
      <imageId>ami-60a54009</imageId>
      <InstanceState>
        <code>0</code>
    <name>pending</name>
      </InstanceState>
      <DNSName>ec2-67-202-51-176.compute-1.amazonaws.com </DNSName>
    </item>
    <item>
      <InstanceId>i-2be64332</InstanceId>
      <imageId>ami-60a54009</imageId>
      <InstanceState>
        <code>0</code>
    <name>pending</name>
      </InstanceState>
      <DNSName>ec2-67-202-51-122.compute-1.amazonaws.com</DNSName>
      <keyName>example-key-name</keyName>
      <instanceType>m1.small</instanceType>
      <launchTime>2007-08-07T11:54:42.000Z</launchTime>
    </item>
  </instancesSet>
</RunInstancesResponse>
```

Web Services References

For more information about using web services, go to any of the following resources:

- Web Service Description Language (WSDL)
- WS-Security BinarySecurityToken Profile

Using the SDKs

The following table lists the available SDKs and third-party libraries you can use to access Elastic Load Balancing programmatically.

Type of Access	Description
AWS SDKs	AWS provides the following SDKs: • AWS SDK for Java Documentation • AWS SDK for .NET Documentation • AWS SDK for PHP Documentation • AWS SDK for Ruby Documentation
Third-Party Libraries	Developers in the AWS developer community also provide their own libraries, which you can find at the following AWS developer centers: • AWS Java Developer Center • AWS PHP Developer Center • AWS Python Developer Center • AWS Ruby Developer Center • AWS Windows and .NET Developer Center

Choosing Listeners for Your Load Balancer

Elastic Load Balancing supports the load balancing of applications using HTTP, HTTPS (Secure HTTP), TCP, and SSL (Secure TCP) protocols. You can specify the protocols for the front-end connections (client to load balancer) and the back-end connections (load balancer to back-end instance) independently. If you choose HTTPS/SSL protocols for your front-end connection, the back-end connection to the instance can either be in plain text or HTTPS/SSL. If you are choosing HTTP/TCP for your front-end connection, the back-end connection to the instance can be HTTPS/SSL.

The acceptable ports for both HTTPS/SSL and HTTP/TCP connections are 25, 80, 443, and 1024-65535.

By default, your load balancer is set to use the HTTP protocol with port 80 for the front-end connection and the back-end connection. The default settings can be changed using the AWS Management Console, the Query API, the command line interface (CLI), or the SDKs.

Using TCP/SSL with Elastic Load Balancing

When you use TCP for both front-end and back-end listeners, your load balancer will forward the request to the back-end instances without modification.

If you use SSL (secure TCP) for your front-end listener, you will have to install an SSL server certificate on your load balancer. The load balancer uses the certificate to first terminate and then decrypt the requests before sending the requests to the back-end instances. With this configuration, you can also choose to configure ciphers for SSL negotiation between your client and the load balancer.

Using HTTP/HTTPS with Elastic Load Balancing

When you use HTTP for both front-end and back-end listeners, your load balancer terminates the request and also parses the headers in the request. This is the default configuration provided by Elastic Load Balancing.

If you use HTTPS (secure HTTP) for your front-end listener, you must install 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. Additionally, when you use HTTPS, the load balancer inserts or updates the X-Forward headers and can insert or update cookies for sticky sessions. For more information on X-Forward headers, see Elastic Load Balancing Concepts (p. 4) With this configuration, you can also choose to configure ciphers for SSL negotiation between your client and the load balancer.

Not all HTTP extensions are supported in the load balancer, and 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.

Advantages of Using HTTPS/SSL with Elastic Load Balancing

Elastic Load Balancing provides SSL support for connections between clients and the load balancer and also for connections between the load balancer and your back-end application instances. Support for an end-to-end HTTPS/SSL connection enables traffic encryption on those network segments that initiate HTTPS/SSL connections.

There are several advantages to using HTTPS/SSL connections with your load balancer:

- The SSL server certificate used to terminate client connections can be managed centrally on the load balancer, rather than on every individual application instance.
- The work of encrypting and decrypting SSL traffic is moved from the application instance to the load balancer.
- The load balancer can ensure session affinity or "sticky sessions" by terminating the incoming HTTPS request and then re-encrypting the content to send to the back-end application instance.
- All of the features available for HTTP can be used with HTTPS connections.

Using HTTPS/SSL protocols for both front-end and back-end connections ensures end-to-end traffic encryption. If you are using SSL and do not want Elastic Load Balancing to terminate, you can use a TCP listener and install certificates on all the back-end instances handling requests.

To enable HTTPS support, use AWS Identity and Access Management (IAM) to upload your SSL certificate and key. After you upload your certificate, specify its Amazon Resource Name (ARN) when you create a new load balancer or update an existing load balancer. For more information see How to Create a Load Balancer a HTTPS/SSL Load Balancer (p. 42).

To update an existing SSL certificate, use IAM to upload your new SSL certificate. After you upload the new certficate, update your load balancer with the new certificate. For more information, see How to Update an SSL Certificate for a Load Balancer (p. 87).

Using SSL Cipher Settings with Elastic Load Balancing

If you choose HTTPS/SSL for your front-end connection, you can either use the pre-defined SSL cipher set or use a cipher set of your choice to enable or disable the ciphers based on your specific requirements.

The Secure Sockets Layer (SSL) protocol uses a combination of protocols and algorithms to protect your information over the internet. A SSL cipher is an encryption algorithm that uses encryption keys to create a ciphered (coded) message. There are multiple forms of SSL cipher algorithms available.

Elastic Load Balancing configures your load balancer with a pre-defined cipher set that is used for SSL negotiation when a connection is established between a client and your load balancer. The pre-defined cipher set provides compatibility with a broad range of clients and uses high strength cryptographic algorithms. However, in some use cases there might be a requirement for all data on the network to be encrypted and for allowing only specific ciphers. Some use cases might require specific protocols (such as PCI, SOX, etc.) from clients to ensure that standards are met. In such cases, Elastic Load Balancing provides options for you to select different configurations for SSL protocols and ciphers. You can choose to enable or disable the ciphers depending on your specific requirements.

For information on how to configure the cipher settings, see How to Create a Load Balancer a HTTPS/SSL Load Balancer (p. 42).

Using Back-End Server Authentication with Elastic Load Balancing

If you choose to 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.

Elastic Load Balancing Listener Configurations Quick Reference

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

HTTP/HTTPS Load Balancing

Use Case	Front-End Protocol	Front-End Optional Configuration	Back-End Protocol	Back-End Optional Configuration	Notes
Basic HTTP load balancer.	HTTP	NA	HTTP	NA	Default setting.
Secure website or application using Elastic Load Balancing to offload SSL decryption.	HTTPS	Cipher setting	HTTP	NA	• Supports sticky sessions. • Supports X-Forward headers. • Requires SSL server certificate installed on the load balancer.

Use Case	Front-End Protocol	Front-End Optional Configuration	Back-End Protocol	Back-End Optional Configuration	Notes
Secure website or application using end-to-end encryption with Elastic Load Balancing providing X-Forward headers and sticky sessions.	HTTPS	Cipher setting	HTTPS	Back-end authentication	• Supports sticky sessions. • Supports X-Forward headers. • Requires SSL server certificate installed on the load balancer and on the registered instances.

The following quick reference table shows you the listener configuration options for typical TCP/SSL load balancer use cases.

TCP/SSL Load Balancing

Use Case	Front-End Protocol	Front-End Optional Configuration	Back-End Protocol	Back-End Optional Configuration	Notes
Basic TCP load balancer.	TCP	NA	TCP	NA	Basic TCP load balancing.
Secure website or application using Elastic Load Balancing to offload SSL decryption.	SSL	Cipher setting	TCP	NA	Requires SSL server certificate installed on the load balancer.
Secure website or application using end-to-end encryption with Elastic Load Balancing.	SSL	Cipher setting	SSL	Back-end authentication	Requires SSL server certificate installed on the load balancer.

Where Do I Go From Here?

- To learn how to set up a basic HTTP/TCP load balancer using the AWS Management Console, see Get Started with Elastic Load Balancing (p. 11).
- For information on how to set up an HTTPS/SSL load balancer with cipher settings and back-end authentication, see Creating a Load Balancer With SSL Cipher Settings and Back-end Server Authentication (p. 42).
- For information on how to set up a basic HTTP/TCP load balancer within Amazon VPC, see Deploying Elastic Load Balancing in Amazon VPC (p. 67).

Note

You can use the command line interface or the Query API to modify the settings of an existing load balancer. This functionality is currently not available in the AWS Management Console.

- If you haven't already, install the tools you plan to use for performing Elastic Load Balancing tasks. For information on installing the command line interface or the Query API, see Get Set Up with Elastic Load Balancing Interfaces (p. 20).
- For information on adding a listener to an existing load balancer, see Adding a Listener to your Load Balancer (p. 81). This functionality is currently not avaliable in the AWS Management Console. You'll have to use either the Query API or the command line interface.
- For information on deleting a listener from an existing load balancer, see Delete a Listener from Your Load Balancer (p. 83). This functionality is currently not avaliable in the AWS Management Console. You'll have to use either the Query API or the command line interface.
- For detailed descriptions of the Elastic Load Balancing API operations, see the Elastic Load Balancing API Reference.
- For detailed descriptions of the Elastic Load Balancing commands, see the Elastic Load Balancing Quick Reference Card.

Using Elastic Load Balancing

Elastic Load Balancing makes it easy for you to distribute application load to your Amazon EC2 instances. Load balancing increases the availability and scalability of your applications. This section discusses some common Elastic Load Balancing user scenarios, and it walks you through the various tasks needed to accomplish efficient distribution of application loads among your Amazon EC2 instances.

Before you explore the Elastic Load Balancing scenarios, be sure that you have done the following tasks:

- The examples in this guide assume that your EC2 instances are in the US East (Northern Virginia) Region. If your instances are in Europe, you can specify the EU (Ireland) Region by using the --region eu-west-1 parameter from the command line interface or by setting the AWS_ELB_URL environment variable.

  For more information on using the region parameter with your Elastic Load Balancing commands, go to the Elastic Load Balancing Quick Reference Card. For information on setting your environment variable, see Installing the Command Line Interface (p. 21). For information about this product's Regions and endpoints, go to Regions and Endpoints in the Amazon Web Services General Reference.
- Decided on the listener configurations for your load balancer. For a quick overview of the different configurations available for your load balancer, see Elastic Load Balancing Listener Configurations Quick Reference (p. 36).
- Decided whether you want to load balance your instances launched in Amazon Elastic Compute Cloud (Amazon EC2) or the instances launched within Amazon Virtual Private Cloud (Amazon VPC).

Amazon EC2 provides features such as security groups that can be used for creating and managing the load balancers within Amazon EC2. For information about the using features specific to instances launched within Amazon EC2, see Deploying Elastic Load Balancing in Amazon EC2 (p. 41).

Amazon VPC lets you define a virtual networking environment in a private, isolated section of the Amazon Web Services (AWS) cloud and launch EC2 instances in that environment. Elastic Load Balancing on Amazon VPC works in a similar manner to the way it works in Amazon EC2, and it supports the same set of features. There are however some differences in the procedures for associating your load balancer with the instances. For information on creating and managing load balancers within Amazon VPC, see Deploying Elastic Load Balancing in Amazon VPC (p. 67).

Elastic Load Balancing associates your load balancer with your EC2 instances using IP addresses. When an instance is stopped and then restarted, the IP addresses associated with your instance change. In such cases, we recommend that you de-register your Amazon EC2 instance from your load balancer after you stop your instance, and then register the load balancer with your instance after you've restarted. For detailed instructions on de-registering and registering instances, see De-Registering and Registering Amazon EC2 Instances (p. 85).

Elastic Load Balancing supports the load balancing of applications using HTTP, HTTPS (Secure HTTP), TCP, and SSL (Secure TCP) listener protocols. You can specify the protocols for the front-end connections (client to load balancer) and the back-end connections (load balancer to back-end instance) independently. You choose configurations for the front-end and the back-end connections when you create your load balancer. You can also later add listener to your existing load balancer, or delete a listener from your existing load balancer.

- For information on creating a basic load balancer with HTTP/TCP listener, see Get Started With Elastic Load Balancing (p. 11).
- For information on creating HTTPS/SSL load balancer, see How to Create a Load Balancer a HTTPS/SSL Load Balancer (p. 42).
- For information on adding listener to your existing load balancer, see Adding a Listener to your Load Balancer (p. 81).
- For information on deleting a Listener from your load balancer, see Delete a Listener from Your Load Balancer (p. 83).

If you are using HTTPS/SSL protocol for your listeners, you might have an SSL server certificate installed on your load balancer. You will have to update this certificate periodically. For information on updating your existing SSL certificate, see Updating an SSL Certificate for a Load Balancer (p. 87).

Each Elastic Load Balancing instance that you create has a unique Domain Name System (DNS) name. If you'd prefer to have a custom domain name instead of the load balancer DNS name, you can use Amazon Route 53 to create a custom domain name and associate it with your load balancer DNS name. For more information, see Using Domain Names with Elastic Load Balancing (p. 90).

The load balancer uses a special load-balancer-generated cookie to track the application instance for each request. After the application instance is tracked, the cookie is inserted in the response for binding subsequent requests from the same user to that application instance. Elastic Load Balancing allows you to configure policies to establish the duration of validity of these cookies. For more information about configuring policies for load-balancer-generated cookies, see Enabling Duration-Based Session Stickiness (p. 95).

The load balancer uses a special cookie to associate the session with the original server that handled the request, but follows the lifetime of the application-generated cookie corresponding to the cookie name specified in the policy configuration. Elastic Load Balancing allows you to configure policies for application-generated cookies if the application cookie is explicitly removed or expires. For more information about configuring policies for application-generated cookies, see Enabling Application-Controlled Session Stickiness (p. 97).

Elastic Load Balancing provides data about your load balancers and your back-end application instances to Amazon CloudWatch. Amazon CloudWatch collects the data and presents it as readable, near real-time metrics. For more information on viewing and interpreting your load balancer metrics, see Monitoring Your Load Balancer Using CloudWatch (p. 99).

For information on deleting an existing load balancer, see Deleting Your Load Balancer (p. 102).

Deploying Elastic Load Balancing in Amazon EC2

Amazon Web Services (AWS) lets you launch your instances in either Amazon EC2 or Amazon Virtual Private Cloud (Amazon VPC). If you've launched your instances in Amazon EC2, you will have to create a load balancer in Amazon EC2. If you've launched your instances with Amazon VPC, you will have to create your load balancer within the VPC. Elastic Load Balancing on Amazon VPC works mostly in a similar manner as in Amazon EC2 and supports the same set of features. There is however a significant difference between the procedures involved in launching a load balancer and the way security groups function on Amazon VPC and Amazon EC2.

This section walks you through the process of creating, accessing, and managing your load balancers in Amazon EC2. For information on creating, accessing, and managing your load balancers in Amazon VPC, see Deploying Elastic Load Balancing in Amazon VPC (p. 67).

Elastic Load Balancing provides SSL support for connections between clients and the load balancer and also for connections between the load balancer and your back-end application instances. If you choose HTTPS/SSL for your front-end connection, you can either use the pre-defined cipher set or use a cipher set of your choice to enable or disable the ciphers based on your specific requirements. If you choose to use an HTTPS/SSL connection for your back end, you can enable authentication on your back-end instance. For more information on configuring your load balancer to use cipher settings and for enabling authentication on back-end instance, see Choosing Listeners for Your Load Balancer (p. 34). And for detailed instructions on creating a load balancer in Amazon EC2 with pre-defined cipher settings and enabling the back-end authentication, see Creating a Load Balancer With SSL Cipher Settings and Back-end Server Authentication (p. 42).

As the traffic to your instances increases, you might consider expanding your EC2 instances to run in an additional Availability Zone (`us-east-1b`). For detailed instructions on expanding the Availability Zone, see Expanding a Load Balanced Application to an Additional Availability Zone (p. 59).

And when the trafic to your instances decreases, you might consider scaling down the availability of your instances by disabling some Availibility Zones. For detailed instructions on disabling your Availability Zones, see Disabling an Availability Zone from a Load-Balanced Application (p. 62).

Elastic Load Balancing provides a special Amazon EC2 source security group that you can use to ensure that a back-end Amazon EC2 instance receives traffic only from Elastic Load Balancing. For more information on using, managing, and creating security groups for your Amazon EC2 instances, see Managing Security Groups in Amazon EC2 (p. 63).

After you create your load balancer, Elastic Load Balancing returns a public DNS name that combines your load balancer's name and Region. This base public DNS name returns only IPv4 records. Elastic Load Balancing supports both Internet Protocol version 6 (IPv6) and Internet Protocol version 4 (IPv4). Clients can connect to your load balancer using either IPv4 or IPv6. However, communication between the load balancer and its back-end instances uses only IPv4. You might want to use the dualstack-prefixed DNS name to enable IPv6 support for communications between client and the load balancers so that clients are able to access the load balancer using either IPv4 or IPv6 as their individual connectivity needs dictate. For more information on enabling IPv6 support, see Using IPv6 with Elastic Load Balancing (p. 65).

Note

IPv6 support is currently not available in all regions. For current IPv6 support, go to Elastic Load Balancing.

Note

IPv6 support is not currently available for load balancers in Amazon VPC.

Creating a Load Balancer With SSL Cipher Settings and Back-end Server Authentication

This example walks you through the process of creating your own load balancer with custom settings. The following task list describes the process of creating a load balancer.

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

- Sign up for Amazon Web Services (AWS). If you haven't signed up for AWS yet, go to http://aws.amazon.com and click the **Sign Up Now** button.
- Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
  Alternatively, you can create a load balancer using the command line interfaces or the Query API. Install the tools you'll need to perform Elastic Load Balancing tasks. For information on installing the command line interfaces and the Query API, see Get Set Up with Elastic Load Balancing Interfaces (p. 20).
- Download and install the AWS Identity and Access Management command line interfaces. For more information, go to Get the Command Line Tools in the *AWS Identity and Access Management Getting Started Guide*.
- In Availability Zone us-east-1a, launch the instances you intend to register with your load balancer. For more information about launching Amazon EC2 instances, see Launching and Using Instances.
- Elastic Load Balancer maintains a 60 second timeout setting for idle connections to back-end application servers. Update these settings on your back-end server to a timeout of at least 60 seconds for the communication to work properly.
- The instances to be registered with your load balancer must respond to the target of the health check with an HTTP status code 200.
- Create a signed certificate. For information on how to create a signed certificate, go to Creating and Uploading Server Certificates in *Using AWS Identity and Access Management*.

Tasks for Creating a Load Balancer with SSL Cipher Settings and Back-end Server Authentication

1	Configure the listeners for your load balancer by specifying the ports and protocols to use for front-end connection (client to load balancer) and back-end connection (load balancer to back-end instance).
2	Configure SSL ciphers for SSL negotiation when a connection is established between the client and your load balancer.
3	[Optional] Enable the back-end server authentication.
4	Configure an application health check for your back-end instances.
5	Add Amazon EC2 instances to your load balancer.
6	Launch your load balancer.

The following sections include instructions for creating a load balancer using the AWS Management Console, command line interfaces, or the Query API.

Using AWS Management Console

Topics

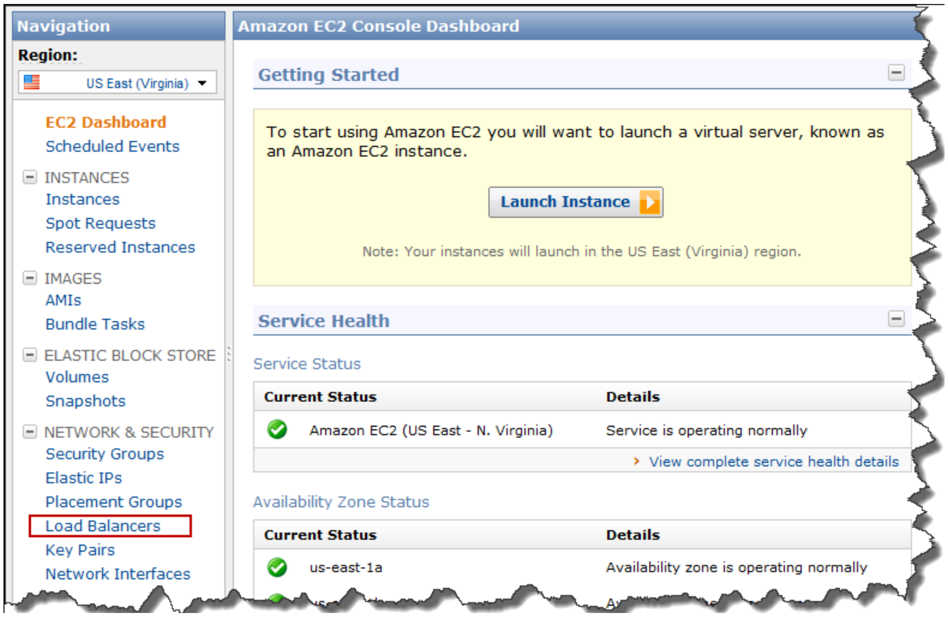
Configuring Listeners

Configure the listeners for your load balancer by specifying the ports and protocols to use for front-end connection (client to load balancer) and back-end connection (load balancer to back-end instance). The first listener accepts HTTP requests on port 80 and sends the request to the back-end application instances on port 8080 using HTTP. The second listener accepts HTTPS requests on port 443 and sends the request to back-end application instances using HTTPS on port 443.

To configure listeners for your load balancer

1. Start the **Create Load Balancer** wizard:

   a. In the Amazon EC2 Console Dashboard page, click **Load Balancers** in the **Navigation** pane.



To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

Note: Your instances will launch in the US East (Virginia) region.

Current Status	Details
✓ Amazon EC2 (US East - N. Virginia)	Service is operating normally

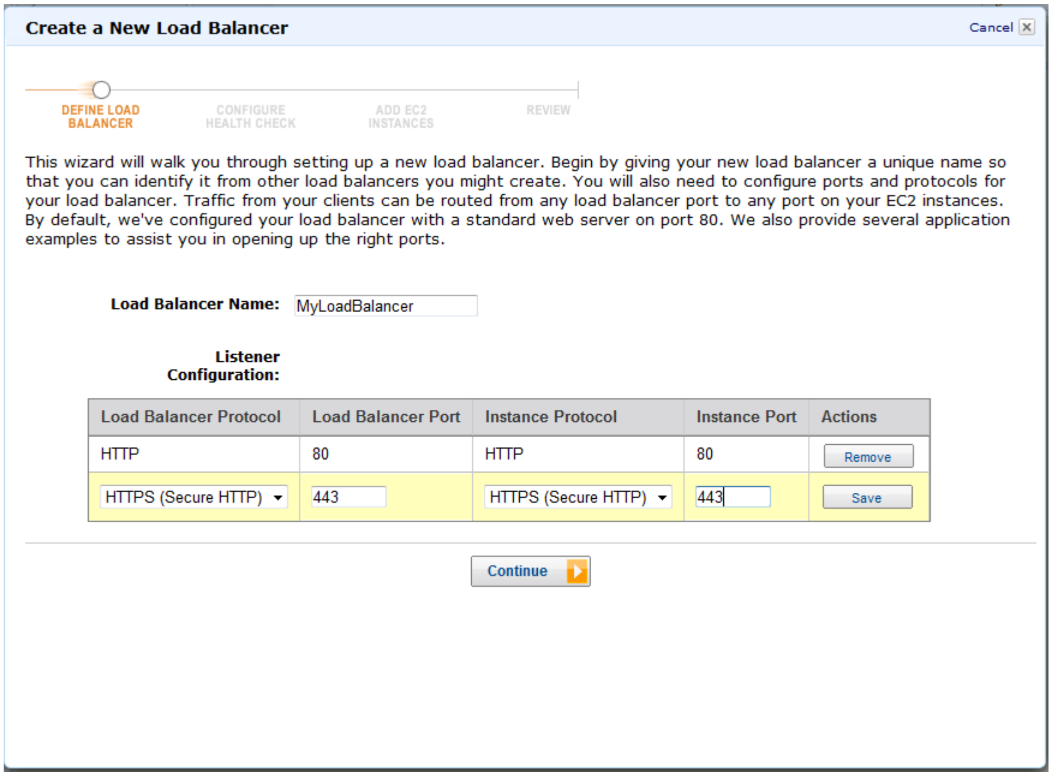
Current Status	Details
✓ us-east-1a	Availability zone is operating normally

   b. On the **Load Balancers** page, click **Create Load Balancer**.

   c. The **DEFINE LOAD BALANCER** page of the **Create a New Load Balancer** wizard opens.

2. On the **DEFINE LOAD BALANCER** page, enter a name for your load balancer (e.g., MyLoadBalancer).
3. Leave the **Listener Configuration** set to the default value for the first listener.
4. Select **HTTPS (Secure HTTP)** from the drop-down box in the **Load Balancer Protocol** box. This populates the **Load Balancer Port** box. Select **HTTPS (Secure HTTP)** from the drop-down box in

the **Instance Protocol** box, then enter port number 443 for the instance port in the **Instance Port** box.



This wizard will walk you through setting up a new load balancer. Begin by giving your new load balancer a unique name so that you can identify it from other load balancers you might create. You will also need to configure ports and protocols for your load balancer. Traffic from your clients can be routed from any load balancer port to any port on your EC2 instances. By default, we've configured your load balancer with a standard web server on port 80. We also provide several application examples to assist you in opening up the right ports.

Load Balancer Name: MyLoadBalancer

Listener Configuration:

Load Balancer Protocol	Load Balancer Port	Instance Protocol	Instance Port	Actions
HTTP	80	HTTP	80	Remove
HTTPS (Secure HTTP)	443	HTTPS (Secure HTTP)	443	Save

5. Click **Save**, then click **Continue** to upload your SSL certificate.
6. Select **Choose from your existing SSL Certificates** to use the previously uploaded SSL certificate and select the certificate from the drop-down box.
7. Or, select **Upload a new SSL Certificate** to define a new SSL certificate.

   a. Enter the name of the certificate to upload.

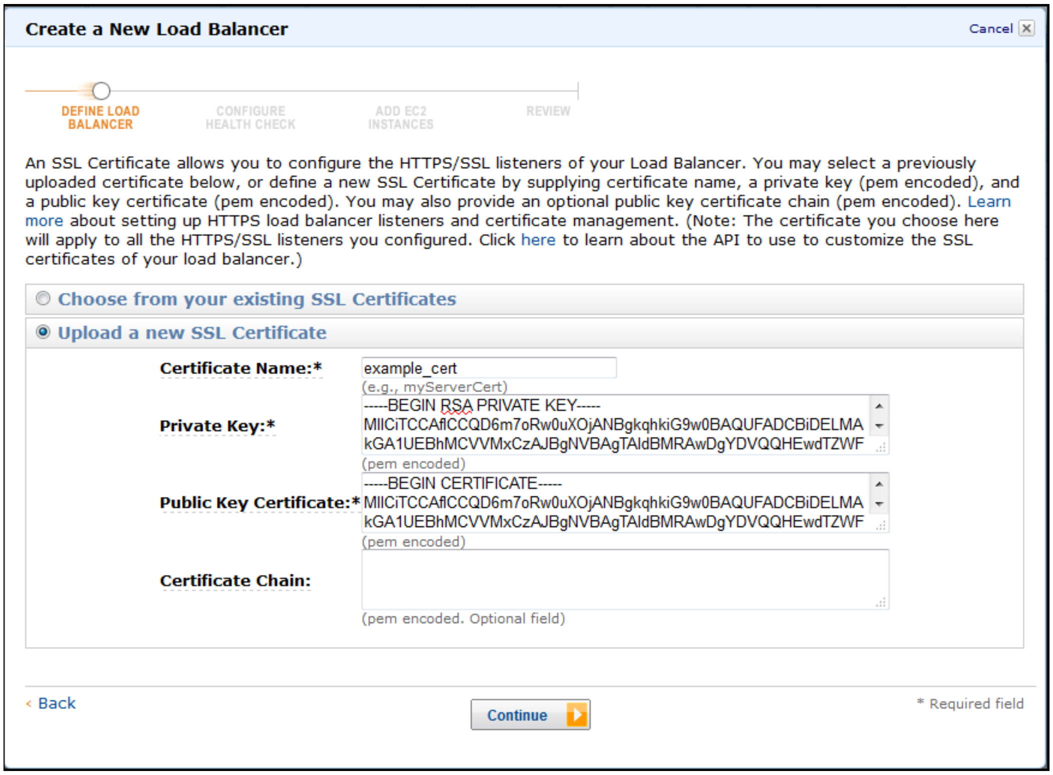
   b. Copy and paste the contents of the private key file (PEM-encoded) in the **Private Key** box.

   c. Copy and paste the contents of the public key certificate file (PEM-encoded) in the **Public Key Certificate** box.

   d. [Optional] Copy and paste the contents of the public key certificate chain file (PEM-encoded) in the **Certificate Chain** box.

   ### Note

   The certificate chain must be ordered such that the root certificate is the last certificate in the chain. If you use a certificate chain in a different order, you will receive an error.



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 by supplying certificate name, a private key (pem encoded), and a public key certificate (pem encoded). You may also provide an optional public key certificate chain (pem encoded). Learn more about setting up HTTPS load balancer listeners and certificate management. (Note: The certificate you choose here will apply to all the HTTPS/SSL listeners you configured. Click here to learn about the API to use to customize the SSL certificates of your load balancer.)

○ Choose from your existing SSL Certificates

● Upload a new SSL Certificate

Certificate Name:* example_cert
(e.g., myServerCert)

Private Key:*
-----BEGIN RSA PRIVATE KEY-----
MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMA
kGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF
(pem encoded)

Public Key Certificate:*
-----BEGIN CERTIFICATE-----
MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMA
kGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF
(pem encoded)

Certificate Chain:
(pem encoded. Optional field)

8. Click **Continue** to configure SSL ciphers for the HTTPS/SSL listeners.

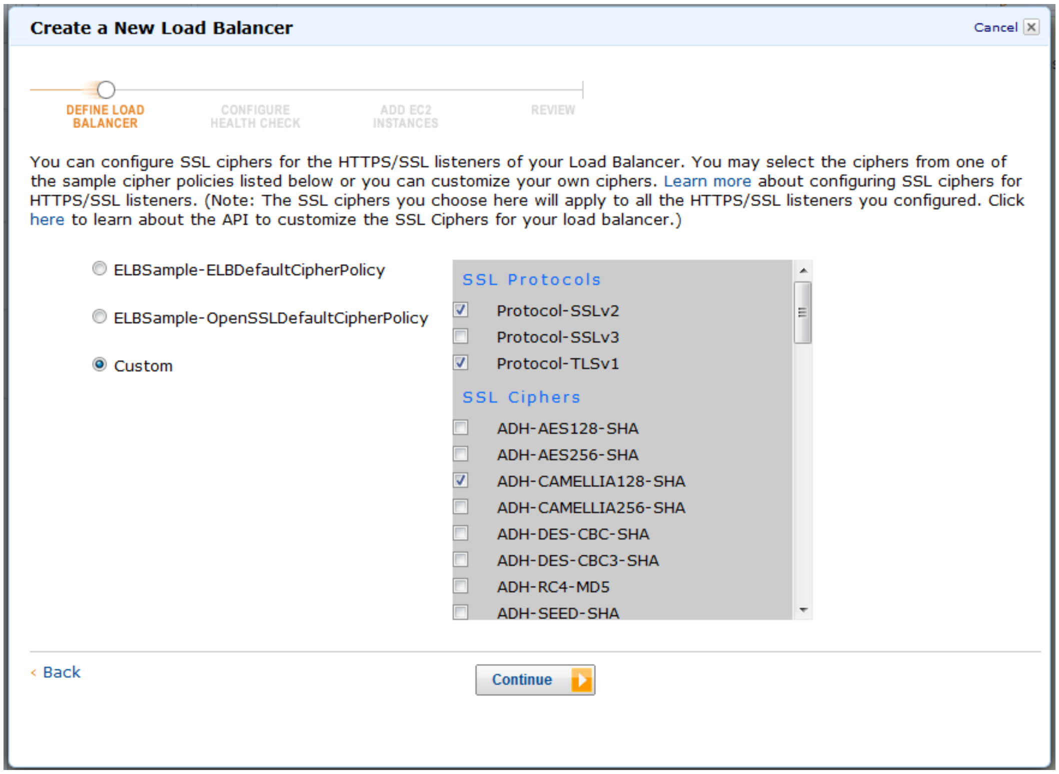
Configuring SSL Ciphers

Next the wizard takes you through the steps for configuring SSL ciphers for your HTTPS/SSL listeners. The Elastic Load Balancing service provides you with sample cipher policies, **ELBSample-ELBDefaultCipherPolicy** and **ELBSample-OpenSSLDefaultCipherPolicy**. You can select one of the sample policies or customize your own ciphers.

1. To customize the SSL ciphers, select **Custom** on the **DEFINE LOAD BALANCER** page, then select the protocol version and the ciphers from the list box.

   Note

   You must enable at least one protocol version and one cipher for SSL negotiation to take place.



You can configure SSL ciphers for the HTTPS/SSL listeners of your Load Balancer. You may select the ciphers from one of the sample cipher policies listed below or you can customize your own ciphers. Learn more about configuring SSL ciphers for HTTPS/SSL listeners. (Note: The SSL ciphers you choose here will apply to all the HTTPS/SSL listeners you configured. Click here to learn about the API to customize the SSL Ciphers for your load balancer.)

○ ELBSample-ELBDefaultCipherPolicy

○ ELBSample-OpenSSLDefaultCipherPolicy

● Custom

SSL Protocols

☑ Protocol-SSLv2
☐ Protocol-SSLv3
☑ Protocol-TLSv1

SSL Ciphers

☐ ADH-AES128-SHA
☐ ADH-AES256-SHA
☑ ADH-CAMELLIA128-SHA
☐ ADH-CAMELLIA256-SHA
☐ ADH-DES-CBC-SHA
☐ ADH-DES-CBC3-SHA
☐ ADH-RC4-MD5
☐ ADH-SEED-SHA

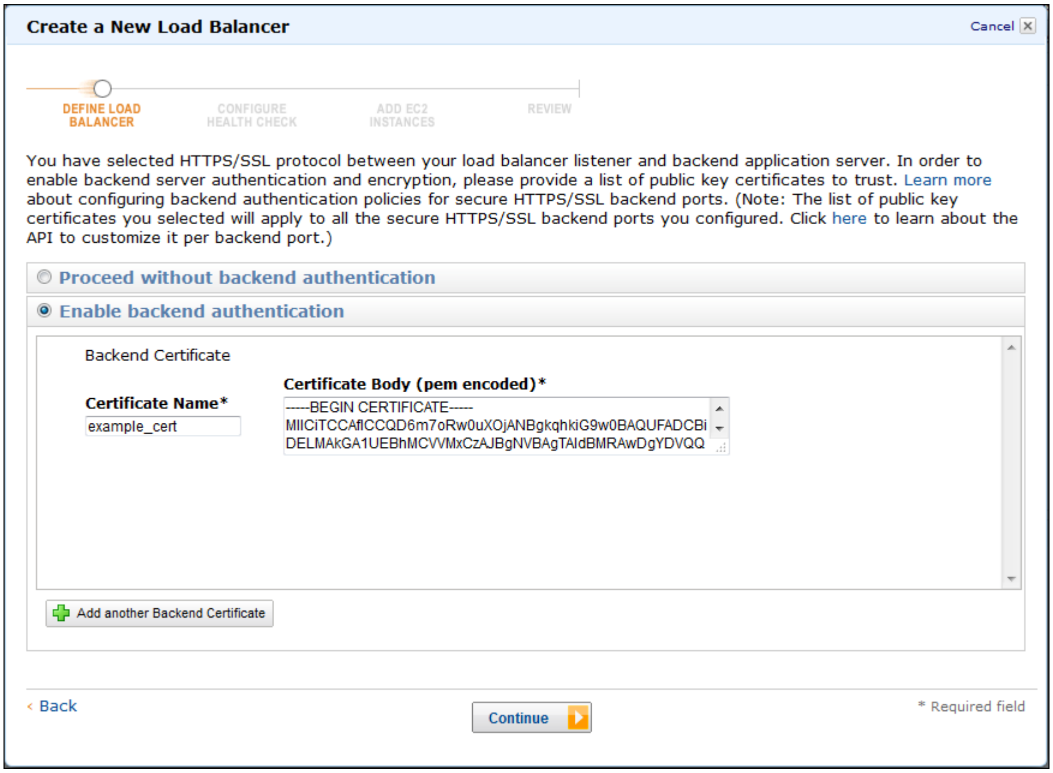
2. Click **Continue** to configure back-end server authentication.

Configuring Back-end Server Authentication

Next the wizard gives you an option to enable authentication for your back-end server if you have selected HTTPS/SSL protocol between your load balancer and the back-end instance.

1. Select **Proceed without backend authentication** if you do not want to enable authentication for your back-end server
2. Or, select **Enable backend authentication** to enable back-end server authentication.

   a. Enter the name of the public key certificate in the **Certificate Name** box, and then copy and paste the contents of the certificate (PEM-encoded) in the **Certificate body** box.



Create a New Load Balancer

You have selected HTTPS/SSL protocol between your load balancer listener and backend application server. In order to enable backend server authentication and encryption, please provide a list of public key certificates to trust. Learn more about configuring backend authentication policies for secure HTTPS/SSL backend ports. (Note: The list of public key certificates you selected will apply to all the secure HTTPS/SSL backend ports you configured. Click here to learn about the API to customize it per backend port.)

○ Proceed without backend authentication

● Enable backend authentication

Backend Certificate

Certificate Name*: example_cert

Certificate Body (pem encoded)*

```
-----BEGIN CERTIFICATE-----
MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBi
DELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQ
```

b. Click **Add another Backend Certificate** to add multiple certificates.

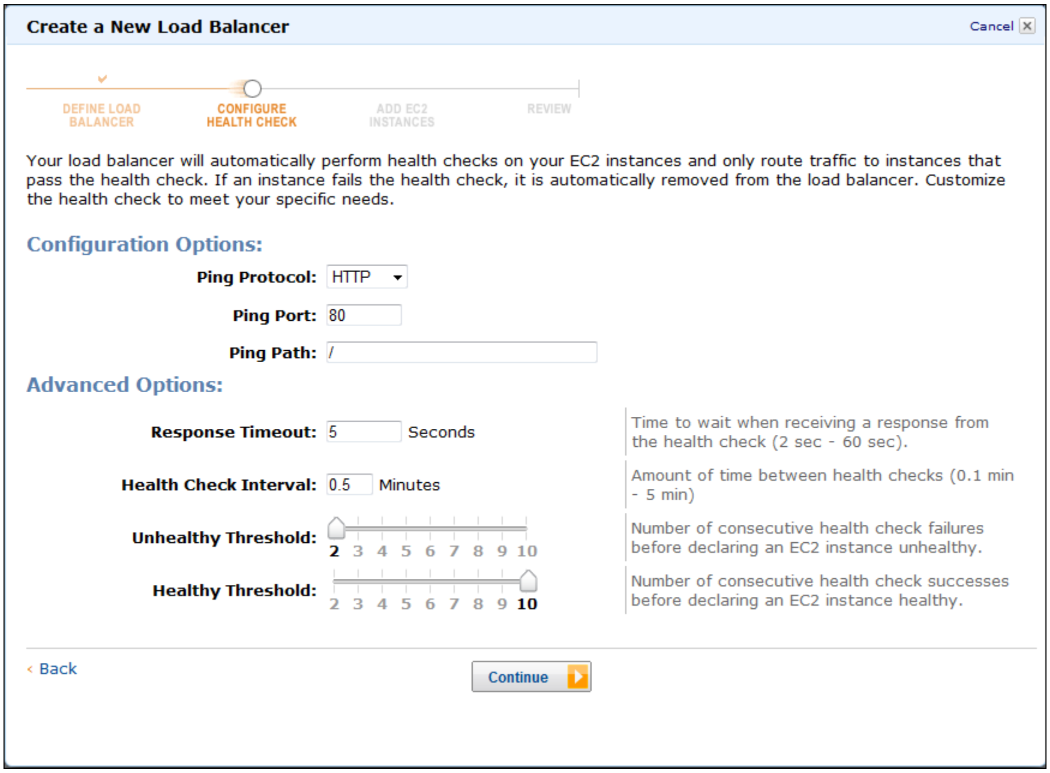
3. Click **Continue** to configure health check for your back-end server.

Configuring Health Check Settings

Next the wizard takes you through the steps for configuring a health check for your back-end instances.

To configure the health check

1. Configure the health check settings that your application requires.



Your load balancer will automatically perform health checks on your EC2 instances and only route traffic to instances that pass the health check. If an instance fails the health check, it is automatically removed from the load balancer. Customize the health check to meet your specific needs.

Configuration Options:

Ping Protocol: HTTP

Ping Port: 80

Ping Path: /

Advanced Options:

Response Timeout: 5 Seconds — Time to wait when receiving a response from the health check (2 sec - 60 sec).

Health Check Interval: 0.5 Minutes — Amount of time between health checks (0.1 min - 5 min)

Unhealthy Threshold: 2 (2 3 4 5 6 7 8 9 10) — Number of consecutive health check failures before declaring an EC2 instance unhealthy.

Healthy Threshold: 10 (2 3 4 5 6 7 8 9 10) — Number of consecutive health check successes before declaring an EC2 instance healthy.

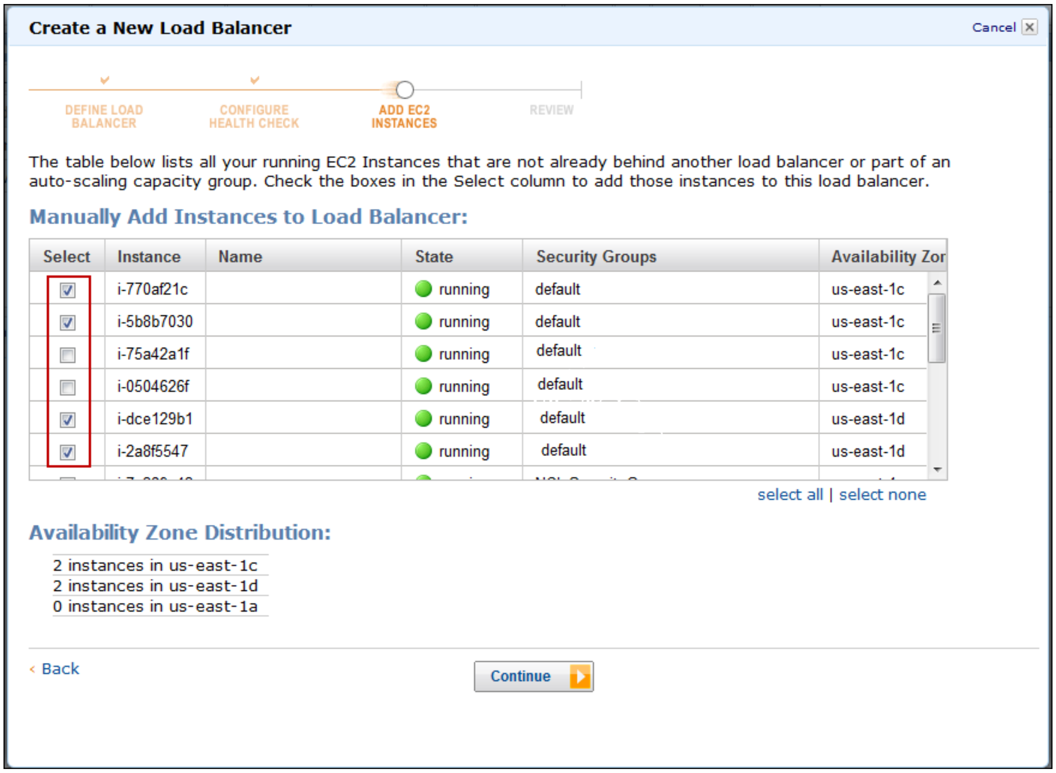
2. Click **Continue** to add your Amazon EC2 instances.

Adding Amazon EC2 Instances

Next the wizard takes through the steps for adding Amazon EC2 instances to your load balancer.

To add Amazon EC2 instances

1. Check the boxes in the **Select** column to add instances to your load balancer.



Create a New Load Balancer

The table below lists all your running EC2 Instances that are not already behind another load balancer or part of an auto-scaling capacity group. Check the boxes in the Select column to add those instances to this load balancer.

Manually Add Instances to Load Balancer:

Select	Instance	Name	State	Security Groups	Availability Zor
☑	i-770af21c		● running	default	us-east-1c
☑	i-5b8b7030		● running	default	us-east-1c
□	i-75a42a1f		● running	default	us-east-1c
□	i-0504626f		● running	default	us-east-1c
☑	i-dce129b1		● running	default	us-east-1d
☑	i-2a8f5547		● running	default	us-east-1d

Availability Zone Distribution:

2 instances in us-east-1c
2 instances in us-east-1d
0 instances in us-east-1a

2. Click **Continue** to review your configuration. On the Review page, click **Create** to create your load balancer.

> **Important**
>
> Elastic Load Balancing associates your load balancer with your EC2 instance using the IP addresses. When the instance is stopped and then restarted, the IP addresses associated with your instance changes. Your load balancer cannot recognize the new IP address, which prevents it from routing traffic to your instances. We recommend that you de-register your Amazon EC2 instances from your load balancer after you stop your instance, and then register the load balancer with your instance after you've restarted. For procedures associated with de-registering and then registering your instances with load balancer, see De-Registering and Registering Amazon EC2 Instances (p. 85).

Using Query API

Topics

Configuring Listeners

In this example, you configure the listeners for your load balancer by specifying the ports and protocols to use for front-end connection (client to load balancer) and back-end connection (load balancer to back-end instance). The first listener accepts HTTP requests on port 80 and sends the request to the back-end application instances on port 8080 using HTTP. The second listener accepts HTTPS requests on port 443 and sends the request to back-end application instances using HTTPS on port 443. You also need to specify the Availability Zone that you want to enable for your load balancer.

For detailed descriptions of the Elastic Load Balancing API actions, see Elastic Load Balancing API Reference.

To configure listeners for your load balancer

1. Call the AWS Identity and Access Management UploadServerCertificate action with the following parameters:

   - *ServerCertificateName* = testCert
   - *CertificateBody* = *<encoded certificate body>*
   - *PrivateKey* = *<encoded private key>*
   - *CertificateChain* = *<concatenation of the encoded public key certificates>*

     Note

     CertificateChain is optional. If you are using *CertificateChain*, then you must order the certificates such that the root certificate is the last certificate in the chain. If you use a certificate chain in a different order, you will receive an error.

   - *Path* = /

     Note

     Path is optional. If it is not included, the path defaults to /. For more information about paths, go to Identifiers for IAM Entities in *Using AWS Identity and Access Management*.

   The response includes the ARN of the server certificate. Use this value for the *SSLCertificateId* parameter in the following call to CreateLoadBalancer.
2. Call CreateLoadBalancer with the following parameters:

   - *AvailabilityZones* = us-east-1a
   - *Listener*
     - *Protocol* = HTTP
     - *InstanceProtocol* = HTTP
     - *InstancePort* = 8080
     - *LoadBalancerPort* = 80
   - *Listener*
     - *Protocol* = HTTPS
     - *InstanceProtocol* = HTTPS
     - *InstancePort* = 443
     - *LoadBalancerPort* = 443

  - *SSLCertificateID* = arn:aws:iam::55555555555:server-certificate/production/myCert
- *LoadBalancerName* = MyLoadBalancer

The operation returns the DNS name of your load balancer. You can then map any other domain name (such as www.example.com) to your load balancer's DNS name using CNAME or some other technique.

Configuring SSL Ciphers

In this example, you create an SSL cipher policy to configure SSL ciphers for SSL negotiation when a connection is established between the client and your load balancer. The Elastic Load Balancing service defines a policy called `SSLNegotiationPolicyType`. You create your own SSL cipher policy `MySSLNegotiationPolicy` of the type `SSLNegotiationPolicyType`. After creating the SSL cipher policy, you enable the cipher settings by associating `MySSLNegotiationPolicy` with a listener.

To configure SSL Ciphers

1. List all the policies associated with your load balancer by calling `DescribeLoadBalancerPolicies` with the following parameter:

   - *LoadBalancerName* = MyLoadBalancer

   The response includes the policy names and the attributes of all the policies associated with your load balancer. The attributes associated with `SSLNegotiationPolicyType` list the default cipher settings for your load balancer. Use the attributes in the following call to `CreateLoadBalancerPolicy` to configure your own cipher settings.

   Note

   For more information on the available ciphers, go to http://www.openssl.org/docs/apps/ciphers.html.

2. Call `CreateLoadBalancerPolicy` with the following parameters:

   - *PolicyName* = MySSLNegotiationPolicy
   - *PolicyTypeName* = SSLNegotiationPolicyType
   - *PolicyAttributes*
     - *AttributeName* = Protocol-TLSv1
     - *AttributeValue* = true
   - *LoadBalancerName* = MyLoadBalancer

3. Call `SetLoadBalancerPoliciesOfListener` with the following parameters:

   - *LoadBalancerPort* = 443
   - *PolicyNames* = MySSLNegotiationPolicy
   - *LoadBalancerName* = MyLoadBalancer

4. View the details of `MySSLNegotiationPolicy` by calling `DescribeLoadBalancerPolicies` with the following parameters:

- *LoadBalancerName* = `MyLoadBalancer`
- *PolicyNames* = `MySSLNegotiationPolicy`

Configuring Back-end Server Authentication

In this example, you enable back-end server authentication. First you create a public key policy that uses a public key for authentication. You then use the public key policy to create a back-end server authentication policy. Finally, you enable the backend 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 with SSL/HTTPS protocol set to instance port 443.

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

Note

To extract the public key from a pem-encoded certificate, you can use the following command:

```
PROMPT> openssl x509 -inform pem -in <CERTIFICATE_FILE_PATH> -noout -
pubkey)
```

Remove the BEGIN and END lines from the output so that the output is similar to that described below.

To configure back-end server authentication

1. Call `CreateLoadBalancerPolicy` with the following parameters:

   - *PolicyName* = `MyPublicKeyPolicy`
   - *PolicyTypeName* = `PublicKeyPolicyType`
   - *PolicyAttributes*
     - *AttributeName* = `PublicKey`
     - *AttributeValue* =

```

MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD
VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z
b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
rDHudUZg3qX4waLG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
```

   - *LoadBalancerName* = `MyLoadBalancer`

2. Call `CreateLoadBalancerPolicy` with the following parameters:
   - *`PolicyName`* = `MyBackendServerAuthenticationPolicy`
   - *`PolicyTypeName`* = `BackendServerAuthenticationPolicyType`
   - *`PolicyAttributes`*
     - *`AttributeName`* = `PublicKeyPolicyName`
     - *`AttributeValue`* = `MyPublicKeyPolicy`
   - *`LoadBalancerName`* = `MyLoadBalancer`

3. Call `SetLoadBalancerPoliciesForBackendServer` with the following parameters:
   - *`LoadBalancerName`* = `MyLoadBalancer`
   - *`InstancePort`* = `443`
   - *`PolicyNames`* = `MyBackendServerAuthenticationPolicy`

4. To list all the policies associated with your load balancer,
   call `DescribeLoadBalancerPolicies` with the following parameters:
   - *`LoadBalancerName`* = `MyLoadBalancer`

5. To view the details of `MyBackendServerAuthenticationPolicy`,
   call `DescribeLoadBalancerPolicies` with the following parameters:
   - *`LoadBalancerName`* = `MyLoadBalancer`
   - *`PolicyNames`* = `MyBackendServerAuthenticationPolicy`

Configuring Health Check Settings

In this example, you configure the health check settings for your back-end servers.

To configure health check settings

- Call `ConfigureHealthCheck` with the following parameters:
  - *`LoadBalancerName`* = `MyLoadBalancer`
  - *`Target`* = `http:8080/ping`

    Note

    Make sure your instances respond to ping on port 8080 with an HTTP 200 status code.
  - *`Interval`* = `30`
  - *`Timeout`* = `3`
  - *`HealthyThreshold`* = `2`
  - *`UnhealthyThreshold`* = `2`

Adding Amazon EC2 Instances

In this example, you register your newly created load balancer with your Amazon EC2 instances.

Important

You should only register instances that are in the *Pending* or *Running* state and are not in a Virtual Private Cloud(VPC). If you are using Elastic Load Balancing in a VPC, see Deploying Elastic Load Balancing in Amazon VPC (p. 67)

To add Amazon EC2 instances

- Call `RegisterInstancesWithLoadBalancer` with the following parameters:
  - *`LoadBalancerName`* = `MyLoadBalancer`
  - *`Instances`* = [`i-4f8cf126`, `i-0bb7ca62`]

Note

To allow communication between Elastic Load Balancing and your back-end instances, create a security group ingress rule that applies to all of your back-end instances. The security group rule can either allow ingress traffic from all IP addresses (the 0.0.0.0/0 CIDR range) or allow ingress traffic only from Elastic Load Balancing. To ensure that your back-end EC2 instances can receive traffic only from Elastic Load Balancing, enable network ingress for the Elastic Load Balancing security group on all of your back-end EC2 instances. For more information, see Managing Security Groups in Amazon EC2 (p. 63).

Important

Elastic Load Balancing registers your load balancer with the instance using the IP addresses. When the instance is stopped and then restarted, the IP addresses associated with your instance changes. Your load balancer cannot recognize the new IP address, which prevents it from routing traffic to your instances. We recommend you de-register your Amazon EC2 instances from your load balancer after you stop your instance, and then register the new instance ID with the load balancer after you restart your instance. For procedures associated with de-registering and then registering your instances with load balancer, see De-Registering and Registering Amazon EC2 Instances (p. 85).

Using the Command Line Interface

Topics

Configuring Listeners

In this example, you configure the listeners for your load balancer by specifying the ports and protocols to use for front-end connection (client to load balancer) and back-end connection (load balancer to back-end instance). The first listener accepts HTTP requests on port 80 and sends the request to the back-end application instances on port 8080 using HTTP. The second listener accepts HTTPS requests on port 443 and sends the request to back-end application instances using HTTPS on port 443. You also need to specify the Availability Zone that you want to enable for your load balancer.

For descriptions of all the Elastic Load Balancing commands, see Elastic Load Balancing Quick Reference Card.

To configure listeners for your load balancer

1. Enter the command `iam-servercertupload` in verbose mode to upload your digitally signed certificate to the AWS IAM service.

   Note

   For information on how to create a signed certificate, go to Creating and Uploading Server Certificates in *Using AWS Identity and Access Management*.

   ```
   PROMPT> iam-servercertupload -b <encoded certificate body> -k <encoded
   private key> -s myCert [-c <concatenation of the encoded public key certi
   ficates>] -v
   ```

   Note

   -c is optional. If you are using *-c*, then you must order the certificates such that the root certificate is the last certificate in the chain. If you use a certificate chain in a different order, you will receive an error.

   The response includes the server certificate Amazon Resource Name (ARN) and GUID.

   ```
   arn:aws:iam::55555555555:server-certificate/production/myCert
   ASCACexampleKEZUQ4K
   ```

2. Copy the ARN for the next step.
3. Enter the command `elb-create-lb` as in the following example.

   ```
   PROMPT> elb-create-lb  MyLoadBalancer --headers --listener "lb-port=80,in
   stance-port=8080,protocol=http,instance-protocol=http"
     --listener "lb-port=443,instance-port=443,protocol=https,instance-pro
   tocol=https, cert-id=arn:aws:iam::55555555555:server-certificate/produc
   tion/myCert" --availability-zones us-east-1a
   ```

   Elastic Load Balancing returns the following:

   ```
   DNS-NAME  DNS-NAME
   DNS-NAME  MyLoadBalancer-2111276808.us-east-1a.elb.amazonaws.com
   ```

Configuring SSL Ciphers

When you first create your ELB, it is created with a default set of SSL ciphers and protocols. You can create overrides to this default by specifying your own cipher policy.

In this example, you create an SSL cipher policy to configure SSL ciphers for SSL negotiation when a connection is established between the client and your load balancer. The Elastic Load Balancing service defines a policy called `SSLNegotiationPolicyType`. You create your own SSL cipher policy `MySSLNegotiationPolicy` of the type `SSLNegotiationPolicyType`. After creating the SSL cipher policy, you enable the cipher settings by associating `MySSLNegotiationPolicy` with a listener.

To configure SSL ciphers

1. Enter the command `elb-describe-lb-policies`, as in the following example, to list all the policies associated with `MyLoadBalancer`.

   ```
   PROMPT>elb-describe-lb-policies MyLoadBalancer --headers
   ```

   Elastic Load Balancing returns the following:

   ```
   POLICY  NAME                      TYPE_NAME
   POLICY  MyAppStickinessPolicy     AppCookieStickinessPolicyType
   POLICY  MyLBStickinessPolicy      LBCookieStickinessPolicyType
   POLICY  MySSLNegotiationPolicy    SSLNegotiationPolicyType
   ```

   The response includes the policy names of all the policies associated with your load balancer. We will be using `SSLNegotiationPolicyType` to create a new policy by changing the pre-defined cipher settings. For more information on all the available ciphers, go to http://www.openssl.org/docs/apps/ciphers.html.
2. Enter the command `elb-describe-lb-policy-types`, as in the following example to retrieve a list of available ciphers and policies associated with `SSLNegotiationPolicyType`.

   ```
   PROMPT>elb-describe-lb-policy-types SSLNegotiationPolicyType --show-long
   ```

   We will be changing the cipher settings and the protocols associated with `SSLNegotiationPolicyType` to create `MySSLNegotiationPolicy`.
3. Enter the command `elb-create-lb-policy`, as in the following example, to create a new policy for your load balancer that accepts TLSv1 protocol, does not accept SSLv2 protocol, and accepts the cipher DHE-RSA-AES256-SHA. Protocol SSLv3 is still enabled, because that is part of the default policy.

   ```
   PROMPT>elb-create-lb-policy MyLoadBalancer --policy-name MySSLNegotiation
   Policy --policy-type SSLNegotiationPolicyType --attribute "name=Protocol-
   TLSv1,value=true" --attribute "name=Protocol-SSLv2,value=false  --attribute
    "name=DHE-RSA-AES256-SHA,value=true"
   ```

4. Enter the command `elb-set-lb-policies-of-listener`, as in the following example, to enable the cipher settings by setting the `MySSLNegotiationPolicy` with a listener.

   ```
   PROMPT>elb-set-lb-policies-of-listener MyLoadBalancer --lb-port 443 --policy-
   name MySSLNegotiationPolicy
   ```

5. Enter the command `elb-describe-lb-policies`, as in the following example, to view details of `MySSLNegotiationPolicy`.

   ```
   PROMPT>elb-describe-lb-policies MyLoadBalancer --policy-names MySSLNegoti
   ationPolicy
   ```

   Following is the partial listing of the example response:

   ```
   POLICY,NAME,TYPE_NAME,POLICY_ATTRIBUTE_DESCRIPTIONS
   POLICY,MySSLNegotiationPolicy,SSLNegotiationPolicyType,"{name=Protocol-
   SSLv2,value=true},{name=EDH-DSS-DES-CBC3-SHA,value=false},{name=DHE-RSA-
   CAMELLIA128-SHA,value=false},{name=DES-CBC-MD5,value=false},{name=KRB5-RC4-
   SHA,value=false},{name=ADH-CAMELLIA128-SHA,value=false},{name=EXP-KRB5-RC4-
   MD5,value=false}
   ```

Configuring Back-end Server Authentication

In this example, you enable the back-end server authentication by creating a public key policy that uses a public key for authentication. You then use the 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 with SSL/HTTPS protocol set to instance port 443.

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

To configure back-end server authentication

1. Enter the command `openssl x509` to retrieve the public key.

   ```
   openssl x509 -in PublicKey -pubkey -noout
   ```

2. Enter the command `elb-create-lb-policy`, as in the following example, to create a public key policy.

   ```
   PROMPT>elb-create-lb-policy MyLoadBalancer --policy-name MyPublicKeyPolicy
    --policy-type PublicKeyPolicyType --attribute "name=PublicKey,value=
   ```

```
MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD
VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z
b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
rDHudUZg3qX4waLG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
```

```
FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=

"
```

Note

To specify a public key value for the *`attribute`* 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 CLI does not accept white space characters inside the value for the *`attribute`* argument.

3. Enter the command `elb-create-lb-policy`, as in the following example, to create a back-end server authentication policy by referring to `MyPublicKeyPolicy`. You can refer to multiple public key policies. When multiple public key policies are used, the load balancer will try all the keys one by one for authentication. If one of the public keys matches the server certificate, authentication passes.

```
PROMPT>elb-create-lb-policy MyLoadBalancer --policy-name MyBackendServerAu
thenticationPolicy --policy-type BackendServerAuthenticationPolicyType --
attribute "name=PublicKeyPolicyName,value=MyPublicKeyPolicy"
```

4. Enter the command `elb-set-lb-policies-for-backend`, as in the following example, to set `MyBackendServerAuthenticationPolicy` to the back-end server port.

```
PROMPT>elb-set-lb-policies-for-backend-server MyLoadBalancer --instance-port
 443 --policy-names MyBackendAuthenticationPolicy
```

5. Enter the command `elb-describe-lb-policies`, as in the following example, to list all the policies created for `MyLoadBalancer`.

```
PROMPT>elb-describe-lb-policies MyLoadBalancer
```

6. Enter the command `elb-describe-lb-policies`, as in the following example, to view details of `MyBackendServerAuthenticationPolicy`.

```
PROMPT>elb-describe-lb-policies MyLoadBalancer --policy-names MyBackend
ServerAuthenticationPolicy
```

Configuring Health Check Settings

In this example, you configure the health check settings for your back-end servers.

- **To configure health check settings for your back-end server**

  Enter the command `elb-configure-healthcheck` as in the following example.

```
PROMPT> elb-configure-healthcheck  MyLoadBalancer  --headers --target "HT
TP:8080/ping" --interval 30 --timeout 3 --unhealthy-threshold 2 --healthy-
threshold 2
```

Elastic Load Balancing returns the following:

```
HEALTH-CHECK TARGET INTERVAL TIMEOUT HEALTHY-THRESHOLD UNHEALTHY-THRESHOLD
HEALTH-CHECK  HTTP:8080/ping  30     3      2       2
```

Adding Amazon EC2 Instances

In this example, you register your newly created load balancer with your Amazon EC2 instances.

Important

You should only register instances that are in the *Pending* or *Running* state and are not in a Virtual Private Cloud (VPC). If you are using Elastic Load Balancing in a VPC, see Deploying Elastic Load Balancing in Amazon VPC (p. 67).

- **To add Amazon EC2 instances**

  Use the `elb-register-instances-with-lb` command as in the following example.

```
PROMPT> elb-register-instances-with-lb  MyLoadBalancer  --headers --instances
 i-4f8cf126,i-0bb7ca62
```

Elastic Load Balancing returns the following:

```
INSTANCE  INSTANCE-ID
INSTANCE  i-4f8cf126
INSTANCE  i-0bb7ca62
```

Important

Elastic Load Balancing registers your load balancer with the instance using the IP addresses. When the instance is stopped and then restarted, the IP addresses associated with your instance changes. Your load balancer cannot recognize the new IP address, which prevents it from routing traffic to your instances. We recommend you de-register your Amazon EC2 instances from your load balancer after you stop your instance, and then register the new instance ID with the load balancer after you restart your instance. For procedures associated with de-registering and then registering your instances with load balancer, see De-Registering and Registering Amazon EC2 Instances (p. 85).

Expanding a Load Balanced Application to an Additional Availability Zone

In this example, you expand your EC2 application to run in an additional Availability Zone (`us-east-1b`). To do so, you first register the instances in the Availability Zone us-east-1b with the load balancer. You wait for the instances to show up in the OutOfService state for the load balancer. Finally you enable Availability Zone us-east-1b for your load balancer.

Note

It is important to register instances in the new Availability Zone with your load balancer *before* adding the Availability Zone. When you call `EnableAvailabilityZonesForLoadBalancer`, the load balancer begins to route traffic equally amongst all the enabled Availability Zones. If the instances have not been registered, requests going to the new Availability Zone may fail.

Preconditions:

- You have set up an HTTP load balancer in Availability Zone us-east-1a as in How to Create a Load Balancer a HTTPS/SSL Load Balancer (p. 42).
- In Availability Zone us-east-1b, you have launched the instances you intend to register with your load balancer.

Using Query API

To expand a load balanced application to an additional Availability Zone

1. Call `RegisterInstancesFromLoadBalancer` with the following parameters:
   - *`LoadBalancerName`* = `MyLoadBalancer`
   - *`Instances`* = [`i-3a8cf324`, `i-2603ca33`]

2. Call `DescribeInstanceHealth` with the following parameters.
   - *`LoadBalancerName`* = `MyLoadBalancer`
   - Instances = `i-3a8cf324`, `i-2603ca33`

3. When the instances from the previous step are in the *OutOfService* state, you can proceed to the next step. Call `EnableAvailabilityZonesForLoadBalancer`.
   - *`LoadBalancerName`* = `MyLoadBalancer`
   - Availability Zones = `us-east-1b`

The operation returns the updated list of Availability Zones enabled for your load balancer.

Using the Command Line Interface

To expand a load balanced application to an additional Availability Zone

1. Use the `elb-register-instances-with-lb` command as in the following example.

```
PROMPT> elb-register-instances-with-lb  MyLoadBalancer  --headers --instances
 i-3a8cf324, i-2603ca33
```

   Elastic Load Balancing returns the following:

```
INSTANCE  INSTANCE-ID
INSTANCE  i-3a8cf324
INSTANCE  i-2603ca33
INSTANCE  i-4f8cf126
INSTANCE  i-0bb7ca62
```

2. Use the `elb-describe-instance-health` command as in the following example.

   ```
   PROMPT> elb-describe-instance-health  MyLoadBalancer  --headers --instances
    i-3a8cf324,i-2603ca33
   ```

   Elastic Load Balancing returns the following:

   ```
   INSTANCE  INSTANCE-ID STATE
   INSTANCE  i-3a8cf324 OutOfService
   INSTANCE  i-2603ca33 OutOfService
   ```

3. Use the `elb-enable-zones-for-lb` command as in the following example.

   ```
   PROMPT>elb-enable-zones-for-lb  MyLoadBalancer  --headers --availability-
   zones us-east-1b
   ```

   Elastic Load Balancing returns the following:

   ```
   AVAILABILITY_ZONES  AVAILABILITY-ZONES
   AVAILABILITY_ZONES  us-east-1a, us-east-1b
   ```

Disabling an Availability Zone from a Load-Balanced Application

In this example, you disable the Availability Zone us-east-1a for your EC2 application.

This scenario assumes that you have an HTTP load balancer enabled in Availability Zones us-east-1a and us-east-1b.

You disable the Availability Zone for the load balancer first, then give the instances time to go into the OutOfService state before deregistering them from your load balancer.

Note

Your load balancer always distributes traffic to all the enabled Availability Zones. If all the instances in an Availability Zone are deregistered or unhealthy before that Availability Zone is disabled for the load balancer, all requests sent to that Availability Zone may fail until `DisableAvailabilityZonesForLoadBalancer` calls for that Availability Zone.

Using Query API

To disable an availability zone from a Load Balanced Application

1. Call `DisableAvailabilityZonesForLoadBalancer` with the following parameters:

   - *`LoadBalancerName`* = `MyLoadBalancer`
   - Availability Zones = `us-east-1a`

   The operation returns the updated list of Availability Zones enabled for your load balancer.
2. Call `DescribeInstanceHealth` with the following parameters. You have to wait until all of the instances in the disabled Availability Zones are in the *OutOfService* state.

   - *`LoadBalancerName`* = `MyLoadBalancer`
   - *`Instances`* = `i-4f8cf126`, `i-0bb7ca62`

3. Call `DeregisterInstances` with the following parameters:

   - *`LoadBalancerName`* = `MyLoadBalancer`
   - Instances = `i-4f8cf126`, `i-0bb7ca62`

Using the Command Line Interface

To disable an availability zone from a Load Balanced Application

1. Use the `elb-disable-zones-for-lb` command as in the following example.

```
PROMPT> elb-disable-zones-for-lb  MyLoadBalancer  --headers --availability-
zones us-east-1a
```

   Elastic Load Balancing returns the following:

```
AVAILABILITY_ZONES  AVAILABILITY-ZONES
AVAILABILITY_ZONES  us-east-1b
```

2. Use the `elb-describe-instance-health` command as in the following example.

   ```
   PROMPT> elb-describe-instance-health  MyLoadBalancer  --headers --instances
    i-4f8cf126,i-0bb7ca62
   ```

   Elastic Load Balancing returns the following:

   ```
   INSTANCE  INSTANCE-ID STATE
   INSTANCE i-4f8cf126 OutOfService
   INSTANCE i-0bb7ca62 OutOfService
   ```

3. Use the `elb-deregister-instances-from-lb` command as in the following example.

   ```
   PROMPT> elb-deregister-instances-from-lb  MyLoadBalancer  --headers --in
   stances i-4f8cf126,i-0bb7ca62
   ```

   Elastic Load Balancing returns the following:

   ```
   INSTANCE  INSTANCE-ID
   INSTANCE i-3a8cf324
   INSTANCE i-2603ca33
   ```

Managing Security Groups in Amazon EC2

A security group acts as a firewall that controls the traffic allowed into a group of instances. When you launch an Amazon EC2 instance, you can assign it to one or more security groups. You can add rules that govern the allowed inbound traffic to instances in each security group. All other inbound traffic is discarded. You can modify rules for a security group at any time. The new rules are automatically enforced for all existing and future instances in the group.

Elastic Load Balancing provides a special Amazon EC2 source security group that you can use to ensure that a back-end Amazon EC2 instance receives traffic only from Elastic Load Balancing. This feature involves two security groups—the source security group and a security group that defines the ingress rules for your back-end instance. Use the source security group to help define the ingress rules for your back-end instances. Specifically, add or modify a rule to your back-end security group that limits ingress traffic so that it can come only from the source security group. For information on Amazon EC2 security groups, go to Using Security Groups.

The name of the source security group can differ between load balancers. To get the name of the source security group for your load balancer, use the CLI command `elb-describe-lbs` or the Query API action `DescribeLoadBalancers`. You can also find this information in the AWS Management Console. Look in your load balancer's detail section.

To lock down traffic between an Amazon EC2 instance and Elastic Load Balancing

1. Enter the `elb-describe-lbs` command to get the name of the source security group. You must include the `--`*`show-long`* parameter to display the security group's name. For information on using additional parameters with this command, go to Elastic Load Balancing Quick Reference Card.

The following example returns a description of the myLB load balancer.

```
elb-describe-lbs myLB --show-long --headers
```

The following is an example response with emphasis added for the source security group.

```
LOAD_BALANCER,NAME,DNS_NAME,CANONICAL_HOSTED_ZONE_NAME,CANONICAL_HOS
TED_ZONE_NAM
E_ID,HEALTH_CHECK,AVAILABILITY_ZONES,INSTANCE_ID,LISTENER_DESCRIP
TIONS,SOURCE_SE
CURITY_GROUP,CREATED_TIME
LOAD_BALANCER,myLB,myLB-1600421271.us-east-1.elb.amazon
aws.com,(nil),(nil),"{int
erval=30,target=HTTP:80/,timeout=5,healthy-threshold=10,unhealthy-
threshold=2}",
"us-east-1b, us-east-1d","i-f1c4b69d, i-cb8df0a7","{protocol=HTTP,lb-
port=80,ins
tance-port=80,policies=}",example-elb/example-elb-sg,2011-02-13T20:43:23.220Z
```

Important

Do not use the example value *`example-elb-sg`* as part of your ingress security group rule. Enter the `elb-describe-lbs` command to get the owner and name of the source security group for your load balancer.

2. Enter the `ec2-authorize` command to create or update an existing rule so that your back-end instance accepts traffic only from Elastic Load Balancing. Use the name of the security group returned in the previous step as the value for the `--`*`source-group`* parameter.

   In the following example, `ec2-authorize` limits ingress traffic for all back-end instances that belong to a security group named backend-default-sg.

   ```
   ec2-authorize -C cert-X509.pem -K pk.pem backend-default-sg -u example-elb
    -o example-elb-sg
   ```

3. If your security group has rules that are less restrictive than the rule you added in the previous step, use the `ec2-revoke` command to remove the less restrictive rules. For example, an existing rule might allow ingress traffic from the CIDR range 0.0.0.0/0 (all IPv4 addresses).

   The following example uses `ec2-revoke` to remove a rule that allows HTTP traffic from all IPv4 addresses from a security group named backend-default-sg.

   ```
   ec2-revoke backend-default-sg -p 80 -s 0.0.0.0/0
   ```

   ### Important

   If you want to connect directly to your back-end instances, do not revoke ingress rules that allow you to do so. For example, you might have rules that allow ingress traffic on ports 22 (SSH) and 3389 (RDP).

Using IPv6 with Elastic Load Balancing

Elastic Load Balancing supports both Internet Protocol version 6 (IPv6) and Internet Protocol version 4 (IPv4). Clients can connect to your load balancer using either IPv4 or IPv6. Communication between the load balancer and its back-end instances uses only IPv4 (regardless of how the client communicates with your load balancer). This means that your back-end Amazon EC2 instances do not need native IPv6 support.

Elastic Load Balancing provides a public DNS name that combines your load balancer's name and Region. For example, a load balancer named myLB in the US-East Region might be represented by the DNS name myLB-1234567890.us-east-1.elb.amazonaws.com. This base public DNS name returns only IPv4 records.

In addition to the base public DNS name, Elastic Load Balancing provides two additional public DNS names. The first combines the string *ipv6* with the name of your load balancer and Region. This might look like ipv6.myLB-1234567890.us-east-1.elb.amazonaws.com. The ipv6-prefixed DNS name returns only IPv6 records. The second public DNS name combines the string *dualstack* with the name of your load balancer and Region. This might look like dualstack.myLB-1234567890.us-east-1.elb.amazonaws.com. The dualstack-prefixed DNS name returns both IPv4 and IPv6 records.

Most customers will want to use the dualstack-prefixed DNS name to enable IPv6 support for their load balancers. Because the dualstack-prefixed DNS name returns both IPv6 and IPv4 records, clients are able to access the load balancer using either IPv4 or IPv6 as their individual connectivity needs dictate. The ipv6-prefixed DNS name returns only IPv6 addresses, which means that clients with only IPv4 connectivity will not be able to reach the load balancer if they use the ipv6-prefixed DNS name.

Elastic Load Balancing supports X-Forwarded-For request headers for clients that connect using either IPv4 or IPv6. If a client connects using IPv6, Elastic Load Balancing inserts the IPv6 address of the client into the request header. For more information on X-Forwarded-For support, see Terminology and Key Concepts (p. 4).

Elastic Load Balancing allows you to map DNS names to your load balancer with IPv6 in much the same way as you map DNS names with IPv4. If you use a CNAME record to map your DNS name to your load balancer, you can continue to use that method. If you use an Amazon Route 53 hosted zone, you can use the same Elastic Load Balancing command to create a resource record for both IPv4 and IPv6.

Note

IPv6 support is currently not available in all regions. For current IPv6 support, go to Elastic Load Balancing.

IPv6 and CNAME records for Elastic Load Balancing

If you use a CNAME record to map a DNS name such as www.example.com to your load balancer, you can use any of the three public DNS names as the alias in a CNAME record. For example, the following CNAME record maps www.example.com to a load balancer's IPv4 address.

```
www.example.com    CNAME    myLB-1234567890.us-east-1.elb.amazonaws.com
```

The following example maps www.example.com to a load balancer's IPv6 address.

```
www.example.com   CNAME    ipv6.myLB-1234567890.us-east-1.elb.amazonaws.com
```

To handle a mixture of IPv4 and IPv6 address resolution, use the dualstack prefix in your CNAME record.

```
www.example.com    CNAME    dualstack.myLB-1234567890.us-east-1.elb.amazonaws.com
```

IPv6 and Hosted Zones for Elastic Load Balancing

If you use an Amazon Route 53 hosted zone to map a domain name or zone apex to your load balancer, you can use the `elb-associate-route53-hosted-zone` command to create resource records that work with IPv4, IPv6, or both.

To create an IPv4 resource record, specify the value A for the *`--rr-type`* parameter. You can also omit this parameter because its default value is A.

```
elb-associate-route53-hosted-zone myLB --rr-name example.com --rr-type A --
hosted-zone-id Z123456789 --weight 100
```

To create an IPv6 resource record, specify the value AAAA for the *`--rr-type`* parameter.

```
elb-associate-route53-hosted-zone myLB --rr-name example.com --rr-type AAAA --
hosted-zone-id Z123456789 --weight 100
```

To create the equivalent of a dualstack resource record, create a resource record that specifies the value A for the *`--rr-type`* parameter and another resource record that specifies the value AAAA.

```
elb-associate-route53-hosted-zone myLB --rr-name example.com --rr-type A --
hosted-zone-id Z123456789 --weight 100
elb-associate-route53-hosted-zone myLB --rr-name example.com --rr-type AAAA --
hosted-zone-id Z123456789 --weight 100
```

For more information about using Amazon Route 53 with Elastic Load Balancing, see Create a Zone Apex Alias that Points to a Load Balancer (p. 91).

Deploying Elastic Load Balancing in Amazon VPC

Amazon Virtual Private Cloud (VPC) lets you define a virtual networking environment in a private, isolated section of the Amazon Web Services (AWS) cloud.

You can define subnets within your Amazon VPC so that you can group similar kinds of instances based on their IP address range. To load balance your EC2 instances in Amazon VPC you can register the load balancers in multiple Availability Zones by specifying one subnet in each Availability Zone to attach the load balancer to. Because a subnet is created for an Availability Zone, specifying the subnet to attach your load balancer to also ensures that the load balancer is configured to listen to requests in the corresponding Availability Zone. You have complete control of this virtual network, and you can use advanced security features and network access control at the instance level and subnet level.

For more information on Amazon VPC, go to Amazon Virtual Private Cloud User Guide. For information on Amazon VPC subnets, go to Your VPC and Subnets.

When you attach your load balancer to a subnet, you are defining the subnet that traffic should enter through in order to forward the request to registered instances. The registered instances do not need to be in the same subnet that you attach to the load balancer. In order to ensure that your load balancer can scale properly, the subnet that you attach the load balancer to should be at least a /25 CIDR block and should have enough free IP addresses to scale.

Elastic Load Balancing on Amazon VPC works mostly in a similar manner as in Amazon EC2 and supports the same set of features. There is however a significant difference between the way the security groups function on Amazon VPC and Amazon EC2. Within Amazon VPC, you have complete control over the security groups assigned to your load balancer. Unlike in EC2, your load balancer is not a member of a security group that is automatically created; it is only a member of the security groups that you specify. Having complete control over the security groups allows you to choose the ports and protocols to accept. For example, in VPC you can open Internet Control Message Protocol (ICMP) connections for the load balancer to respond to ping requests (however, ping requests will not be forwarded to any registered instances).

Note

IPv6 support is not currently available for load balancers in Amazon VPC.

Note

Dedicated tenancy VPCs are not currently supported by Elastic Load Balancing.

This section walks you through the process of creating, accessing, and managing your load balancers on Amazon VPC.

Prerequisites For Using Elastic Load Balancing in Amazon VPC

Before you get started, be sure you've done the following:

- Sign up for Amazon Web Services (AWS). If you haven't signed up for AWS yet, go to http://aws.amazon.com and click the **Sign Up Now** button.
- Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

Alternatively, you can accomplish load balancing tasks using the command line interfaces or the Query API. Install the tools you'll need to perform Elastic Load Balancing tasks. For information on installing the command line interfaces and the Query API, see Get Set Up with Elastic Load Balancing Interfaces (p. 20).

- Create your Amazon VPC with an internet gateway and create subnets in each Availibility Zone in which you want to load balance your instances. For information on creating Amazon VPC and subnets go to Get Started with Amazon VPC.
- Launch the Amazon EC2 instances that you want to register with your load balancer in your Amazon VPC. For more information about launching Amazon EC2 instances, see Launching and Using Instances.
- The examples in this guide assume that your VPCs are in the US East(Northern Virginia) Region. If your instances are in Europe, you can specify the *EU(Ireland)* Region by using the `--region eu-west-1` parameter from the command line interface or by setting the `AWS_ELB_URL` environment variable.

  For more information on using the region parameter with your Elastic Load Balancing commands, go to the Elastic Load Balancing Quick Reference Card. For information on setting your environment variable, see Installing the Command Line Interface (p. 21). For information about this product's Regions and endpoints, go to Regions and Endpoints in the Amazon Web Services General Reference.

Creating a Basic Load Balancer in Amazon VPC

This topic uses an example to walk you through the process for creating a basic HTTP load balancer on Amazon VPC and registering your EC2 instances with the newly created VPC load balancer. This example uses default configurations for security group, listener protocols and ports, and for the health check.

The following task list gives you a general overview of the steps you'll need to follow to create a basic load balancer in Amazon VPC. After this task list you'll step through detailed procedures for each part of the creation process.

Creating a Basic Load Balancer in Amazon VPC

1	Configure the listeners for your load balancer by specifying the ports and protocols to use for front-end connection (client to load balancer) and back-end connection (load balancer to back-end instance).
2	Configure a health check for your Amazon EC2 back-end instances.
3	Select the subnets to launch your load balancer.
4	Select security groups to assign to your load balancer.
5	Add instances to your load balancer.
6	Review settings.
7	Create and test your load balancer.

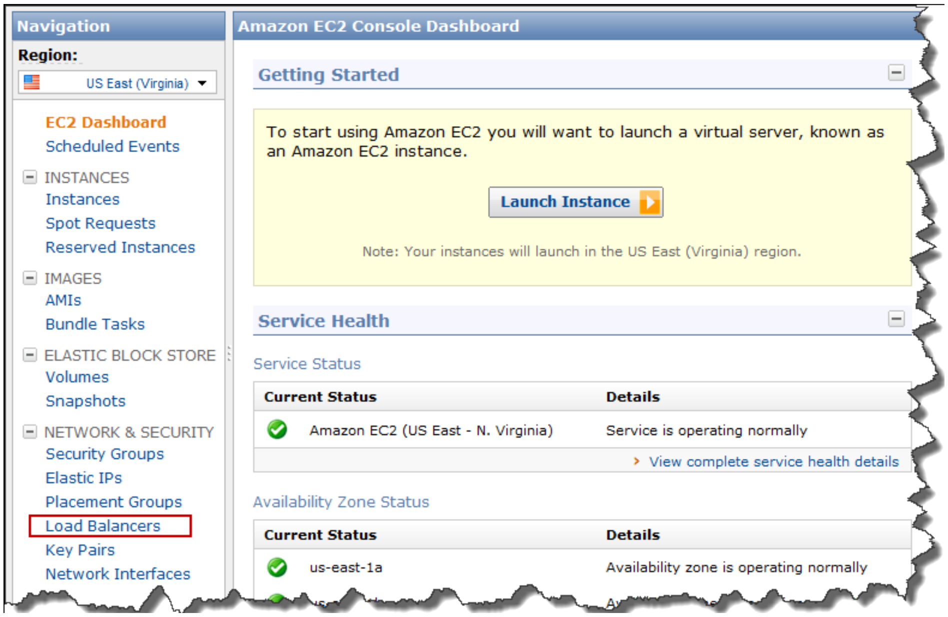
You can choose to create your load balancer within Amazon VPC using the AWS Management Console, the command line interfaces, or Query API.

Using the AWS Management Console

To create a basic load balancer in your Amazon VPC

1. Start the **Create Load Balancer** wizard:

a. In the Amazon EC2 Console Dashboard page, click **Load Balancers** in the **Navigation** pane.



To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

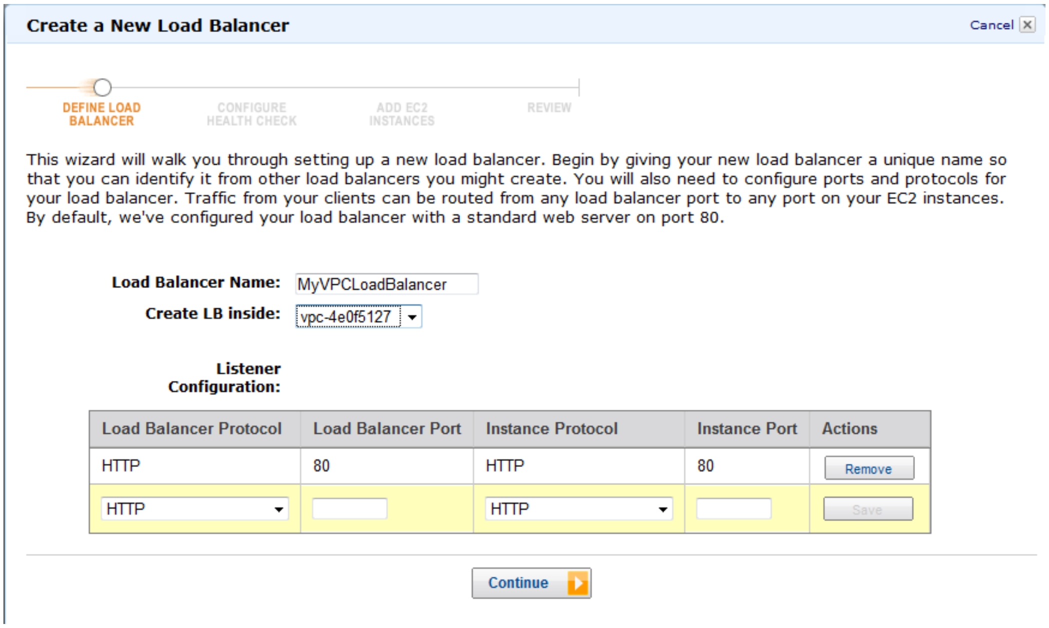
Note: Your instances will launch in the US East (Virginia) region.

Current Status	Details
✓ Amazon EC2 (US East - N. Virginia)	Service is operating normally

Current Status	Details
✓ us-east-1a	Availability zone is operating normally

b. On the **Load Balancers** page, click **Create Load Balancer**.

2. On the **DEFINE LOAD BALANCER** page, enter a name for your Amazon VPC load balancer (e.g., MyVPCLoadBalancer).
3. Click the arrow in the **Create LB inside** box and select the virtual private cloud in which you want to create your load balancer.
4. Leave the **Listener Configuration** set to the default value.



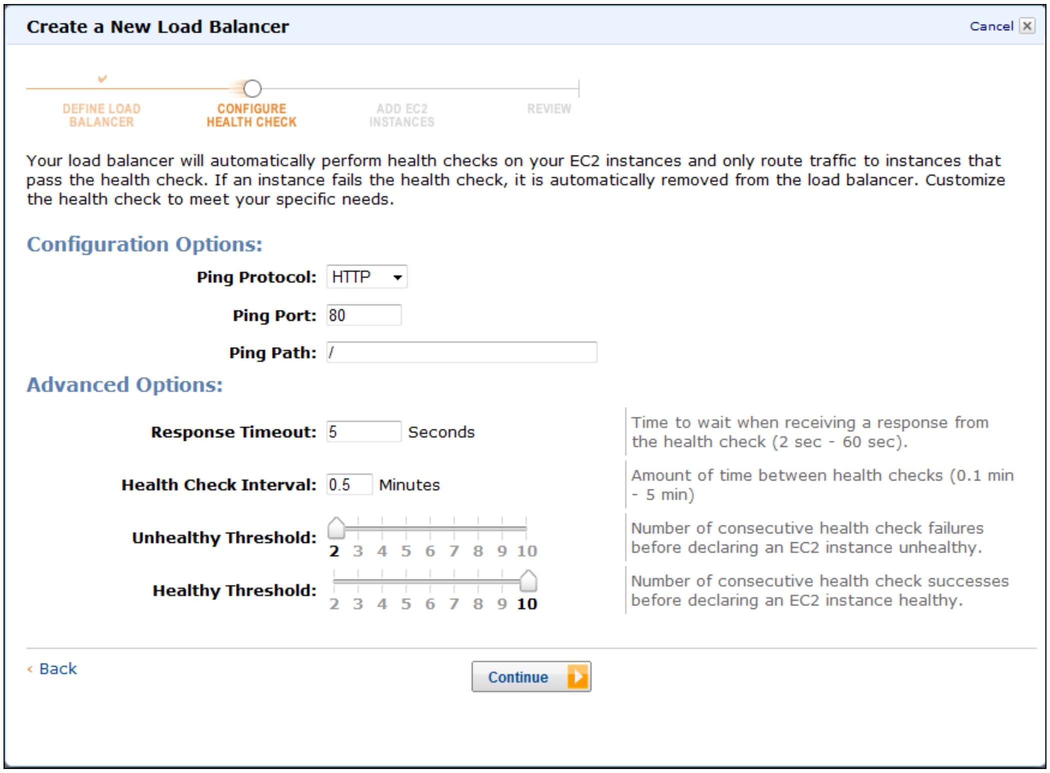
This wizard will walk you through setting up a new load balancer. Begin by giving your new load balancer a unique name so that you can identify it from other load balancers you might create. You will also need to configure ports and protocols for your load balancer. Traffic from your clients can be routed from any load balancer port to any port on your EC2 instances. By default, we've configured your load balancer with a standard web server on port 80.

Load Balancer Name: MyVPCLoadBalancer
Create LB inside: vpc-4e0f5127

Listener Configuration:

Load Balancer Protocol	Load Balancer Port	Instance Protocol	Instance Port	Actions
HTTP	80	HTTP	80	Remove
HTTP		HTTP		Save

5. Click **Continue** to configure the health check for your instances.
6. Configure the health check settings that your application requires.



Create a New Load Balancer

Your load balancer will automatically perform health checks on your EC2 instances and only route traffic to instances that pass the health check. If an instance fails the health check, it is automatically removed from the load balancer. Customize the health check to meet your specific needs.

Configuration Options:

Ping Protocol: HTTP

Ping Port: 80

Ping Path: /

Advanced Options:

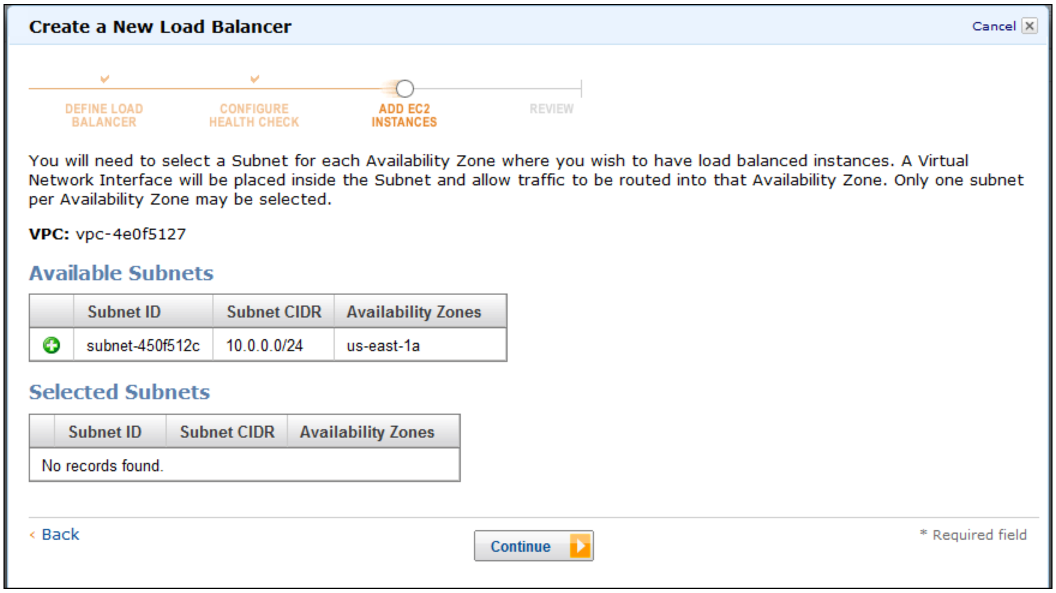
Response Timeout: 5 Seconds — Time to wait when receiving a response from the health check (2 sec - 60 sec).

Health Check Interval: 0.5 Minutes — Amount of time between health checks (0.1 min - 5 min)

Unhealthy Threshold: 2 (2 3 4 5 6 7 8 9 10) — Number of consecutive health check failures before declaring an EC2 instance unhealthy.

Healthy Threshold: 10 (2 3 4 5 6 7 8 9 10) — Number of consecutive health check successes before declaring an EC2 instance healthy.

7. Click **Continue** to select the subnet in which you want to launch your load balancer instance.
8. In the **Available Subnets** table, click the green button at the left to select the subnet in which you want to have your load balanced instances.



Create a New Load Balancer

You will need to select a Subnet for each Availability Zone where you wish to have load balanced instances. A Virtual Network Interface will be placed inside the Subnet and allow traffic to be routed into that Availability Zone. Only one subnet per Availability Zone may be selected.

VPC: vpc-4e0f5127

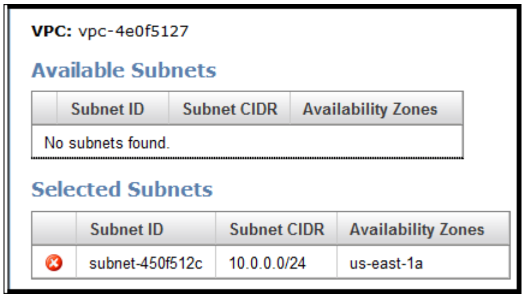
Available Subnets

	Subnet ID	Subnet CIDR	Availability Zones
⊕	subnet-450f512c	10.0.0.0/24	us-east-1a

Selected Subnets

	Subnet ID	Subnet CIDR	Availability Zones
No records found.			

Your selected subnets are displayed in the **Selected Subnets** table.



VPC: vpc-4e0f5127

Available Subnets

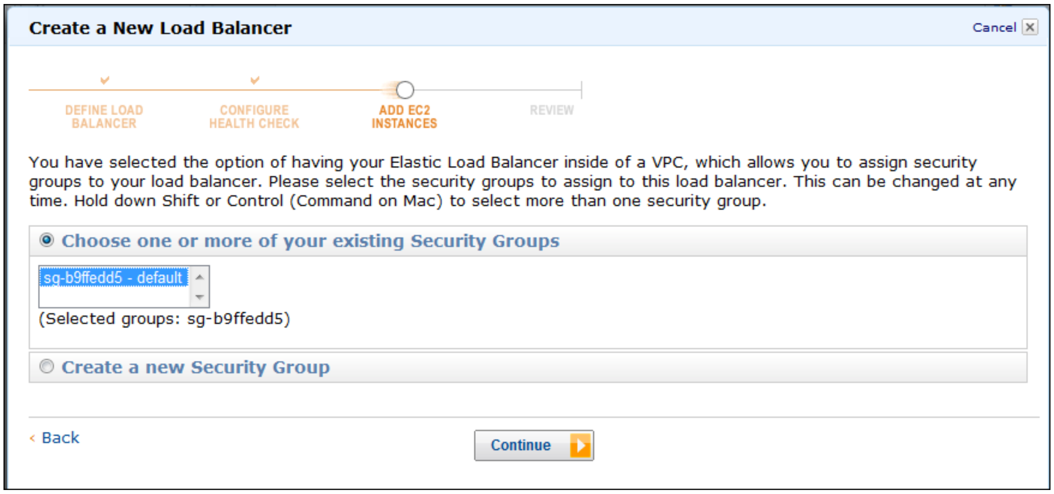
	Subnet ID	Subnet CIDR	Availability Zones
No subnets found.			

Selected Subnets

	Subnet ID	Subnet CIDR	Availability Zones
☒	subnet-450f512c	10.0.0.0/24	us-east-1a

9. Click **Continue** to select security groups to assign to your load balancer.
10. If you use a pre-existing security group, ensure that it allows ingress to the ports that you configured the load balancer to use. If you create a security group in this step, the console will define these ports to be open for you. This example uses the default security group associated with your virtual private cloud.

    Select **Choose one or more of your existing Security Groups** and then select the default security group.



You have selected the option of having your Elastic Load Balancer inside of a VPC, which allows you to assign security groups to your load balancer. Please select the security groups to assign to this load balancer. This can be changed at any time. Hold down Shift or Control (Command on Mac) to select more than one security group.

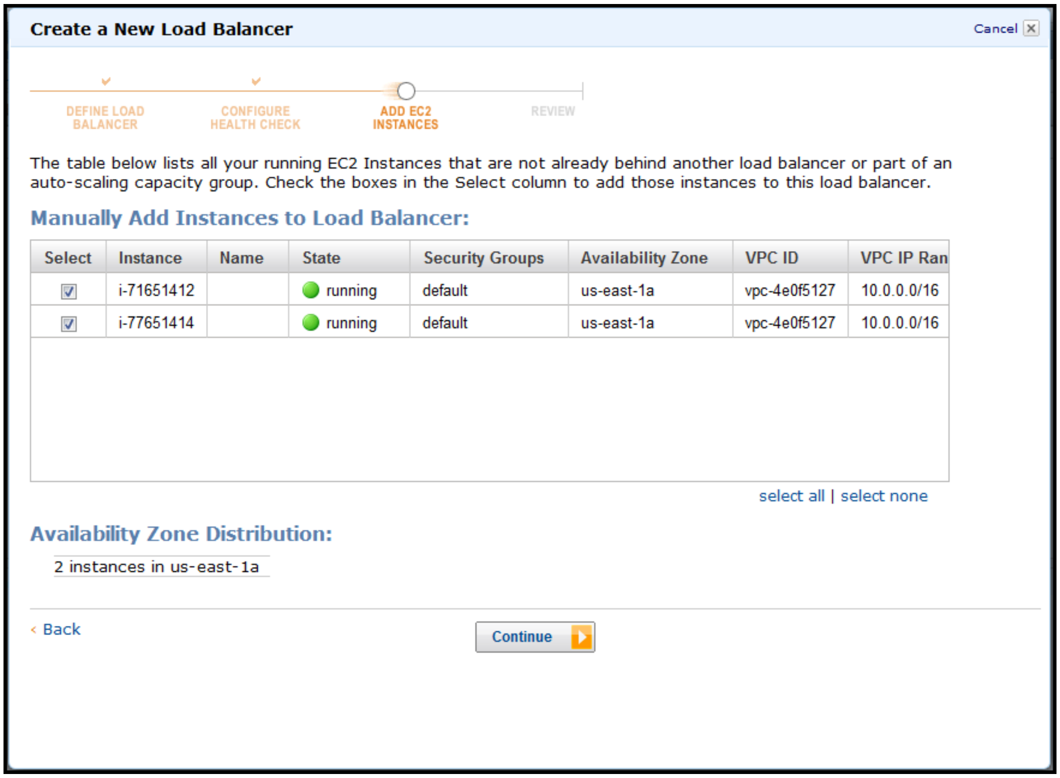
● Choose one or more of your existing Security Groups

sg-b9ffedd5 - default

(Selected groups: sg-b9ffedd5)

○ Create a new Security Group

11. Click **Continue** to add running EC2 instances to your load balancer.
12. In the **Manually Add Instances to LoadBalancer** table, check the boxes in the **Select** column to add instances to your load balancer.



The table below lists all your running EC2 Instances that are not already behind another load balancer or part of an auto-scaling capacity group. Check the boxes in the Select column to add those instances to this load balancer.

Manually Add Instances to Load Balancer:

Select	Instance	Name	State	Security Groups	Availability Zone	VPC ID	VPC IP Ran
☑	i-71651412		● running	default	us-east-1a	vpc-4e0f5127	10.0.0.0/16
☑	i-77651414		● running	default	us-east-1a	vpc-4e0f5127	10.0.0.0/16

Availability Zone Distribution:

2 instances in us-east-1a

Note

When you register a multi-homed instance (an instance that has an Elastic Network Interface attached) with your load balancer, the load balancer will route traffic to the primary IP address of the instance (eth0). For more information on using Elastic Network Interfaces, go to Elastic Network Interfaces.

13. Click **Continue** to review your configuration. On the **REVIEW** page, click **Create** to create your load balancer.

Using the Query API

This example walks you through the process for creating a basic HTTP load balancer on Amazon VPC and registers Amazon EC2 instances with the newly created VPC load balancer. This example uses a default security group.

To create a basic load balancer in your Amazon VPC

1. Call `CreateLoadBalancer` using the following parameters:

   - *`Subnets`* = `subnet-450f512c`
   - *`Listener`*
     - *`Protocol`* = `HTTP`
     - *`InstanceProtocol`* = `HTTP`
     - *`InstancePort`* = `8080`
     - *`LoadBalancerPort`* = `80`
   - *`LoadBalancerName`* = `MyVPCLoadBalancer`

  - *SecurityGroups* = `sg-b9ffedd5`

2. The operation returns the DNS name of your load balancer. You can then map any other domain name (such as www.example.com) to your load balancer's DNS name using CNAME or some other technique.

To register your Amazon EC2 instances with your VPC load balancer

You should only register instances that are in the *Pending* or *Running* state and are in a Virtual Private Cloud (VPC).

- Call `RegisterInstancesWithLoadBalancer` with the following parameters:
  - *LoadBalancerName* = `MyVPCLoadBalancer`
  - Instances = [`i-4f8cf126`, `i-0bb7ca62`]

Note

When you register a multi-homed instance (an instance that has an Elastic Network Interface attached) with your load balancer, the load balancer will route traffic to the primary IP address of the instance (eth0). For more information on using Elastic Network Interfaces, go to Elastic Network Interfaces.

For detailed descriptions of the Elastic Load Balancing API actions, see Elastic Load Balancing API Reference.

Using the Command Line Interface

This example walks you through the process for creating a basic HTTP load balancer on Amazon VPC and registers Amazon EC2 instances with the newly created VPC load balancer. This example uses a default security group that is open to the internet on port 80.

To create a basic load balancer in your Amazon VPC

1. Enter the command `elb-create-lb` as in the following example.

```
PROMPT> elb-create-lb  MyVPCLoadBalancer --subnets subnet-4e05f721  --groups
 sg-b9ffedd5 --listener "lb-port=80,instance-port=8080,protocol=http,instance-
protocol=http"
```

2. Elastic Load Balancing returns the following:

```
DNS-NAME  DNS-NAME
DNS-NAME  MyVPCLoadBalancer-2111276808.us-east-1a.elb.amazonaws.com
```

To register your Amazon EC2 instances with your VPC load balancer

You should only register instances that are in the *Pending* or *Running* state and are in a Virtual Private Cloud (VPC).

1. Use the `elb-register-instances-with-lb` command as in the following example.

```
PROMPT> elb-register-instances-with-lb  MyVPCLoadBalancer  --instances i-
4f8cf126,i-0bb7ca62
```

2. Elastic Load Balancing returns the following:

```
INSTANCE  INSTANCE-ID
INSTANCE  i-4f8cf126
INSTANCE  i-0bb7ca62
```

Note

When you register a multi-homed instance (an instance that has an Elastic Network Interface attached) with your load balancer, the load balancer will route traffic to the primary IP address of the instance (eth0). For more information on using Elastic Network Interfaces, go to Elastic Network Interfaces.

For detailed descriptions of the Elastic Load Balancing commands, see the Elastic Load Balancing Quick Reference Card.

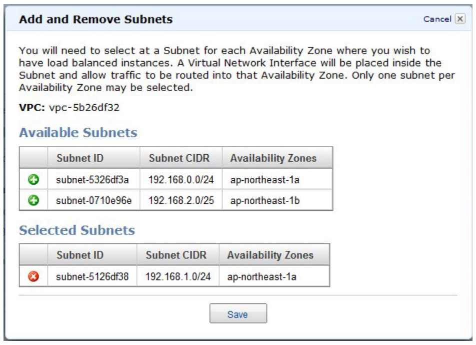
Attaching Your Load Balancer to a Subnet

This example walks you through the process of attaching a subnet to an existing load balancer using either the AWS Management Console, the Query API, or the command line interfaces.

Using AWS Management Console

To attach your load balancer to a subnet

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **Load Balancers** in the **Navigation** pane.
3. On the **Load Balancers** page, select the load balancer that you created for your VPC.
4. The bottom pane displays the details of your load balancer.
5. Click the **Instances** tab.
6. In the **Availability Zones** table, click the green plus sign at the right to attach a subnet.
7. The **Add and Remove Subnets** page is displayed.



You will need to select at a Subnet for each Availability Zone where you wish to have load balanced instances. A Virtual Network Interface will be placed inside the Subnet and allow traffic to be routed into that Availability Zone. Only one subnet per Availability Zone may be selected.

VPC: vpc-5b26df32

Available Subnets

	Subnet ID	Subnet CIDR	Availability Zones
⊕	subnet-5326df3a	192.168.0.0/24	ap-northeast-1a
⊕	subnet-0710e96e	192.168.2.0/25	ap-northeast-1b

Selected Subnets

	Subnet ID	Subnet CIDR	Availability Zones
⊗	subnet-5126df38	192.168.1.0/24	ap-northeast-1a

8. In the **Available Subnets** table, click the green button at the left to select the subnet. You can select only one subnet per Availability Zone.
9. The selected subnet is displayed in the **Selected Subnets** table.
10. Click **Save** to attach the subnet to your load balancer.

Using the Query API

To attach your load balancer to a subnet

1. Call `AttachLoadBalancerToSubnets` with the following parameters:
   - *`Subnets`* = `subnet-4e05f721`
   - *`LoadBalancerName`* = `MyVPCLoadBalancer`
2. The operation returns the subnet ID of the attached subnet.

For detailed descriptions of the Elastic Load Balancing API actions, see the Elastic Load Balancing API Reference.

Using the Command Line Interface

To attach your load balancer to a subnet

1. Enter the command `elb-attach-lb-to-subnets` as in the following example.

   ```
   PROMPT> elb-attach-lb-to-subnets  MyVPCLoadBalancer --subnets subnet-450f512c
   ```

2. The operation returns the subnet ID of the attached subnet.

For detailed descriptions of the Elastic Load Balancing commands, see Elastic Load Balancing Quick Reference Card.

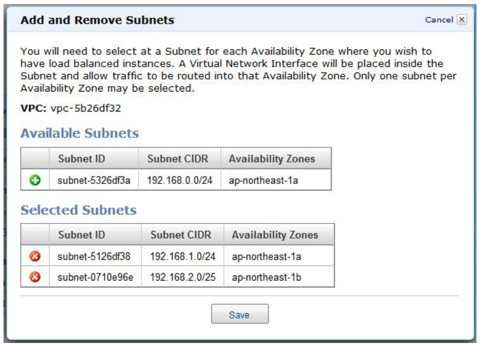
Detaching Your Load Balancer from a Subnet

This example walks you through the process of detaching a subnet from your load balancer using either the AWS Management Console, the Query API, or the command line interfaces.

Using AWS Management Console

To detach your load balancer from subnet

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **Load Balancers** in the **Navigation** pane.
3. On the **Load Balancers** page, select the load balancer that you created for your VPC.
4. The bottom pane displays the details of your load balancer.
5. Click the **Instances** tab.
6. In the **Availability Zones** table, click the red minus sign at the right to detach a subnet.
7. The **Add and Remove Subnets** page is displayed.



You will need to select at a Subnet for each Availability Zone where you wish to have load balanced instances. A Virtual Network Interface will be placed inside the Subnet and allow traffic to be routed into that Availability Zone. Only one subnet per Availability Zone may be selected.

VPC: vpc-5b26df32

Available Subnets

	Subnet ID	Subnet CIDR	Availability Zones
⊕	subnet-5326df3a	192.168.0.0/24	ap-northeast-1a

Selected Subnets

	Subnet ID	Subnet CIDR	Availability Zones
⊗	subnet-5126df38	192.168.1.0/24	ap-northeast-1a
⊗	subnet-0710e96e	192.168.2.0/25	ap-northeast-1b

8. In the **Selected Subnets** table, click the red circle x to the left of the subnet you want to detach.
9. The detached subnet becomes available and is displayed in the **Available Subnets** table.
10. Click **Save** to detach the subnet from your load balancer.

Using the Query API

To detach your load balancer from subnet

1. Call `DetachLoadBalancerFromSubnets` with the following parameters:
   - *`Subnets`* = `subnet-450f512c`
   - *`LoadBalancerName`* = `MyVPCLoadBalancer`
2. The operation returns the subnet ID of the detached subnet.

For detailed descriptions of the Elastic Load Balancing API actions, see the Elastic Load Balancing API Reference.

Using the Command Line Interface

To detach your load balancer from subnet

1. Enter the command `elb-detach-lb-from-subnets` as in the following example.

   ```
   PROMPT> elb-detach-lb-from-subnets  MyVPCLoadBalancer --subnets subnet-
   450f5127
   ```

2. The operation returns the subnet ID of the detached subnet.

For detailed descriptions of the Elastic Load Balancing commands, see the Elastic Load Balancing Quick Reference Card.

Managing Security Groups in Amazon VPC

A security group acts as a firewall that controls the traffic allowed into an instance. When you launch an instance in an Amazon Virtual Private Cloud, you can assign the instance to up to five VPC security groups. The groups act at the instance level, not the subnet level. Therefore, each instance in a subnet in your Amazon VPC could belong to a different set of security groups. If you don't specify a particular group at launch time, the instance automatically belongs to the VPC's default security group. For each group, you add rules that govern the allowed inbound traffic to instances in the group, and a separate set of rules that govern the allowed outbound traffic.

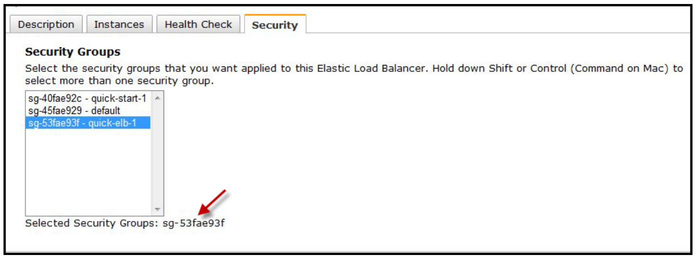
The security groups you've created for Amazon EC2 (i.e., EC2 security groups) are not available to use in your VPC. You must create a separate set of security groups to use in your Amazon VPC (i.e., VPC security groups). The rules you create for a VPC security group can't reference a EC2 security group in your account, and vice versa. Also, VPC security groups have additional capabilities not available to EC2 security groups. For more information on Amazon VPC security groups, go to Security in Your VPC.

This section walks you through the process of assigning a security group to your existing load balancer in Amazon VPC using either the AWS Management Console, Query API or the command line interfaces.

Using the AWS Management Console

To assign a security group to your load balancer

1. In the AWS Management Console, click the Amazon EC2 tab.
2. Click **Load Balancers** in the **Navigation** pane.
3. On the **Load Balancers** page, select the load balancer that you created for your VPC.
4. The bottom pane displays the details of your load balancer.
5. Click the **Security** tab.
6. In the **Security Groups** pane, select the security group.
7. A list of assigned security groups for your load balancer is displayed below the **Security Groups** pane.



Security Groups

Select the security groups that you want applied to this Elastic Load Balancer. Hold down Shift or Control (Command on Mac) to select more than one security group.

sg-40fae92c - quick-start-1
sg-45fae929 - default
sg-53fae93f - quick-elb-1

Selected Security Groups: sg-53fae93f

Using the Query API

To assign a security group to an existing load balancer

1. Call `ApplySecurityGroupsToLoadBalancer` with the following parameters:
   - *`SecurityGroups`* = `sg-53fae93f`
   - *`LoadBalancerName`* = `MyVPCLoadBalancer`

2. The operation returns the security group ID of the assigned security group.

For detailed descriptions of the Elastic Load Balancing API actions, see Elastic Load Balancing API Reference.

Using the Command Line Interface

To assign a security group to your existing load balancer in Amazon VPC

Enter the command `elb-apply-security-groups-to-lb` as in the following example.

```
PROMPT>elb-apply-security-groups-to-lb  MyVPCLoadBalancer --groups sg-53fae93f
```

The operation returns the security group ID of the assigned security group.

For detailed descriptions of the Elastic Load Balancing commands, see the Elastic Load Balancing Quick Reference Card.

Adding a Listener to Your Load Balancer

Elastic Load Balancing supports the load balancing of applications using HTTP, HTTPS (Secure HTTP), TCP, and SSL (Secure TCP) protocols. You can specify the protocols for the front-end connections (client to load balancer) and the back-end connections (load balancer to back-end instance) independently. You choose configurations for the front-end and the back-end connections when you create your load balancer. By default, your load balancer is set to use HTTP protocol for both the connections. The Elastic Load Balancing Listener Configurations Quick Reference (p. 36) table provides information on different configurations, along with the use case best suited for that configuration.

This section describes how to add a new listener on your existing load balancer. Before you get started, be sure you've done the following:

- Created a load balancer with Elastic Load Balancing. For information on how to set up an HTTPS/SSL load balancer with the AWS Management Console, command line interfaces (CLI), or Query API, see Creating a Load Balancer With SSL Cipher Settings and Back-end Server Authentication (p. 42). To learn how to set up a HTTP/TCP load balancer with the AWS Management Console, go to the Get Started with Elastic Load Balancing (p. 11).
- Installed the Elastic Load Balancing tools that you plan to use to perform load balancing tasks. You can add or delete listeners on your existing load balancer using the command line interface (CLI) or the Query API. This functionality is currently not available in the AWS Management Console. For information on installing the command line interface (CLI) or the Query API, see Get Set Up with Elastic Load Balancing Interfaces (p. 20).
  - For detailed descriptions of the Elastic Load Balancing Query API actions, see Elastic Load Balancing API Reference.
  - For detailed descriptions of the Elastic Load Balancing commands, see the Elastic Load Balancing Quick Reference Card.

The following sections include instructions for adding a listener to your existing load balancer using the command line interface (CLI) or the Query API. In this example you configure a new listener for your existing load balancer `MyLoadBalancer` that accepts HTTPS requests on port 443 for the front-end connection and HTTP requests on port 80 for the back-end connection.

Using the Command Line Interface

To add a new listener to your load balancer

1. Enter the command `elb-create-lb-listeners` as in the following example.

```
PROMPT> elb-create-lb-listeners MyLoadBalancer --listener "protocol=HTTPS,lb-
port=443,instance-port=80,instance-protocol=HTTP, cert-
id=arn:aws:iam::55555555555:server-certificate/production/myCert"
```

2. Enter the command `elb-describe-lbs` as in the following example to view the updated details of your load balancer `MyLoadBalancer`.

```
PROMPT> elb-describe-lbs MyLoadBalancer
```

The operation returns a list of updated configurations of your load balancer.

Using the Query API

To add a new listener to your load balancer

Call `CreateLoadBalancerListeners` with the following parameters:

- *`Listener`*
  - *`Protocol`* = `HTTPS`
  - *`InstanceProtocol`* = `HTTP`
  - *`InstancePort`* = `80`
  - *`LoadBalancerPort`* = `443`
  - *`SSLCertificateID`* = `arn:aws:iam::55555555555:server-certificate/production/myCert`
- *`LoadBalancerName`* = **`MyLoadBalancer`**

Call `DescribeLoadBalancers` as in the following example to view the updated configuration information of your load balancer.

- *`LoadBalancerName`* = `MyLoadBalancer`

The operation returns a list of updated configurations of your load balancer.

For detailed descriptions of this Elastic Load Balancing API action, see CreateLoadBalancerListeners in the Elastic Load Balancing API Reference.

Deleting a Listener from Your Load Balancer

This section describes how to delete a listener from your existing load balancer. Before you get started, be sure you've done the following:

- Created a load balancer with Elastic Load Balancing. For information on how to set up a HTTPS/SSL load balancer with the AWS Management Console, command line interface (CLI), or Query API, see Creating a Load Balancer With SSL Cipher Settings and Back-end Server Authentication (p. 42). To learn how to set up a HTTP/TCP load balancer with the AWS Management Console, see Get Started with Elastic Load Balancing (p. 11).
- Installed the Elastic Load Balancing tool that you plan to use to perform load balancing tasks. You can add or delete listeners on your exisiting load balancer using command line interface (CLI) or the Query API. This functionality is currently not available in the AWS Management Console. For information on installing the command line interface (CLI) or the Query API, see Get Set Up with Elastic Load Balancing Interfaces (p. 20).
    - For detailed descriptions of the Elastic Load Balancing Query API actions, see Elastic Load Balancing API Reference.
    - For detailed descriptions of the Elastic Load Balancing commands, see the Elastic Load Balancing Quick Reference Card.

The following sections include instructions for deleting listener from the specified port of your existing load balancer using the command line interface (CLI) or the Query API. In this example you delete a listener from Load Balancer port 80 of your load balancer `MyLoadBalancer`.

Using the Command Line Interface

To delete a listener from your load balancer

1. Enter the command `elb-delete-lb-listeners` as in the following example.

    ```
    PROMPT> elb-delete-lb-listeners MyLoadBalancer lb-ports 80
    ```

2. Enter the command `elb-describe-lbs` as in the following example to view the updated details of your load balancer `MyLoadBalancer`.

    ```
    PROMPT> elb-describe-lbs MyLoadBalancer
    ```

    The operation returns a list of updated configurations of your load balancer.

Using the Query API

To delete a listener from your load balancer

Call `DeleteLoadBalancerListeners` with the following parameters:

- *`LoadBalancerPorts`* = **`80`**
- *`LoadBalancerName`* = **`MyLoadBalancer`**

Call `DescribeLoadBalancers` as in the following example to view the updated configuration information of your load balancer.

- *LoadBalancerName* = `MyLoadBalancer`

The operation returns a list of updated configurations of your load balancer.

For detailed descriptions of this Elastic Load Balancing API action, see the DeleteLoadBalancerListeners in the Elastic Load Balancing API Reference.

De-Registering and Registering Amazon EC2 Instances

Elastic Load Balancing associates your load balancer with your EC2 instance using IP addresses. When the instance is stopped and then restarted, the IP addresses associated with your instance changes. Your load balancer cannot recognize the new IP address, which prevents it from routing traffic to your instances. We recommend that you de-register your Amazon EC2 instances from your load balancer after you stop your instance, and then register the load balancer with your instance after you've restarted.

In this example you de-register your load balancer from your back-end instance that has been stopped, and then register it after you restart your instance.

Before you get started, be sure you've done the following:

- Stop one of your back-end Amazon EC2 instance. For more information, go to Stopping and Starting Instances.

The following sections include instructions for de-registering and registering your back-end instances using the AWS Management Console, the Query API, or the command line interface.

De-Registering Your Amazon EC2 Instances from Your Load Balancer

Using AWS Management Console

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

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **Load Balancers** in the **Navigation** pane.
3. In the **Load Balancers** page, select your load balancer.
4. The bottom pane displays the details of your load balancer.
5. Click **Instances** tab.
6. In the **Instances** table, click **Remove from Load Balancer** of the instance that you want to de-register.

Using the Query API

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

- Call `DeregisterInstancesFromLoadBalancer` with the following parameters:
  - *`Instances`* = **`i-4e05f721`**
  - *`LoadBalancerName`* = **`MyLoadBalancer`**

The operation returns an updated list of remaining instances registered with the load balancer.

For detailed descriptions of this Elastic Load Balancing API action, see DeregisterInstancesFromLoadBalancer in the Elastic Load Balancing API Reference.

Using the Command Line Interface

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

- Enter the command `elb-deregister-instances-from-lb` as in the following example.

```
PROMPT> elb-deregister-instances-from-lb MyLoadBalancer --instances i-4e05f721
```

The operation returns an updated list of remaining instances registered with the load balancer.

For detailed descriptions of the Elastic Load Balancing commands, see the Elastic Load Balancing Quick Reference Card.

Registering Your Amazon EC2 Instances with Your Load Balancer

If you haven't done so yet, you should now restart the instance that you stopped in the previous step.

Using AWS Management Console

To register your back-end instance with your load balancer

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click **Load Balancers** in the **Navigation** pane.
3. In the **Load Balancers** page, select your load balancer.
4. The bottom pane displays the details of your load balancer.
5. Click **Instances** tab.
6. In the **Instances** table, click the green plus sign at the right to add instances to your load balancer.
7. In the **Manually Add Instances to LoadBalancer** table, check the boxes in the **Select** column to add instances to your load balancer.
8. Click **Save** to register your instances.

Using the Query API

To register your back-end instance with your load balancer

Call `RegisterInstancesWithLoadBalancer` with the following parameters:

- *`Instances`* = **`i-802ffe77b`**
- *`LoadBalancerName`* = **`MyLoadBalancer`**

The operation returns an updated list of instances registered with the load balancer.

For detailed descriptions of this Elastic Load Balancing API action, see RegisterInstancesWithLoadBalancer in the Elastic Load Balancing API Reference.

Using the Command Line Interface

To register your back-end instance from your load balancer

Enter the command `elb-register-instances-with-lb` as in the following example.

```
PROMPT> elb-register-instances-with-lb MyLoadBalancer --instances i-802ffe77b
```

The operation returns an updated list of instances registered with the load balancer.

For detailed descriptions of the Elastic Load Balancing commands, see the Elastic Load Balancing Quick Reference Card.

Updating an SSL Certificate for a Load Balancer

In this example, you update an expired SSL certificate for a load balancer.

Note

This example uses APIs and command line interfaces from Elastic Load Balancing and AWS Identity and Access Management. For more information on using Elastic Load Balancing APIs and command line interfaces, see Get Set Up with Elastic Load Balancing Interfaces (p. 20). For more information on using AWS Identity and Access Management APIs and command line interfaces, go to Using AWS Identity and Access Management.

Before you begin, you must have the following:

- An AWS account is signed up for Amazon EC2.
- A load balancer with a HTTPS listener.
- A signed server certificate to replace the expired server certificate.

Important

For information on how to create a signed certificate, go to Creating and Uploading Server Certificates in *Using AWS Identity and Access Management*.

- If you plan to use the command line interface, install the AWS Identity and Access Management command line tools.

  For more information, go to Get the Tools in the *AWS Identity and Access Management Getting Started Guide*.

Using the Query API

To update an SSL certificate for an HTTPS load balancer

1. Call the AWS Identity and Access Management UploadServerCertificate API with the following parameters:

   - *`ServerCertificateName`* = `newCert`

> **Important**
>
> You cannot use the name of the expired certificate. You must use a new name for the *`ServerCertificateName`* parameter.

- *`CertificateBody`* = *`<encoded certificate body>`*
- *`PrivateKey`* = *`<encoded private key>`*
- *`CertificateChain`* = *`<concatenation of the encoded public key certificates>`*

> **Note**
>
> *`CertificateChain`* is optional. If you are using *`CertificateChain`*, then you must order the certificates such that the root certificate is the last certificate in the chain. If you use a certificate chain in a different order, you will receive an error.

- *`Path`* = *`/`*

> **Note**
>
> *`Path`* is optional. If it is not included, the path defaults to `/`. For more information about paths, go to Identifiers for IAM Entities in *Using AWS Identity and Access Management*.

The response includes an Amazon Resource Name (ARN) for your new certificate. Use this new ARN for the *`SSLCertificateId`* parameter in the next step.

2. Call SetLoadBalancerListenerSSLCertificate to replace the expired certificate with the new one.

   - *`LoadBalancerName`* = `test-lb`
   - *`LoadBalancerPort`* = `443`
   - *`SSLCertificateId`* = `arn:aws:iam::322191361670:server-certificate/newCert`

Using the Command Line Interface

To update an SSL certificate for an HTTPS load balancer

1. Enter the command `iam-servercertupload` in verbose mode to upload your certificate to the AWS IAM service.

> **Important**
>
> You cannot use the name of the expired certificate. You must use a new name for the certificate.

```
PROMPT> iam-servercertupload -b /tmp/newCert.pem -k /tmp/test-pri-key.pem
-s newCert [-c <concatenation of the encoded public key certificates>] -v
```

Note

`-c` is optional. If you are using *`-c`*, then you must order the certificates such that the root certificate is the last certificate in the chain. If you use a certificate chain in a different order, you will receive an error.

The response includes the server certificate Amazon Resource Name (ARN) and GUID.

```
arn:aws:iam::322191361670:server-certificate/testCert
ASCACexampleKEZUQ4K
```

2. Copy the ARN for the next step.
3. Enter the command `elb-set-lb-listener-ssl-cert` with an HTTPS listener, as in the following example.

```
PROMPT> elb-set-lb-listener-ssl-cert  test-lb --lb-port 443 --cert-id
arn:aws:iam::322191361670:server-certificate/newCert
```

Using Domain Names with Elastic Load Balancing

Topics

Each Elastic Load Balancing instance that you create has a unique Domain Name System (DNS) name. For example, if you create a load balancer named myLB in the US-East Region, your load balancer might have a DNS name such as myLB-1234567890.us-east-1.elb.amazonaws.com.

If you'd rather use a custom domain name, such as www.example.com, instead of the load balancer DNS name, you can map the custom domain name to the load balancer DNS name. You have two options—either create a Canonical Name (CNAME) record with your existing domain name provider or use Amazon Route 53 to create a hosted zone. A *hosted zone* is an Amazon Route 53 concept that is similar to a zone file on a DNS name server. Like a zone file, a hosted zone contains information about a domain name, including names within the domain and mappings between names and IP addresses.

This section describes how to associate your Elastic Load Balancing instance with a custom domain name—including subdomain names and the *zone apex*. When you register a domain name, you reserve not only the domain name itself, but also an entire set of subdomain names. For example, if you register *example.com*, you can create subdomain names such as *www.example.com* and *foo.bar.example.com*. This set of a domain and its subdomain names is called a *zone*, and a domain name that you reserve, such as *example.com*, is called the zone apex because it sits atop the zone's hierarchy.

Important

This section assumes that you've reserved a domain name. For a list of registrar web sites you can use to register your domain name, go to ICANN.org.

Note

The Time-to-Live (TTL) for an ELB DNS entry is set to 60 seconds. This setting ensures that IP addresses can be re-mapped quickly to respond to events that cause ELB to scale up or down.

Prerequisites For Using Domain Names With Elastic Load Balancing

Before you get started, be sure you've done the following:

- Created a load balancer with Elastic Load Balancing. For information on how to set up a HTTPS/SSL load balancer with the AWS Management Console, command line interfaces (CLI), or Query API, see Creating a Load Balancer With SSL Cipher Settings and Back-end Server Authentication (p. 42). To learn how to set up a HTTP/TCP load balancer with the AWS Management Console, see Get Started with Elastic Load Balancing (p. 11).
- Installed the Elastic Load Balancing tool/s that you plan to use to perform load balancing tasks. You can associate your Elastic load Balancing instance with a custom domain name using command line interface (CLI) or the Query API. This functionality is currently not available in the AWS Management Console. For information on installing the command line interface or the Query API, see Get Set Up with Elastic Load Balancing Interfaces (p. 20).

- For detailed descriptions of the Elastic Load Balancing Query API actions, see Elastic Load Balancing API Reference.
- For detailed descriptions of the Elastic Load Balancing commands, see the Elastic Load Balancing Quick Reference Card.

- You've registered a domain name for your load balancer. For a list of registrar web sites you can use to register your domain name, go to ICANN.org.

Create a CNAME Record for Your Subdomain and Load Balancer

To use a subdomain name that directs traffic to your load balancer, create a CNAME record that associates the subdomain name with your load balancer. The advantage of this method is that creation of CNAME records can be a simple process.

The disadvantage of this method is that you can't use a CNAME record to associate your zone apex with your Elastic Load Balancing instance. DNS rules prohibit the creation of a CNAME record at the zone apex (e.g., example.com). For example, if you own the example.com domain name, you can use a CNAME record for the www.example.com subdomain name, but not for the example.com zone apex. To create a zone apex alias that points to your Elastic Load Balancing instance, use Amazon Route 53.

To associate a subdomain with an Elastic Load Balancing instance

1. Retrieve the public DNS name of your load balancer.

   If you created a LoadBalancer in the previous step, the public DNS name is available as the return value for calls to the CLI command `elb-create-lb` and the Query API action `CreateLoadBalancer`. If you are using an existing load balancer, use `elb-describe-lbs` or `DescribeLoadBalancers` to retrieve your load balancer's public DNS name. You can also find your load balancer's DNS name in the AWS Management Console. Look in your load balancer's detail section.
2. Create a CNAME record with your subdomain name and your load balancer's public DNS name.

   Ask the company that provides your DNS name services (your domain name registrar) to create a CNAME record for your zone. Many domain name registrars provide self-service tools that you can use to create the CNAME record yourself. The following example shows a CNAME record that associates an alias, www.example.com, with a canonical name, the DNS name of an Elastic Load Balancing instance.

   ```
   www.example.com CNAME myLB-1234567890.us-east-1.elb.amazonaws.com
   ```

Create a Zone Apex Alias that Points to a Load Balancer

To map a zone apex (e.g., example.com) to your load balancer, use Amazon Route 53 to create a hosted zone for your domain, then use Elastic Load Balancing to add a zone apex alias to your hosted zone. The zone apex alias associates your zone apex with your Elastic Load Balancing instance. After you create a hosted zone, you can also associate subdomain names with your Elastic Load Balancing instance.

Two separate hosted zones make the zone apex alias possible:

- The custom hosted zone that you create for your domain name with Amazon Route 53.

- The Elastic Load Balancing hosted zone for the Region that contains your load balancer.

For each Region, Elastic Load Balancing maintains a hosted zone that contains the public DNS names for all load balancers in that Region. You can find the Elastic Load Balancing hosted zone ID for your Region by calling the `DescribeLoadBalancers` API action or the `elb-describe-lbs` CLI command. You can also find the Elastic Load Balancing hosted zone ID in the AWS Management Console.

Note

The hosted zone ID listed in your load balancer's **Description** tab is the Elastic Load Balancing hosted zone ID, not your custom hosted zone ID.

If you use the CLI to associate or disassociate domain names, you do not need to use the Elastic Load Balancing hosted zone ID. You need only your custom hosted zone ID and your domain name. If you prefer to use the Amazon Route 53 Query API, however, you need both your custom hosted zone ID and the Elastic Load Balancing hosted zone ID for your LoadBalancer's Region.

Note

This section assumes that you've registered a domain name. For a list of registrar web sites you can use to register your domain name, go to ICANN.org.

Using Command Line Interface

To associate a zone apex with an Elastic Load Balancing instance

1. Create an Amazon Route 53 hosted zone for your domain name.

   If you haven't used Amazon Route 53 before, go to the Amazon Route 53 Getting Started Guide and follow the instructions to create your hosted zone.

   Important

   Note the ID of your hosted zone. You'll need the ID to associate your hosted zone with a load balancer.

2. Create a LoadBalancer with Elastic Load Balancing or choose an existing load balancer.

   For information on how to set up an HTTP/HTTPS load balancer with the AWS Management Console, command line interfaces (CLI) or Query API, see Creating a Load Balancer With SSL Cipher Settings and Back-end Server Authentication (p. 42) . To learn how to set up an HTTP load balancer with the AWS Management Console, see Get Started with Elastic Load Balancing (p. 11).

   For detailed descriptions of the Elastic Load Balancing Query API operations, see Elastic Load Balancing API Reference. For descriptions of all the Elastic Load Balancing commands, see Elastic Load Balancing Quick Reference Card.

   Note

   If you create a new load balancer, wait a few seconds before moving on to the next step. The load balancer's public DNS name can take several seconds to become available to Amazon Route 53. Calls to `elb-associate-route53-hosted-zone` will fail until propagation of your load balancer's public DNS name is complete.

3. Enter the `elb-associate-route53-hosted-zone` command.

   This command creates an association between your zone apex and your Elastic Load Balancing instance by adding an alias resource record set to your hosted zone. The following example creates an association between example.com and a load balancer named myLoadBalancer.

   ```
   elb-associate-route53-hosted-zone myLoadBalancer --rr-name example.com --
   hosted-zone-id Z123456789 --weight 100
   ```

   Note

   For the `hosted-zone-id` parameter, use the hosted zone ID of your custom domain name rather than the Elastic Load Balancing hosted zone ID.

You might have to wait several minutes for your changes to propagate to all Amazon Route 53 DNS servers. For information on how to check the status of your change, go to Checking the Status of Your Change in the *Amazon Route 53 Developer Guide*.

You can also use `elb-associate-route53-hosted-zone` to create aliases for subdomains that are part of your hosted zone. The following example associates the subdomain www.example.com to a customer's Amazon Route 53 hosted zone with ID number Z123456789.

```
elb-associate-route53-hosted-zone myLoadBalancer --rr-name www.example.com --
hosted-zone-id Z123456789 --weight 100
```

Note

Both the `elb-associate-route53-hosted-zone` and `elb-disassociate-route53-hosted-zone` commands work only with AWS Secret Key authentication. Unlike other Elastic Load Balancing CLI commands, these two new Elastic Load Balancing commands do not work with X.509/RSA-PrivateKey credentials.

To remove an association between a zone apex or subdomain and your load balancer, use `elb-disassociate-route53-hosted-zone` to delete the appropriate alias resource record set from your hosted zone. The following example removes the association between the zone apex example.com and the load balancer named myLb.

To disassociate a zone apex from an Elastic Load Balancing instance

- Enter the `elb-disassociate-route53-hosted-zone` command.

  This command removes the association between your zone apex or subdomain and your Elastic Load Balancing instance by deleting an alias resource record set from your hosted zone. The following example removes an association between example.com and a load balancer named myLB. The *`hosted-zone-id`* parameter is your custom hosted zone ID.

  ```
  elb-disassociate-route53-hosted-zone myLB --rr-name example.com --hosted-zone-
  id Z123456789 --weight 100
  ```

Note

The *weight* parameter value must match the value you used to create the resource record set specified in the *rr-name* parameter. If you don't remember the original weight value, use the Amazon Route 53 ListResourceRecordSets action to retrieve the value. For more information, go to ListResourceRecordSets in the *Amazon Route 53 API Reference Guide*. For more information about the *weight* parameter, go to Setting Up Weighted Resource Record Sets in the *Amazon Route 53 API Reference Guide*.

Using the Query API

If you prefer to configure your alias resource record sets with the Query API, you must use the Amazon Route 53 API. You can configure your AliasTarget (obtained from the DescribeLoadBalancers API or elb-describe-lbs CLI) with the following:

- The value of the *CANONICAL_HOSTED_ZONE_NAME* specifies the value of the DNSName element in the AliasTarget.
- The value of the *CANONICAL_HOSTED_ZONE_NAME_ID* specifies the value of the HostedZoneId element in the AliasTarget.

For more information on creating alias resource record sets, go to Creating Alias Resource Record Sets in the *Amazon Route 53 Developer Guide*.

Creating Sticky Sessions

By default, a load balancer routes each request independently to the application 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 application instance. This ensures that all requests coming from the user during the session will be sent to the same application instance.

The key to managing the sticky session is determining how long should your load balancer consistently route the user's request to the same application instance. If your application has its own session cookie, then you can set Elastic Load Balancing to create the session cookie to follow the duration specified by the application's session cookie. If your application does not have its own session cookie, then you can set Elastic Load Balancing to create a session cookie by specifying your own stickiness duration. You can associate stickiness duration for only HTTP/HTTPS load balancer listeners.

Enabling Duration-Based Session Stickiness

The load balancer uses a special load balancer-generated cookie to track the application instance for each request. 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 application instance specified in the cookie. If there is no cookie, the load balancer chooses an application 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 application instance. The policy configuration defines a cookie expiry, which establishes the duration of validity for each cookie.

In this example, you create a stickiness policy and then use it to enable sticky sessions for a load balancer that has load balancer-generated HTTP cookies. Before you get started, be sure you've done the following:

- Created a load balancer with Elastic Load Balancing. For information on how to set up a HTTPS/SSL load balancer with the AWS Management Console, command line interface (CLI), or Query API, see Creating a Load Balancer With SSL Cipher Settings and Back-end Server Authentication (p. 42). To learn how to set up a HTTP load balancer with the AWS Management Console, see Get Started with Elastic Load Balancing (p. 11).
- Installed the Elastic Load Balancing tool that you plan to use to perform load balancing tasks. You can create a stickiness policy using command line interface (CLI) or the Query API. This functionality is currently not available in the AWS Management Console. For information on installing the command line interface (CLI) or the Query API, see Get Set Up with Elastic Load Balancing Interfaces (p. 20).
  - For detailed descriptions of the Elastic Load Balancing commands, see the Elastic Load Balancing Quick Reference Card.
  - For detailed descriptions of the policy configuration Query API for load balancer-generated HTTP cookies, go to CreateLBCookieStickinessPolicy in the *Elastic Load Balancing API Reference*.

Using the Query API

To enable duration-based sticky sessions for a load balancer

1. Call `CreateLBCookieStickinessPolicy` with the following parameters to create a load balancer-generated cookie stickiness policy with a cookie expiration period of 60 seconds.

   - *`LoadBalancerName`* = `MyLoadBalancer`
   - *`PolicyName`* = `MyLoadBalancerPolicy`
   - *`CookieExpirationPeriod`* = `60`

2. Call `SetLoadBalancingPoliciesOfListener` with the following parameters to enable session stickiness for a load balancer using the MyLoadBalancer policy.

   - *`LoadBalancerName`* = `MyLoadBalancer`
   - *`LoadBalancerPort`* = `80`
   - *`PolicyNames`* = `MyLoadBalancerPolicy`

Using the Command Line Interface

To enable duration-based sticky sessions for a load balancer

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

   ```
   PROMPT>elb-create-lb-cookie-stickiness-policy example-lb --policy-name My
   LoadBalancerPolicy --expiration-period 60
   ```

   Elastic Load Balancing returns the following:

   ```
   OK-Creating LB Stickiness Policy
   ```

2. Use the `elb-set-lb-policies-of-listener` command to enable session stickiness for a load balancer using the MyLoadBalancerPolicy.

   ```
   PROMPT>elb-set-lb-policies-of-listener example-lb --lb-port 80 --policy-
   names MyLoadBalancerPolicy
   ```

   Elastic Load Balancing returns the following:

   ```
   OK-Setting Policies
   ```

Enabling Application-Controlled Session Stickiness

The load balancer uses a special cookie to associate the session with the original server that handled the request, but follows the lifetime of the application-generated cookie corresponding to the cookie name specified in the policy configuration. The load balancer only inserts a new stickiness cookie if the application response includes a new application cookie. If the application cookie is explicitly removed or expires, the session stops being sticky until a new application cookie is issued.

In this example, you configure a load balancer for session stickiness when the life of the session follows that of an application-generated cookie. Before you get started, be sure you've done the following:

- Created a load balancer with Elastic Load Balancing. For information on how to set up a HTTPS/SSL load balancer with the AWS Management Console, command line interface (CLI), or Query API, see Creating a Load Balancer With SSL Cipher Settings and Back-end Server Authentication (p. 42). To learn how to set up a HTTP load balancer with the AWS Management Console, see Get Started with Elastic Load Balancing (p. 11).
- Installed the Elastic Load Balancing tool that you plan to use to perform load balancing tasks. You can create a session stickiness policy using command line interface (CLI) or the Query API. This functionality is currently not available in the AWS Management Console. For information on installing the command line interface (CLI) or the Query API, see Get Set Up with Elastic Load Balancing Interfaces (p. 20).
  - For detailed descriptions of the Elastic Load Balancing commands, see the Elastic Load Balancing Quick Reference Card.
  - For detailed descriptions of the policy configuration Query API for application-generated HTTP cookies, go to CreateAppCookieStickinessPolicy in the *Elastic Load Balancing API Reference*

Using the Query API

To Enable Application-Controlled Session Stickiness

1. Call `CreateAppCookieStickinessPolicy` with the following parameters to create an application-generated cookie stickiness policy.

   - *`LoadBalancerName`* = `my-load-balancer`
   - *`PolicyName`* = `my-app-cookie-lb-policy`
   - *`CookieName`* = `my-cookie`

2. Call `SetLoadBalancingPoliciesOfListener` with the following parameters to enable session stickiness for a load balancer using the my-load-balancer policy.

   - *`LoadBalancerName`* = `my-load-balancer`
   - *`LoadBalancerPort`* = `80`
   - *`PolicyNames`* = `my-app-cookie-lb-policy`

Using the Command Line Interface

To Enable Application-Controlled Session Stickiness

1. Use the `elb-create-app-cookie-stickiness-policy` command to create a load application-generated cookie stickiness policy .

```
PROMPT>elb-create-app-cookie-stickiness-policy my-load-balancer -p my-app-
cookie-lb-policy -c my-cookie
```

Elastic Load Balancing returns the following:

```
OK-Creating App Stickiness Policy
```

2. Use the `elb-set-lb-policies-of-listener` command to enable session stickiness for a load balancer using the my-load-balancer.

```
PROMPT>elb-set-lb-policies-of-listener example-lb --lb-port 80 --policy-
names my-app-cookie-lb-policy
```

Elastic Load Balancing returns the following:

```
OK-Setting Policies
```

Monitoring Your Load Balancer Using CloudWatch

Elastic Load Balancing provides data about your load balancers and your back-end application instances to Amazon CloudWatch. The Amazon CloudWatch service collects the data and presents it as readable, near real-time metrics. This metric variable is represented by Amazon CloudWatch as a time-ordered set of data points. Each data point has an associated time stamp and (optionally) a unit of measurement. You can request one or more data points or request an aggregated set of data points called a `statistics set`. For example, you might monitor the number of unhealthy instances behind your load balancer in an Availibility Zone over a specified time period.

This section defines the data that Elastic Load Balancing sends to Amazon CloudWatch. It also explains how to view the data, and how you can set alarms when a metric meets a condition that you specify.

For information about Amazon CloudWatch, go to Introduction to Amazon CloudWatch.

Available Metrics

Elastic Load Balancing sends metrics and dimensions for all the load balancers to Amazon CloudWatch. Amazon CloudWatch uses the metrics to provide detailed monitoring of the load balancers by default. You do not need to specifically enable detailed monitoring.

Note

Elastic Load Balancing only emits Amazon CloudWatch metrics when requests are flowing through the load balancer. If there are no requests or data for a given metric, the metric will not be reported to CloudWatch.

Elastic Load Balancing Metrics

The following Elastic Load Balancing metrics are available from Amazon CloudWatch.

The HTTP response code metrics reflect the count of Elastic Load Balancing response codes that are sent to clients within a given time period. If no response codes in the category 2XX-5XX range are sent to clients within the given time period, values for these metrics will not be recorded in CloudWatch.

Metric	Description
`Latency`	Time elapsed after the load balancer receives a request until it receives the corresponding response. Units: Seconds Valid Statistics: Minimum, Maximum, Average, and Count
`RequestCount`	The number of requests handled by the load balancer. Units: Count Valid Statistics: Sum

Metric	Description
HealthyHostCount	The number of healthy Amazon EC2 instances registered with the load balancer in a specified Availability Zone. Hosts that have not failed more health checks than the value of the unhealthy threshold are considered healthy. When evaluating this metric, the dimensions must be provided for *LoadBalancerName* and *AvailabilityZone*. The metric represents the count of healthy instances in the specified Availability Zone. Instances may become unhealthy due to connectivity issues, health checks returning non-200 responses (in the case of HTTP or HTTPS health checks), or timeouts when performing the health check. To get the total count of all healthy hosts, this metric must be retrieved for each registered Availability Zone and then all the metrics need to be added together. Units: Count Valid Statistics: Minimum, Maximum, and Average
UnHealthyHostCount	The number of unhealthy Amazon EC2 instances registered with the load balancer in a specified Availability Zone. Hosts that have failed more health checks than the value of the unhealthy threshold are considered unhealthy. When evaluating this metric, the dimensions must be provided for *LoadBalancerName* and *AvailabilityZone*. The metric represents the count of unhealthy instances in the specified Availability Zone. Instances may become unhealthy due to connectivity issues, health checks returning non-200 responses (in the case of HTTP or HTTPS health checks), or timeouts when performing the health check. To get the total count of all unhealthy hosts, this metric must be retrieved for each registered Availability Zone and then all the metrics need to be added together. Units: Count Valid Statistics: Minimum, Maximum, and Average
HTTPCode_ELB_4XX	Count of HTTP response codes generated by Elastic Load Balancing that are in the 4xx (client error) series. Units: Count Valid Statistics: Sum
HTTPCode_ELB_5XX	Count of HTTP response codes generated by Elastic Load Balancing that are in the 5xx (server error) series. Elastic Load Balancing may generate 5xx errors if no back-end instances are registered, no healthy back-end instances, or the request rate exceeds Elastic Load Balancing’s current available capacity. Units: Count Valid Statistics: Sum
HTTPCode_Backend_2XX	Count of HTTP response codes generated by back-end instances that are in the 2xx (success) series. Units: Count Valid Statistics: Sum

Metric	Description
HTTPCode_Backend_3XX	Count of HTTP response codes generated by back-end instances that are in the 3xx (user action required) series. Units: Count Valid Statistics: Sum
HTTPCode_Backend_4XX	Count of HTTP response codes generated by back-end instances that are in the 4xx (client error) series. This response count does not include any responses that were generated by Elastic Load Balancing. Units: Count Valid Statistics: Sum
HTTPCode_Backend_5XX	Count of HTTP response codes generated by back-end instances that are in the 5xx (server error) series. This response count does not include any responses that were generated by Elastic Load Balancing. Units: Count Valid Statistics: Sum

Dimensions for Elastic Load Balancing Metrics

To refine the metrics returned by a query, you can use the following dimensions for Elastic Load Balancing. For example, with the *HealthyHostCount* metric, you can use the dimensions *LoadBalancerName* and *AvailabilityZone* to get the average number of healthy instances behind the specified load balancer within the specified Availability Zone for a given period of time. Alternatively, it may be useful to track the minimum number of healthy hosts or the maximum number of unhealthy hosts to better understand how the health and count of backend instances are changing over time.

Elastic Load Balancing data can be aggregated along any of the following dimensions shown in the table below.

Dimension	Description
`LoadBalancerName`	Limits the metric data to Amazon EC2 instances that are connected to the specified load balancer.
`AvailabilityZone`	Limits the metric data to load balancers in the specified Availability Zone.

Viewing Metrics

To view metrics for Elastic Load Balancing, you use Amazon CloudWatch. First, however, you must do the following:

- Create a load balancer and register Amazon EC2 instances with it.
- Download and install the Amazon CloudWatch command line interfaces. For more information, go to Choosing A CloudWatch Interface.

To view metrics using the AWS Management Console

1. Open the Amazon CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the **Navigation** pane, under **Metrics**, click **ELB**.
3. If you want to filter the results by the metric name, under **MetricName**,select the check boxes that correspond to the metrics you want to monitor.
4. If you want to filter the results by Availibility Zone and metric name, under **AvailabilityZone**, select the check boxes that correspond to the Availibility Zone and the metrics you want to monitor.
5. If you want to filter the results by load balancers and metric name, under **LoadBalancerName**, select the check boxes that correspond to the load balancer name and the metrics you want to monitor.

The bottom pane displays the graph for the selected metrics.

To view metrics using the command line interfaces, use the mon-list-metrics command.

To view metrics using the Query API, use the ListMetrics action.

Amazon CloudWatch Alarms

One reason for monitoring metrics in Amazon CloudWatch is to verify that your system is behaving as you expect. If an individual metric goes outside what you consider an acceptable range, you will probably want to be notified. To receive such notification, you can create an alarm in Amazon CloudWatch.

An alarm watches a single metric over a time period you specify. Depending on the value of the metric relative to a threshold that you define, the alarm can send one or more notifications to an Amazon Simple Notification Service (Amazon SNS) topic or to an Auto Scaling policy. A notification is sent only when the changed state has been maintained for a specified period of time.

An alarm has three possible states:

- *OK*—The metric is within the defined threshold.
- *ALARM*—The metric is outside of the defined threshold.
- *INSUFFICIENT_DATA*—The alarm has just started; the metric is not available; or not enough data is available for the metric to determine the alarm state.

Whenever the state of an alarm changes, Amazon CloudWatch sends a notification. For more information on creating a load balancer alarm and sending email notification based on the load balancer alarm, go to Send Email Based on Load Balancer Alarm.

Deleting Your Load Balancer

In this example, you stop using Elastic Load Balancing on a currently load balanced EC2 fleet. You delete the load balancer, which automatically deregisters the associated instances from the load balancer. As soon as the load balancer is deleted, you stop incurring charges for that load balancer.

Note

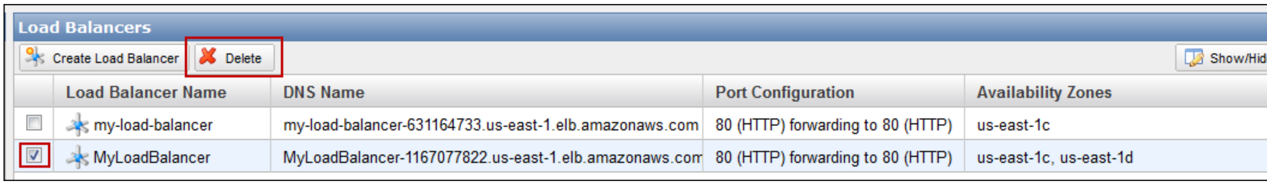
Even after you delete the load balancer and the associated EC2 instances are deregistered, the EC2 instances continue to run. You will continue to incur charges on the Amazon EC2 instances while they are running. For information on stopping your EC2 instances, go to Stopping and Starting Instances in the *Amazon EC2 User Guide*. For information on terminating your EC2 instances, go to Terminate Your Instance.

The following sections include instructions for deleting your load balancer using the AWS Management Console, the Query API, or the command line interface (CLI). If you plan on using either the Query API or the CLI, be sure that you've installed the tools. For information on installing the CLI or the Query API, see Get Set Up with Elastic Load Balancing Interfaces (p. 20).

Using the AWS Management Console

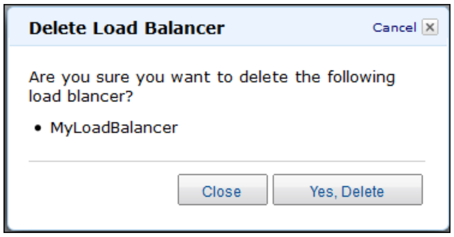
To delete your load balancer

1. On the Amazon EC2 Console Dashboard page, click **Load Balancers** in the **Navigation** pane.
2. On the **Load Balancers** page, select the check box next to the load balancer you want to delete, and then click **Delete**.



	Load Balancer Name	DNS Name	Port Configuration	Availability Zones
☐	my-load-balancer	my-load-balancer-631164733.us-east-1.elb.amazonaws.com	80 (HTTP) forwarding to 80 (HTTP)	us-east-1c
☑	MyLoadBalancer	MyLoadBalancer-1167077822.us-east-1.elb.amazonaws.com	80 (HTTP) forwarding to 80 (HTTP)	us-east-1c, us-east-1d

3. In the **Delete Load Balancer** windows, click **Yes, Delete**.



Are you sure you want to delete the following load blancer?

- MyLoadBalancer

Using the Query API

To delete a load balancer

- Call `DeleteLoadBalancer` with *LoadBalancerName* = `MyLoadBalancer`.

The operation returns an empty response.

Using the Command Line Interface

To delete a load balancer

- Use the `elb-delete-lb` command as in the following example.

```
PROMPT>  elb-delete-lb  MyLoadBalancer
```

Elastic Load Balancing returns the following:

```
Warning: Deleting a LoadBalancer can lead to service disruption to any cus
tomers connected to the load balancer. Are you sure you want to delete this
 load balancer? [Ny]
```

Enter Y to delete the LoadBalancer

Elastic Load Balancing returns the following:

```
OK-Deleting LoadBalancer
```

Controlling User Access to Your AWS Account

Topics

Elastic Load Balancing does not offer its own resource-based permissions system. However, the service integrates with AWS Identity and Access Management (AWS IAM) so that you can specify which Elastic Load Balancing actions a user in your AWS Account can perform with Elastic Load Balancing resources.

Permissions are granted for resources in general. You can't specify a particular Elastic Load Balancing resource in the policy (e.g., a specific load balancer). For example, you could create a policy that gives the Managers group permission to use only `DescribeLoadBalancers`. They could then use those actions with any load balancers that belong to your AWS Account.

Important

Using Elastic Load Balancing with IAM doesn't change how you use Elastic Load Balancing. There are no changes to Elastic Load Balancing actions, and no new Elastic Load Balancing actions related to users and access control.

For examples of policies that cover Elastic Load Balancing actions and resources, see Example Policies for Elastic Load Balancing (p. 106).

No Elastic Load Balancing ARNs

An Amazon Resource Name (ARN) is a unique identifier that some AWS products use to identify resources. For example, you can use an ARN to identify a specific Amazon Simple Queue Service queue or Amazon SimpleDB domain. Elastic Load Balancing has no ARNs for you to use because you can't specify a particular Elastic Load Balancing resource in an IAM policy. When writing a policy to control access to

Elastic Load Balancing actions, you use "*" as the resource. For more information about ARNs, go to ARNs in the *Using AWS Identity and Access Management*.

Elastic Load Balancing Actions

In an IAM policy, you can specify any and all actions that Elastic Load Balancing offers. The action name must be prefixed with the lowercase string `elasticloadbalancing:`. For example: `elasticloadbalancing:ConfigureHealthCheck`, `elasticloadbalancing:*` (for all Elastic Load Balancing Balancing actions). For a list of the actions, refer to the action names in the Elastic Load Balancing Developer Guide.

Elastic Load Balancing Keys

Elastic Load Balancing implements the following policy keys, but no others. For more information about policy keys, go to Condition in the *Using AWS Identity and Access Management*.

AWS-Wide Policy Keys

- `aws:CurrentTime` (for date/time conditions)
- `aws:EpochTime` (the date in epoch or UNIX time, for use with date/time conditions)
- `aws:SecureTransport` (Boolean representing whether the request was sent using SSL)
- `aws:SourceIp` (the requester's IP address, for use with IP address conditions)
- `aws:UserAgent` (information about the requester's client application, for use with string conditions)

If you use `aws:SourceIp`, and the request comes from an Amazon EC2 instance, we evaluate the instance's public IP address to determine if access is allowed.

For services that use only SSL, such as Amazon RDS and Amazon Route 53, the `aws:SecureTransport` key has no meaning.

The key names are case insensitive. For example, `aws:CurrentTime` is equivalent to `AWS:currenttime`.

Example Policies for Elastic Load Balancing

This section shows several simple policies for controlling User access to Elastic Load Balancing.

Note

In the future, Elastic Load Balancing might add new actions that should logically be included in one of the following policies, based on the policy’s stated goals.

Example 1: Allow a group to create and delete load balancers

In this example, we create a policy that gives access to `CreateLoadBalancer` and `DeleteLoadBalancer` The resource is stated as "*", because you can't specify a particular Elastic Load Balancing resource in an AWS IAM policy.

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

Example 2: Allow system administrators to configure load balancers

In this example, we create a group for system administrators, and assign a policy that gives access to the relevant actions.

```
{
   "Statement":[{
   "Effect":"Allow",
      "Action":["elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:ConfigureHealthCheck",
                "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
                "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
                "elasticloadbalancing:DescribeInstanceHealth",
                "elasticloadbalancing:EnableAvailabilityZonesForLoadBalancer",

               "elasticloadbalancing:DisableAvailabilityZonesForLoadBalancer",

                "elasticloadbalancing:CreateAppCookieStickinessPolicy",
                "elasticloadbalancing:CreateLBCookieStickinessPolicy",
                "elasticloadbalancing:SetLoadBalancerPoliciesOfListener"],
      "Resource":"*"
      }
   ]
}
```

Troubleshooting Elastic Load Balancing

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

Troubleshooting Elastic Load Balancing: Error Codes

Topic	Resource
Elastic Load Balancing HTTP error codes	Troubleshooting Elastic Load Balancing Error Codes (p. 109)
HTTP 408	What does a HTTP 408 error indicate? (p. 109)
HTTP 503	What does a HTTP 503 error indicate? (p. 109)
Response codes	How do I know if the HTTP error is being generated by my load balancer or the registered instances? (p. 110)

Troubleshooting Elastic Load Balancing: Health Check

Topic	Resource
Registered instances failing health check	Troubleshooting Elastic Load Balancing Health Check Configuration (p. 110)
Health check URL does not match the configuration	Why is the health check URL different from the URL displayed in API and Console? (p. 111)
Unhealthy instance sending 302 response code to load balancer	What is HTTP 302 Response? (p. 111)

Troubleshooting Elastic Load Balancing : Registering Instances

Topic	Resource
Delay in making a registered instance available for service	Taking too long to register back-end instances. (p. 111)
Unable to register an instance launched from a paid AMI	Unable to register instance launched from a paid AMI. (p. 112)

Troubleshooting Elastic Load Balancing Error Codes

This section provides information about the error codes and messages returned by your load balancer, potential causes, and steps you can take to narrow down and resolve the issue.

What HTTP error codes can my load balancer return?

Your load balancer will return 408 and 503 error responses in addition to any error responses returned by the registered instances.

What does a HTTP 408 error indicate?

- **Description**: Indicates that the client cancelled the request or failed to send a full request.
- **Cause**: 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**: 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.

What does a HTTP 503 error indicate?

Description: Indicates that either the load balancer or the registered instances are causing error. The error can be caused by either one of more of the following:

- **Cause 1**: Insufficient capacity in the load balancer to handle the request.

  Solution: This should be a transient issue and should not last more than a few minutes. If it persists, contact AWS.
- **Cause 2**: Registered instances closing the connection to Elastic Load Balancing.

  Solution: Set idle connection timeouts to more than 60 seconds on your registered instances.
- **Cause 3**: No registered instances.

  Solution: 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.
- **Cause 4**: No healthy instances.

Solution: 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.

How do I know if the HTTP error is being generated by my load balancer or the registered instances?

Your load balancer emits metrics to Amazon CloudWatch for the HTTP response codes sent to clients. For information on using Amazon CloudWatch and defination of the available metrics, see Monitoring Your Load Balancer Using CloudWatch (p. 99). The following metrics are available:

- **Description**: HTTPCode_ELB_4XX
  Cause: At this time, all load-balancer-generated 4XX errors are 408 errors.
  Solution: See What does a HTTP 408 error indicate? (p. 109).
- **Description**: HTTPCode_ELB_5XX
  Cause: At this time, all load-balancer-generated 5XX errors are 503 errors.
  Solution: See What does a HTTP 503 error indicate? (p. 109).
- **Description**: HTTPCode_Backend_2XX
  Cause: Indicates a normal, successful response from the registered instance(s).
  Solution: None
- **Description**: HTTPCode_Backend_3XX
  Cause: Indicates some type of redirect response sent from the registered instance(s).
  Solution:View the access or error logs on your instance(s) to determine the cause of the issue.
- **Description**: HTTPCode_Backend_4XX
  Cause: Indicates some type of client error response sent from the registered instance(s).
  Solution: View the access or error logs on your instance(s) to determine the cause of the issue.
- **Description**: HTTPCode_Backend_5XX
  Cause: Indicates some type of server error response sent from the registered instance(s).
  Solution: View the access or error logs on your instance(s) to determine the cause of the issue.

Troubleshooting Elastic Load Balancing Health Check Configuration

Your load balancer performs health checks on your instances using the protocol, URL, timeout, and interval specified when you configured your load balancer. This section provides information about potential causes and steps you can take to narrow down and resolve the failed health check issues.

Why is my instance failing the load balancer health check?

Your registered instances can fail the health check performed by your load balancer for one or more of the following reasons:

- **Description**: Instance(s) closing the connection to the load balancer.

  Cause: Elastic Load Balancing terminates a connection if it is idle for more than 60 seconds. The *idle connection* is established when there is no read or write event taking place on both the sides of the load balancer (client to load balancer and load balancer to the back-end instance).

  Solution: Set the idle timeout to at least 60 seconds on your registered instances.

- **Description**: Responses timing out.

  Cause: When the load balancer performs a health check, the instance may be under significant load and may take longer than your configured timeout interval to respond.

  Solution: Try adjusting the timeout on your health check settings.

- **Description**: Failing public key authentication.

  Cause: If you are using an HTTPS or SSL load balancer with back-end authentication enabled, the public key authentication will fail if the public key on the certificate does not match the public key configured on the load balancer.

  Solution: Check if your SSL certificate needs to be updated. If your SSL certificate is current, try re-installing the certificate on your load balancer.

Why is the health check URL different from the URL displayed in API and Console?

Description: The health check URL displayed on the console and the API does not match the URL in health check configuration.

Cause: In addition to the health check you configure for your load balancer, a second health check is performed by the service to protect against potential side-effects caused by instances being terminated without being deregistered. To perform this check, the load balancer opens a TCP connection on the same port that the health check is configured to use, and then closes the connection after the health check is completed.

Solution: This extra health check does not affect the performance of your application because it is not sending any data to your back-end instances. You cannot disable or turn off this health check.

What is HTTP 302 Response?

Description : Load balancer returning a HTTP 302 response.

Cause: For an HTTP(S) health check, any response other than a 200 response will be treated as a failed health check. Your load balancer does not follow redirect requests in order to determine if a 200 response will eventually be returned.

Solution: See the preceding section Troubleshooting Elastic Load Balancing Health Check Configuration (p. 110) to troubleshoot the failed health check.

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 will begin sending requests to your instance. This section provides information about potential causes and steps you can take to narrow down and resolve the issues your load balancer might encounter when registering your back-end instances.

Taking too long to register back-end instances.

Description: 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 Why is my instance failing the load balancer health check? (p. 110).

Unable to register instance launched from a paid AMI.

Some older paid AMIs cannot be registered with your load balancer due to special pricing on the bandwidth that would be inaccurately billed if the AMI were used with a load balancer that has its own bandwidth pricing.

Document History

The following table describes the important changes to the *Elastic Load Balancing Developer Guide*. This documentation is associated with the 2011-11-15 release of Elastic Load Balancing. This guide was last updated on 3rd April 2012.

Change	Description	Release Date
Moved Getting Started information	Folded the previously separate *Elastic Load Balancing Getting Started Guide* into this guide. For more information, see Get Started with Elastic Load Balancing (p. 11).	3 April 2012
New listener configuration content	Added a section on choosing the listener configurations for your load balancer that work best for your use case. For more information, see Choosing Listeners for Your Load Balancer (p. 34).	3 April 2012
New troubleshooting content	Added a section on troubleshooting Elastic Load Balancing that provides information on issues you might encounter when using Elastic Load Balancing, discuss the causes, and the steps you can take to resolve the issue. For more information, see Troubleshooting Elastic Load Balancing (p. 108).	3 April 2012
New content on adding and deleting listeners	Added documentation for adding a listener to your load balancer and for deleting a listener from your load balancer using the Query API or the command line tool. For more information, see Adding a Listener to Your Load Balancer (p. 81) and Deleting a Listener from Your Load Balancer (p. 83).	3 April 2012
New features	Updated the API version to 2011-11-15 and added documentation for using Elastic Load Balancing on Amazon VPC. For more information, see Deploying Elastic Load Balancing in Amazon VPC (p. 67).	21 November 2011
New content	Added new content for de-registering Amazon EC2 instances from your load balancer. For more information see, De-Registering and Registering Amazon EC2 Instances (p. 85).	21 November 2011

Change	Description	Release Date
Restructured content	Changed the title of *Using Elastic Load Balancing* to *Accessing Elastic Load Balancing*. Changed the title of *User Scenarios* to *How Do I Use Elastic Load Balancing in Amazon EC2*.	21 November 2011
New content	Added documentation for monitoring the load balancer using Amazon CloudWatch. For more information see Monitoring Your Load Balancer Using CloudWatch (p. 99).	17 October 2011
Restructured content	Moved *Using Domain Names With Elastic Load Balancing* and *Using Ipv6 with Elastic Load Balancing* topics from *Using Elastic Load Balancing* section and placed it under *User Scenarios* section.	17 October 2011
New features	Updated the API version to 2011-08-15 and added documentation for the new configurable SSL ciphers, back-end SSL, and back-end server authentication features. For more information, see How to Create a Load Balancer a HTTPS/SSL Load Balancer (p. 42).	30 August 2011
Restructured content	Consolidated the instructions for setting up a load balancer with HTTP support and with HTTPS support into Creating a Load Balancer With SSL Cipher Settings and Back-end Server Authentication section.	30 August 2011
New content	Added instructions for installing the Elastic Load Balancing command line tool For more information, see Installing the Command Line Interface (p. 21).	04 August 2011
New feature	Updated the API version to 2011-04-05 and added documentation for the new zone apex domain names feature. For more information, see Using Domain Names with Elastic Load Balancing (p. 90).	24 May 2011
New feature	Added documentation for the new Elastic Load Balancing security group for back-end application instance lock-down. For more information, see Managing Security Groups in Amazon EC2 (p. 63).	24 May 2011
New feature	Added documentation for the new Internet Protocol version 6 (IPv6) feature. For more information, see Using IPv6 with Elastic Load Balancing (p. 65).	24 May 2011
Added content	Added information about controlling user access to your AWS account with AWS Identity and Access Management. For more information, see Controlling User Access to Your AWS Account (p. 105).	24 May 2011

Glossary

Access Key ID

An alphanumeric token that uniquely identifies a request sender. This ID is associated with your Secret Access Key.

Amazon Machine Image

An Amazon Machine Image (AMI) is an encrypted machine image stored in Amazon Simple Storage Service (Amazon S3). It contains all the information necessary to boot instances of your software.

Amazon Resource Name (ARN)

A standardized way to refer to an AWS resource. For example: `arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/Bob`. For more information about ARNs, see Using Identifiers in the AWS Identity and Access Management User Guide .

Availability Zone

Amazon EC2 locations are composed of Regions and Availability Zones. Availability Zones are distinct locations that are engineered to be insulated from failures in other Availability Zones and provide inexpensive, low latency network connectivity to other Availability Zones in the same Region.

certificate

A credential that some AWS products use for authentication of AWS Accounts and Users. Also known as an X.509 certificate. The certificate is paired with a private key, and it has an AWS-assigned certificate ID associated with it.

key

A credential that identifies an AWS Account or User to AWS (see Secret Access Key).

Region

Amazon EC2 locations are composed of Regions and Availability Zones. Regions are geographically dispersed and will be in separate geographic areas or countries. Regions consist of one or more Availability Zones.

Secret Access Key

A key assigned to you by Amazon Web Services (AWS) when you sign up for an AWS account. Used for request authentication.

unbounded

Term used in Web Service Definition Language (WSDL), e.g. maxOccurs="unbounded", meaning that the number of potential occurrences is not limited by a set number. Very often used when defining a data type that is a list of other types, such as an unbounded list of integers (element members) or an unbounded list of other complex types that are element/members of the list being defined.

Index