Elastic Load Balancing

Developer Guide

API Version 2012-06-01

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. 4)

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. 14)

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	Configure Listeners for Your Load Balancer (p. 65)
Learn how to install the interfaces needed for accessing the Elastic Load Balancing	Get Set Up with Elastic Load Balancing Interfaces (p. 44)

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. 70)
Learn more about scenarios specific to my instances launched in Amazon EC2	Deploy Elastic Load Balancing in Amazon EC2-Classic (p. 72)
Learn more about scenarios specific to my instances launched in Amazon VPC	Deploy Elastic Load Balancing in Amazon VPC (p. 110)

Where Do I...	Relevant Topics
Learn about potential causes and steps you can take to narrow down and resolve issues with Elastic Load Balancing	Troubleshoot Elastic Load Balancing (p. 171)

What Is Elastic Load Balancing?

Amazon Web Services (AWS) provides Elastic Load Balancing to automatically distribute incoming web traffic across multiple Amazon Elastic Compute Cloud (Amazon EC2) *instances*. With Elastic Load Balancing, you can add and remove EC2 instances as your needs change without disrupting the overall flow of information. If one EC2 instance fails, Elastic Load Balancing automatically reroutes the traffic to remaining running EC2 instances. If the failed EC2 instance is restored, Elastic Load Balancing restores the traffic to that instance. Elastic Load Balancing 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 Elastic Load Balancing, so your servers can focus on their main task.

When you use Elastic Load Balancing to manage traffic to your application, you get 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 Zones.
- Continuous monitoring of the health of Amazon EC2 instances registered with the load balancer so that requests are sent only to the *healthy* instances. If an instance becomes *unhealthy*, Elastic Load Balancing stops sending traffic to that instance and spreads the load across the remaining healthy instances.
- Support for end-to-end traffic encryption on those networks that use secure (HTTPS/SSL) connections.
- The ability to take over the encryption and decryption work from the Amazon EC2 instances, and manage it centrally on the load balancer.
- Support for 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 an Amazon Virtual Private Cloud (Amazon VPC), support for creation and management of security groups associated with your load balancer to provide additional networking and security options.
- Support for use of both 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 associated with the instances. When an 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 have stopped your instance, and then register the load balancer with your instance after you restart it.

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 Elastic Compute Cloud (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 the AWS Management Console to register or deregister instances used by the load balancer as the capacity requirements of your application change over time.

Concepts

Topics

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 the destination to which all requests intended for your load balanced application should be directed. Each load balancer can distribute requests to multiple EC2 instances. A load balancer is represented by a DNS name and a set of ports. 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.

Elastic Load Balancing automatically generates a DNS name for each load balancer instance you create. Typically, the DNS name includes the name of the AWS region in which the load balancer is created. For example, if you create a load balancer named `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 just have to paste the DNS name generated by Elastic Load Balancing into the address field of an Internet-connected web browser to connect to your load balancer.

If you'd rather use a user-friendly domain name, such as www.example.com, instead of the load balancer DNS name, you can create a custom domain name and then associate the custom domain name with the load balancer DNS name. When a request is placed to your load balancer using the custom domain name that you created, it resolves to the load balancer DNS name.

For more information on creating and using a custom domain name for your load balancer, see Configure Custom Domain Name for Your Load Balancer (p. 143).

When you create your load balancer, you have to specify the configurations for your load balancer listeners. A *listener* is a process that listens for connections from incoming requests. It is configured with a protocol and a port number for front-end (load balancer) and back-end (back-end instance) connections. For more information on the ports and protocols supported by Elastic Load Balancing, see Configure Listeners for Your Load Balancer (p. 65).

Registering EC2 Instances

After you've created your load balancer, you have to register your EC2 instances with the load balancer. Your EC2 instances can be within a single Availability Zone or span multiple Availability Zones within a region. Elastic Load Balancing routinely performs health check on all the registered EC2 instances and automatically distributes all incoming requests to the DNS name of your load balancer across your registered, healthy EC2 instances. For more information on the health check of your EC2 instances, see Health Check (p. 7).

Elastic Load Balancing registers your load balancer with your EC2 instances using the IP addresses that are associated with your instances. When an 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-Register and Register Amazon EC2 Instances (p. 136).

Availability Zones and Regions

You can set up your load balancer to load balance incoming requests across EC2 instances in a single Availability Zone or multiple Availability Zones within a region. Your load balancer 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.

If your load balancer detects unhealthy or deregistered instances in an enabled Availability Zone, it stops routing traffic to those instances. Instead, it spreads the load across the remaining healthy Amazon EC2 instances. If all of your instances in a particular Availability Zone are unhealthy or deregistered, but you have set up and registered instances in multiple Availability Zones, Elastic Load Balancing will route traffic to your registered and healthy instances in those other zones. It will resume load balancing to the original instances when they have been restored to a healthy state and are registered with your load balancer.

You can expand the availability of your instances by registering instances in an additional Availability Zone and then enabling that Availability Zone for your load balancer. After you've enabled the new Availability Zone, the load balancer begins to route traffic equally amongst all the enabled Availability Zones. For more information on enabling an Availaibility Zone for your load balancer, see Expanding a Load Balanced Application to an Additional Availability Zone (p. 94).

You can shrink the availability of your instances by first disabling an Availability Zone that was enabled for your load balancer and then deregistering instances in that Availability Zone. After you've disabled the Availability Zone, the load balancer will route the traffic to the registered and healthy instances in the rest of the enabled Availability Zones. For more information on disabling Availability Zone for your load balancer, see Disable an Availability Zone from a Load-Balanced Application (p. 97).

Health Check

Elastic Load Balancing routinely checks the health of each registered 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.

Your load balancer performs health checks on your instances using the protocol, port, URL, timeout, and interval specified when you configured your load balancer. For example, you can configure a health check for your instances as follows - Your load balancer to send request to http://**`node IP address`**:80/index.htm every 5 seconds. Allow 3 seconds for the web server to respond. If the load balancer does not get any response after 2 attempts, take the node out of service. If the load balancer gets 2 successful responses, put the node back in service. Instances that are in service at the time of health check are marked healthy and the instances that are out of service at the time of health check are marked unhealthy.

For information on configuring a health check for the EC2 instances registered with your load balancer, see Configure Health Check for Your Amazon EC2 Instances (p. 17).

Your registered instances can fail the health check for several reasons. The most common reasons for failing a health check are where EC2 instances close connections to your load balancer or where the response from the EC2 instances times out. For information on potential causes and steps you can take to resolve failed health check issues, see Troubleshooting Elastic Load Balancing: Health Check Configuration (p. 174).

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 your load balancer should 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 Enable Duration-Based Session Stickiness (p. 153).
- For more information about creating cookies that allow application-specific session stickiness, see Enable Application-Controlled Session Stickiness (p. 156).

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.

HTTP Methods

The HTTP method (also called the *verb*) specifies the action to be performed on the resource receiving an HTTP request. The standard methods for HTTP requests are defined in RFC 2616, Hypertext Transfer Protocol-HTTP/1.1. Standard methods include GET, POST, PUT, HEAD, and OPTIONS. Some web applications require (and sometimes also introduce) new methods that are extensions of HTTP/1.1 methods. These HTTP extensions can be non-standard. Some common examples of HTTP extended methods include (but are not limited to) PATCH, REPORT, MKCOL, PROPFIND, MOVE, and LOCK. Elastic Load Balancing accepts all standard and non-standard HTTP methods.

When a load balancer receives an HTTP request, it performs checks for malformed requests and for the length of the method. The total method length in an HTTP request to a load balancer must not exceed 127 characters. If the HTTP request passes both the checks, the load balancer sends the request to the back-end EC2 instance. If the method field in the request is malformed, the load balancer responds with a HTTP 400: BAD_REQUEST (p. 172) error message. If the length of the method in the request exceeds 127 characters, the load balancer responds with a HTTP 405: METHOD_NOT_ALLOWED (p. 172) error message.

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

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 the clients that initiate HTTPS sessions with your load balancer and also for connections between the load balancer and your back-end instances.

To enable HTTPS support for your load balancer, you'll have to install an SSL server certificate on your load balancer. The load balancer uses the certificate to terminate and then decrypt requests before sending them to the back-end instances.

For information on using HTTPS/SSL with your load balancer, see Advantages of Using HTTPS/SSL with Elastic Load Balancing (p. 66). For information about creating a load balancer that uses HTTPS, see Create a Load Balancer with SSL Cipher Settings and Back-End Server Authentication (p. 73). For more information on creating and uploading SSL certificates, see Creating and Uploading Server Certificates in the AWS Identity and Access Management documentation.

X-Forwarded-For

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::21f:5bff:febf:ce22:8a2e`.

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

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 the instances in US-east-1a are unhealthy, are deregistered, or if sticky sessions are used and the back-end instance is in US-east-1b, the load balancer in US-east-1a routes the request to the load balancer in US-east-1b. Each of the load balancer then 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-1, previousLoad
BalancerIPAddress-2
```

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:

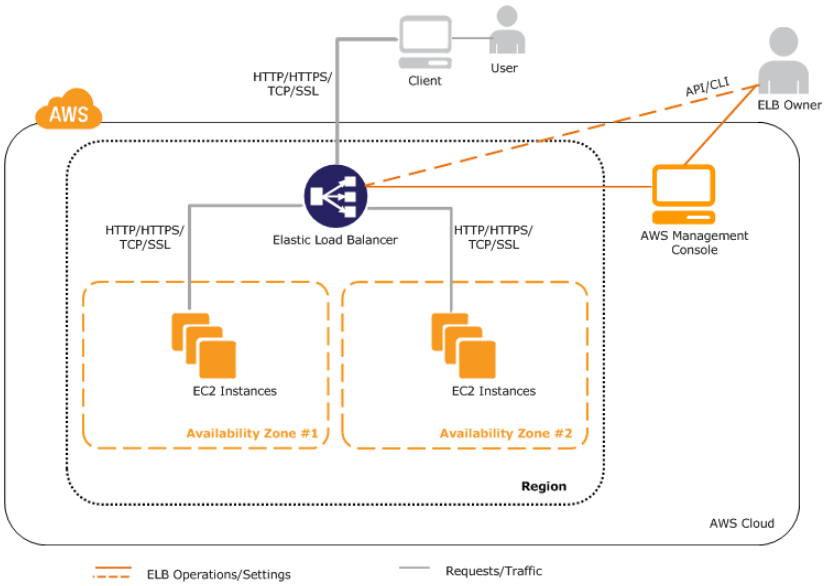
```
X-Forwarded-Proto: originatingProtocol
```

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 take 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 because the clients will need only to 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.

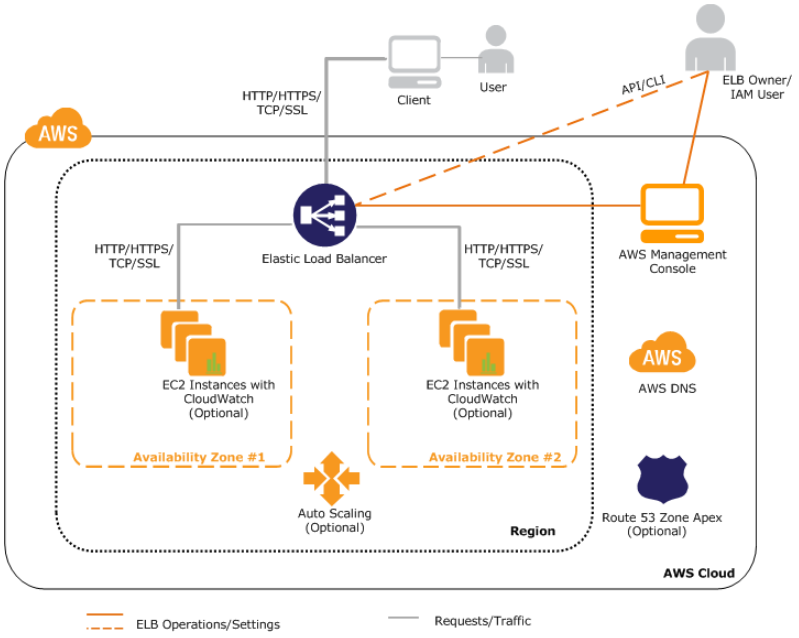
For information on installing and using the command line interfaces, Query APIs, and SDKs provided by AWS, see Get Set Up with Elastic Load Balancing Interfaces (p. 44).

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, see 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 Monitor Your Load Balancer Using Amazon CloudWatch (p. 159). For information on Amazon CloudWatch, see 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 Configure Custom Domain Name for Your Load Balancer (p. 143). For information on Amazon Route 53, see Amazon Route 53 Developer Guide.
AWS Identity and Access Management (IAM)	Manages digital server certificates to use with your load balancer. Your load balancer uses server certificates to terminate and then decrypt requests before sending them to the back-end instances. Use IAM to upload the server certificate to your load balancer. For information on creating and uploading server certificate, see Creating and Uploading Server Certificates. Manages users and user permissions in AWS. IAM provides central control over users and security credentials. Use IAM to 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 Control User Access to Your AWS Account (p. 168). For information on AWS Identity and Access Management, see 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. 14) 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 have read What Is Elastic Load Balancing? (p. 4), and you have decided to load balance your Amazon Elastic Compute Cloud (Amazon EC2) instances, it's time to get started with basic load balancing tasks. Your first task will be to create a load balancer.

Elastic Load Balancing supports load balancing your Amazon EC2 instances launched within any one of the following platforms :

- **EC2-Classic** — Instances launched in EC2-Classic run in a flat network that you share with other customers. We assign each instance with a private IP address from a range of private IP addresses for the EC2-Classic network. We also assign a public IP address for your instance. For more information about Amazon EC2, see What is Amazon EC2? in the *Amazon Elastic Compute Cloud User Guide.*
- **EC2-VPC** — Instances launched in EC2-VPC run in an virtual private cloud (VPC) that is logically isolated in your AWS account. We assign each instance with a private IP address from the private IP address range of your VPC. You have complete control over the VPC assigned to you. For more information about Amazon VPC, see What is Amazon VPC in the *Amazon Virtual Private Cloud User Guide*.

  If your AWS account comes with a default virtual private cloud (default VPC), your instances are launched within the default VPC, by default, unless you specify a subnet from a nondefault VPC. A default VPC combines the benefits of the advanced features provided by EC2-VPC with the ease of use of EC2-Classic. Your default VPC automatically comes with a default subnet in each Availability Zone, an Internet gateway connected to your default VPC, and a default security group associated with your default VPC, among other default configurations. For more information on default VPCs and subnets, see Your Default VPC and Subnets.

For information about how you can tell where you have launched your EC2 instances, see Supported Platforms in the *Amazon Compute Cloud User Guide*.

If you are creating a load balancer for your EC2 instances in EC2-Classic, see Create a Basic Load Balancer in EC2-Classic (p. 15).

If you are creating a load balancer for your EC2 instances in VPC, see Create a Basic Load Balancer in EC2-VPC (p. 23).

If you are creating a load balancer for your EC2 instances in default VPC, see Create a Basic Load Balancer in Default VPC (p. 34).

Create a Basic Load Balancer in EC2-Classic

Topics

This getting started tutorial walks you through the process for creating a basic load balancer for your EC2 instances in EC2-Classic platform.

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.

- 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*.

  If you've already launched your EC2 instances, be sure to create your load balancer in the same region your EC2 instances are launched in.

The following steps outline how to create a basic load balancer in EC2-Classic.

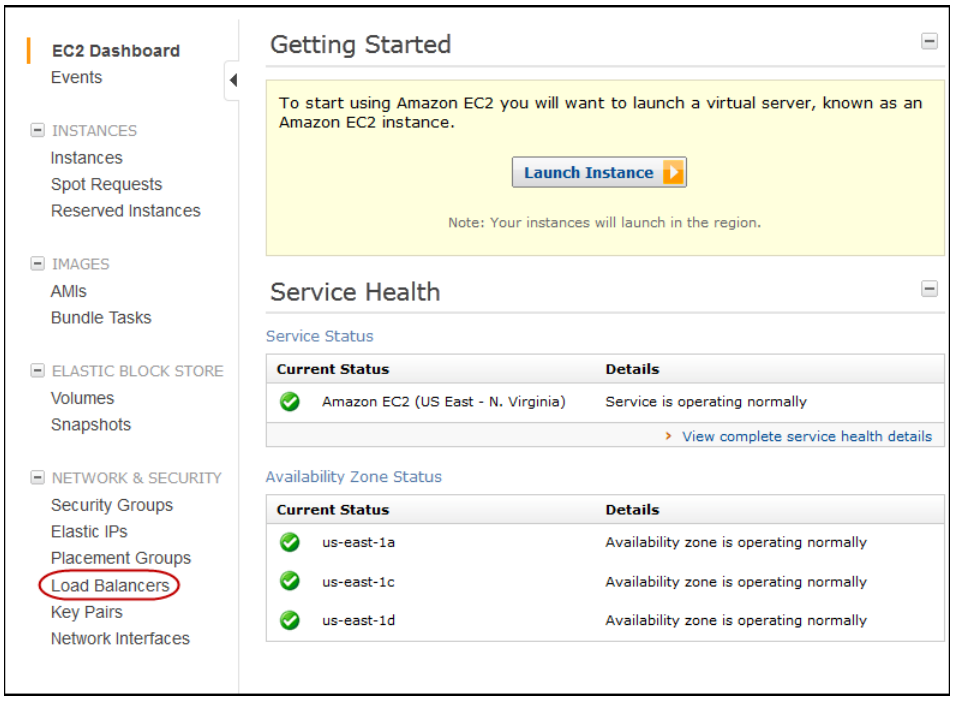
1. Configure the listeners for your load balancer by specifying the ports and protocols to use for the front-end connection (client to load balancer) and the back-end connection (load balancer to back-end instance).
2. Configure a health check for your Amazon EC2 back-end instances.
3. Register your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

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.

Configure Listeners for Your Load Balancer

To configure listeners for 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. Start the **Create Load Balancer** wizard:
   a. On the Amazon EC2 console Getting Started page, in the **EC2 Dashboard** pane, under **NETWORK & SECURITY**, click **Load Balancers**.



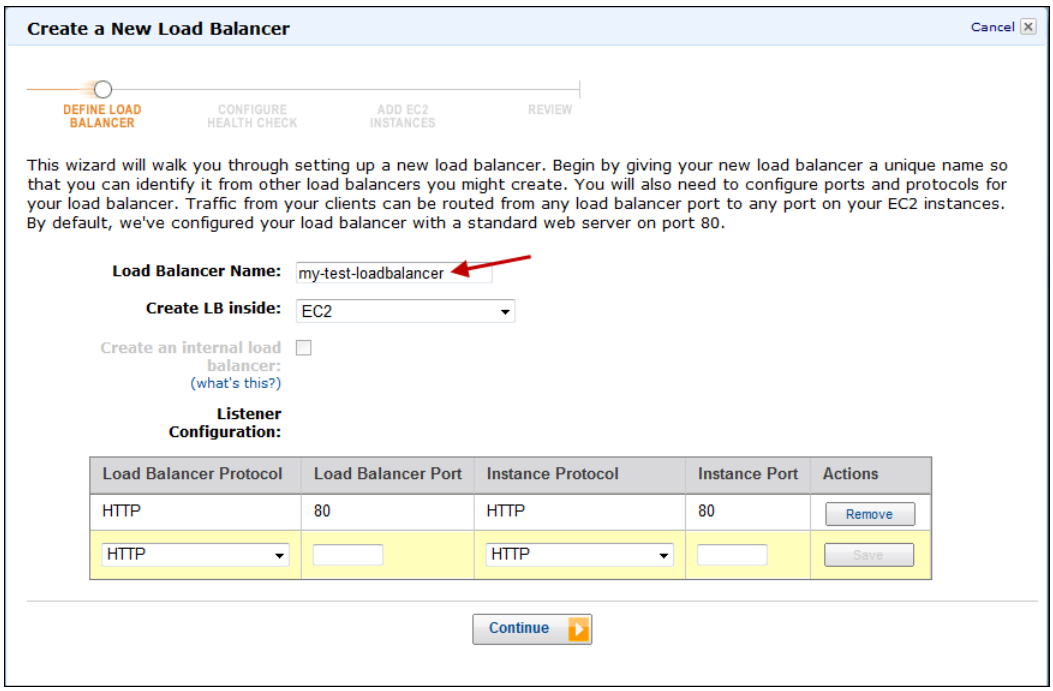
   b. On the **Load Balancers** page, click **Create Load Balancers**.

3. On the **DEFINE LOAD BALANCER** page, make the following selections:
   a. Enter a name for your load balancer (e.g., **`my-test-loadbalancer`**).
   b. Leave **CreateLB inside** set to EC2 for this tutorial.

      Note
      In this tutorial, you are creating a load balancer for your EC2 instances launched within Amazon EC2. If you want to create a load balancer for your EC2 instances inside Amazon Virtual Private Cloud (Amazon VPC), see Create a Basic Load Balancer in EC2-VPC (p. 23)

   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.



4. Click **Continue** to configure health check for your Amazon EC2 instances.

Configure Health Check for Your Amazon EC2 Instances

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

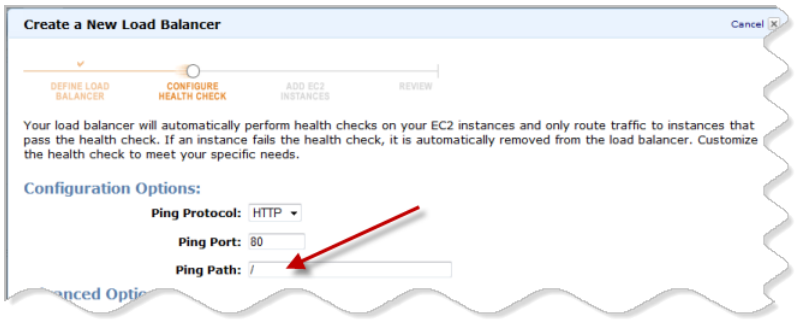
1. On the **CONFIGURE HEALTH CHECK** page of the **Create a New Load Balancer** wizard, set the following configurations:

   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.

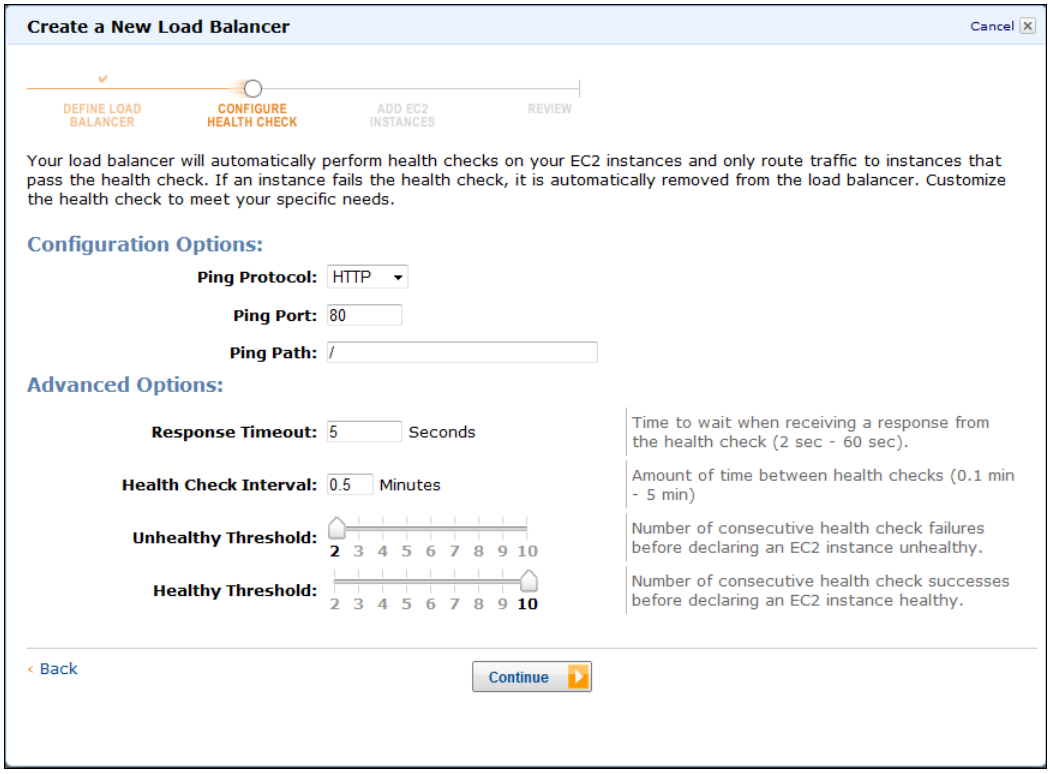
      Important
      Your Amazon EC2 instances must accept incoming traffic on the ping port. This example assumes that each of your instances has a working HTTP server that accepts incoming traffic on port 80.

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



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.



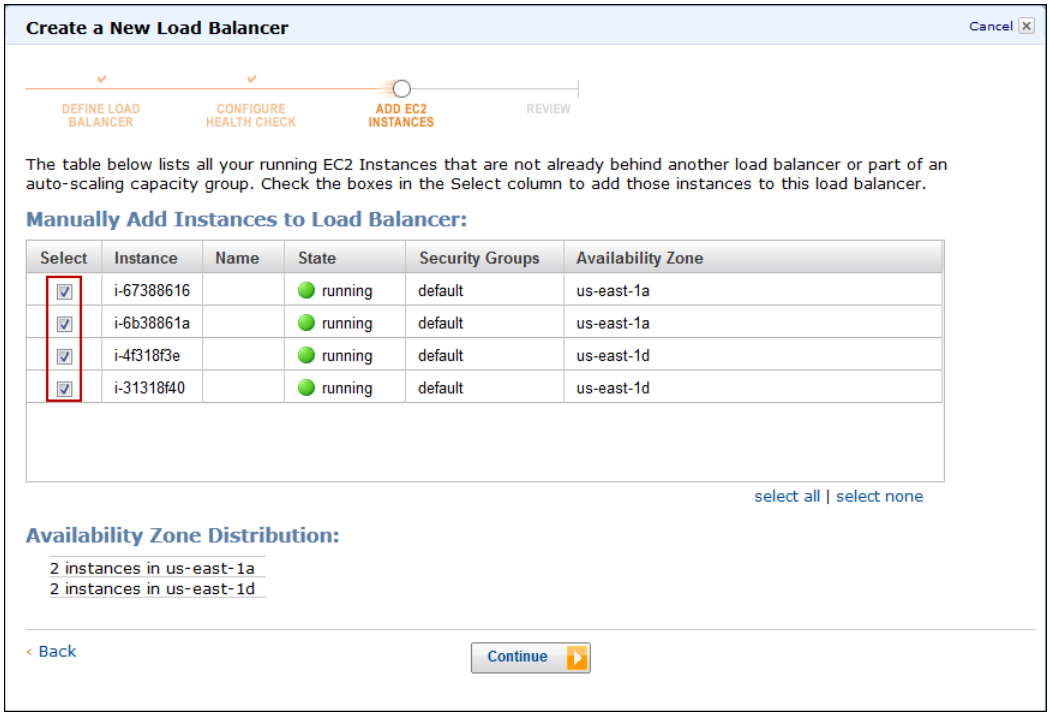
2. Click **Continue** to register your Amazon EC2 instances with your load balancer.

Register Amazon EC2 Instances

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

To register your Amazon EC2 instances

1. On the **ADD EC2 INSTANCES** page, check the boxes in the **Select** column to add instances to your load balancer.



2. Click **Continue** to review your settings and then create your load balancer.

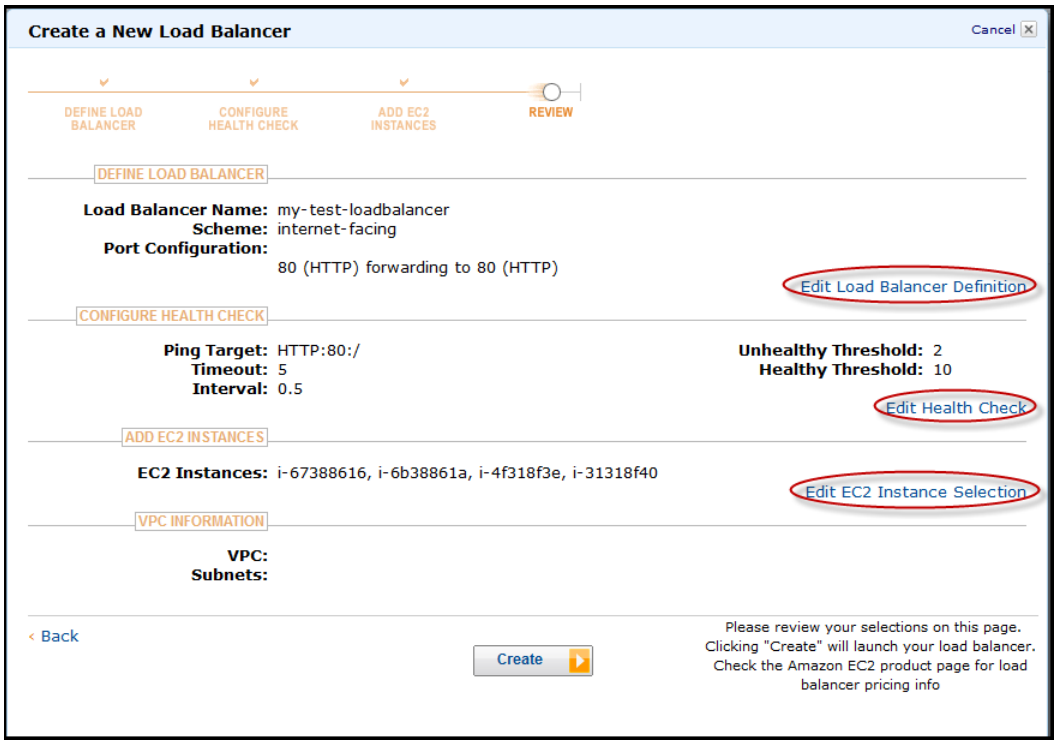
Review Settings and Create Your Load Balancer

Now that you've registered your EC2 instances with your load balancer it's time 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 by clicking the edit link for each setting.

   Note
   You can modify some of the settings even after you've created your load balancer. For example, you can modify the port configurations, health check configurations, and add or remove EC2 instances from your load balancer. For more information, see Using Elastic Load Balancing (p. 70).



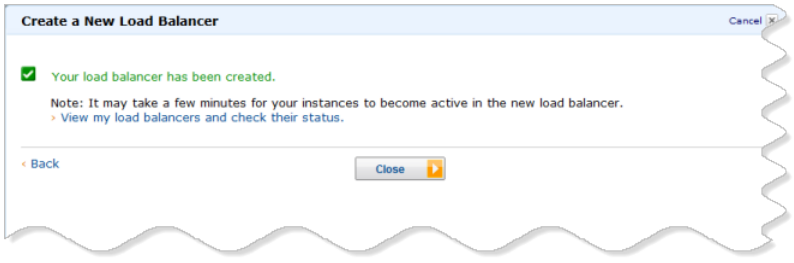
2. Click **Create** to create your load balancer.

Verify the Creation of Your Load Balancer

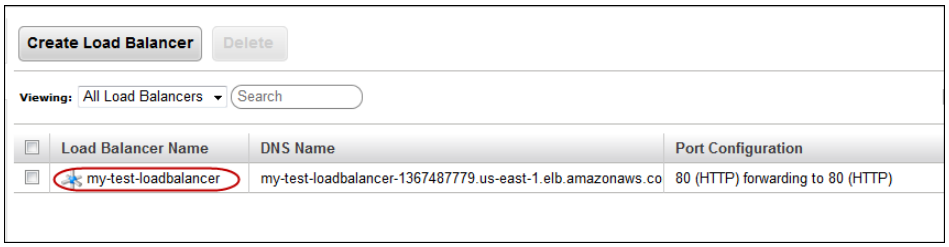
Now that you've created your load balancer, you're ready to verify its settings.

To verify creation of your load balancer

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

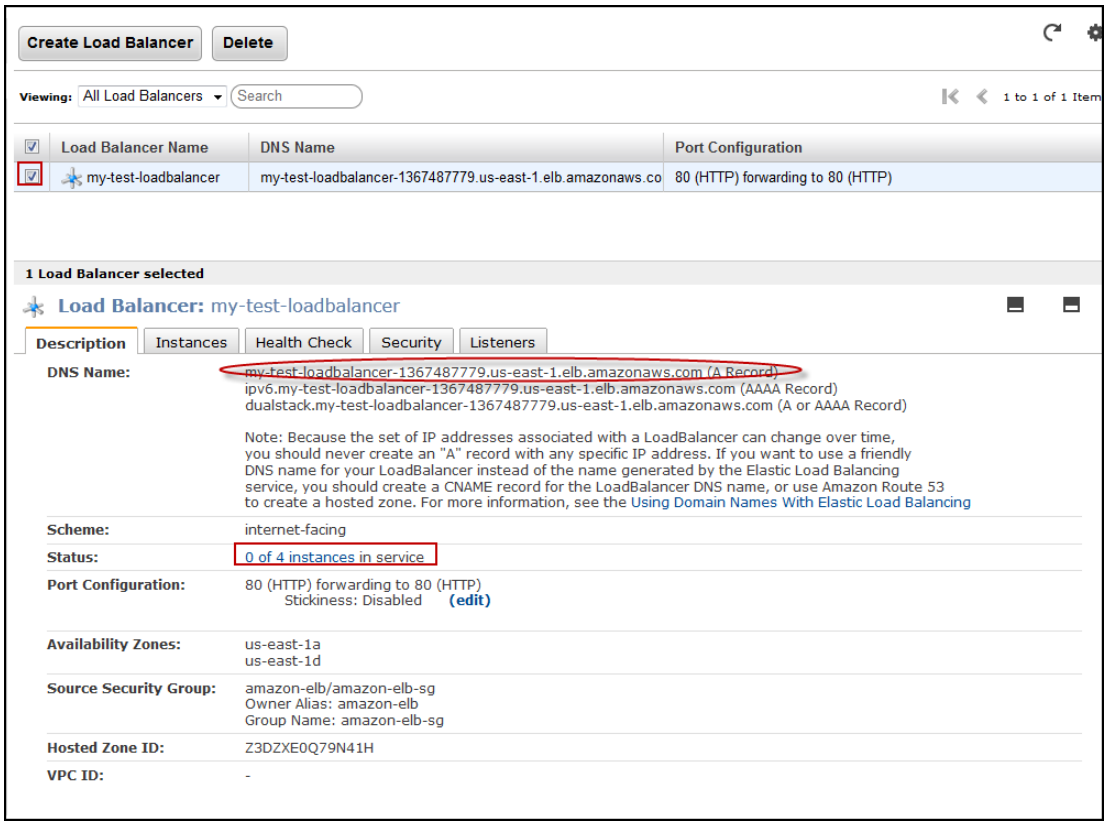


2. When the confirmation window closes, the Load Balancers page opens. Your new load balancer now appears in the list.



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

   The Load Balancer selected pane displays the description of your load balancer. Verify that the descriptions match your specifications.



   If the description in the **Status** row indicates that some of your instances are not in service, its probably because your instances are still in the registration process. For more information, see Troubleshooting Elastic Load Balancing: Registering Instances (p. 175).

4. You can test your load balancer after you've verified that at least one of your EC2 instances is *InService*. 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 you can 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.

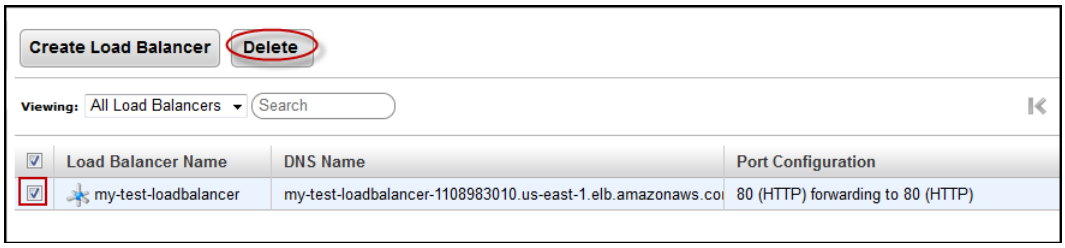
- For information on deleting your load balancer, see the following section.
- Elastic Load Balancing supports the load balancing of applications using HTTP, HTTPS (Secure HTTP), TCP, and SSL (Secure TCP) listener protocols. For a quick overview of the different configurations available for your load balancer, see Elastic Load Balancing Listener Configurations Quick Reference (p. 67).
- 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. 70).
- For information about Elastic Load Balancing usage rates, go to the Elastic Load Balancing product page.

Delete Your Load Balancer

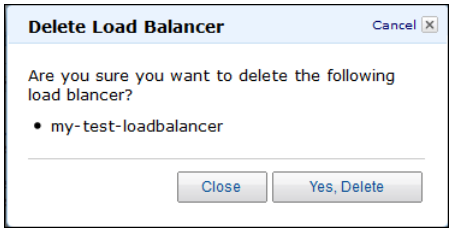
As soon as your load balancer becomes available, you're billed for each hour or partial hour that you keep the load balancer running. After you've decided that you no longer need the load balancer, you can delete it.

To delete your load balancer

1. On the Amazon EC2 console Getting Started page, in the **EC2 Dashboard** pane, under **NETWORK & SECURITY**, click **Load Balancers**.
2. Select the check box next to the load balancer you want to delete, and then click **Delete**.



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



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.

Create a Basic Load Balancer in EC2-VPC

Topics

Amazon Web Services (AWS) supports defining a virtual networking environment in a private, isolated section of the AWS cloud. Within this virtual private cloud (VPC), you can launch Amazon Elastic Compute Cloud (Amazon EC2) instances that have a private IP address, a public IP address, a public DNS hostname, and a private DNS hostname. These instances can communicate with the Internet through an Internet gateway. An Internet gateway enables your instances to connect to the Internet through the Amazon EC2 network edge. You can use a load balancer to monitor and route traffic to your EC2 instances launched within this VPC.

An AWS virtual private cloud (Amazon VPC) can span one or more Availability Zones. Each Availability Zone can have multiple subnets. A subnet in Amazon VPC is a subdivision within an Availability Zone defined by a segment of the IP address range of the VPC. Using subnets you can group your instances based on your security and operational needs. A subnet resides entirely within the Availability Zone it was created in.

When you launch an Amazon EC2 instance within Amazon VPC, you associate it with one or more security groups. A security group acts as a firewall that controls the traffic allowed into an instance.

If your AWS account comes with a default VPC, your Amazon EC2 instances will be launched in default VPC, by default. To load balance your EC2 instances launched within default VPC, you must create your load balancer within the default VPC. For more information, see Detecting Your Supported Platforms and Whether You Have a Default VPC. For information on creating a load balancer within a default VPC, see Create a Basic Load Balancer in Default VPC (p. 34).

The following step-by-step instructions will help you create a basic load balancer in EC2-VPC 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.

- To create and use Elastic Load Balancing load balancers within EC2-VPC, you have to first create your VPC with an Internet gateway, and then launch your EC2 instances within your VPC. For more information, see Get Started with Amazon VPC.
- Make a note of the VPC ID, subnet ID, and the security group associated with the VPC. You will need them later when you create your load balancer.

The following steps outline how to create a basic load balancer in EC2-VPC.

1. Configure the listeners for your load balancer by specifying the ports and protocols to use for the front-end connection (client to load balancer) and the back-end connection (load balancer to back-end instance).
2. Configure a health check for your Amazon EC2 back-end instances.
3. Launch your load balancer instance in a subnet.
4. Assign a security group to your load balancer.
5. Register your Amazon EC2 instances with the load balancer.
6. Review settings and create your load balancer.
7. Verify that your load balancer is created.
8. [Optional] Delete your load balancer.

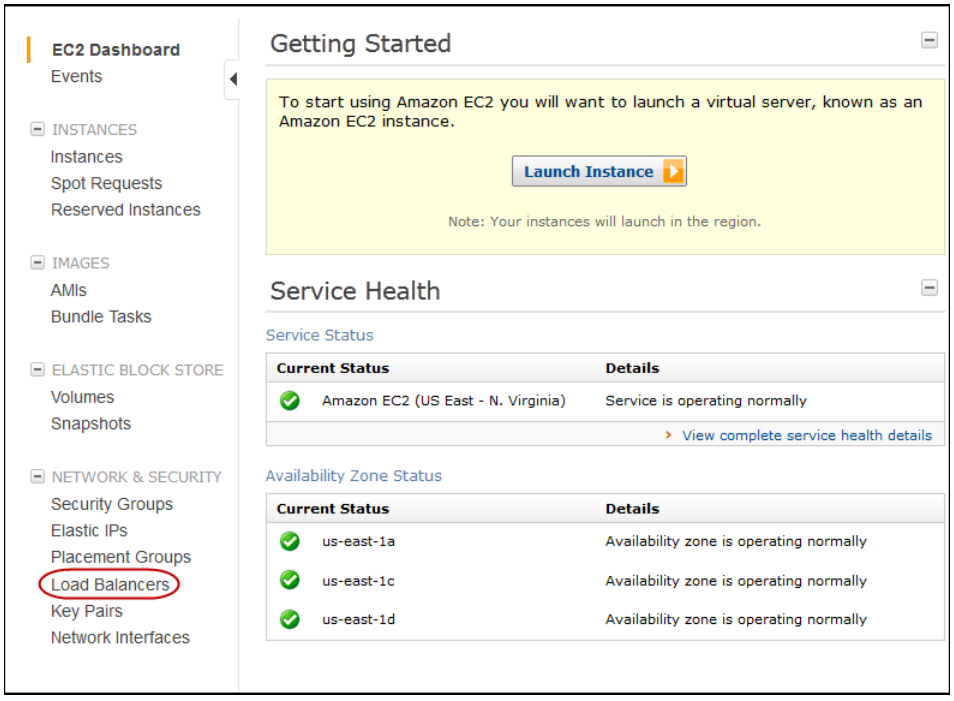
Important
The load balancer you're about to create will be live (and not running in a sandbox). If you are not signed up for the free usage tier, you will incur the standard Elastic Load Balancing usage fees for the load balancer until you terminate it. The total charges will be minimal (typically less 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.

Configure Listeners for Your Load Balancer

To configure listeners for 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. Start the **Create Load Balancer** wizard:

   a. On the Amazon EC2 console Getting Started page, in the **EC2 Dashboard** pane, under **NETWORK & SECURITY**, click **Load Balancers**.



b. Click **Create Load Balancer**.

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

   a. Enter a name for your load balancer (e.g., **`my-test-loadbalancer`**).

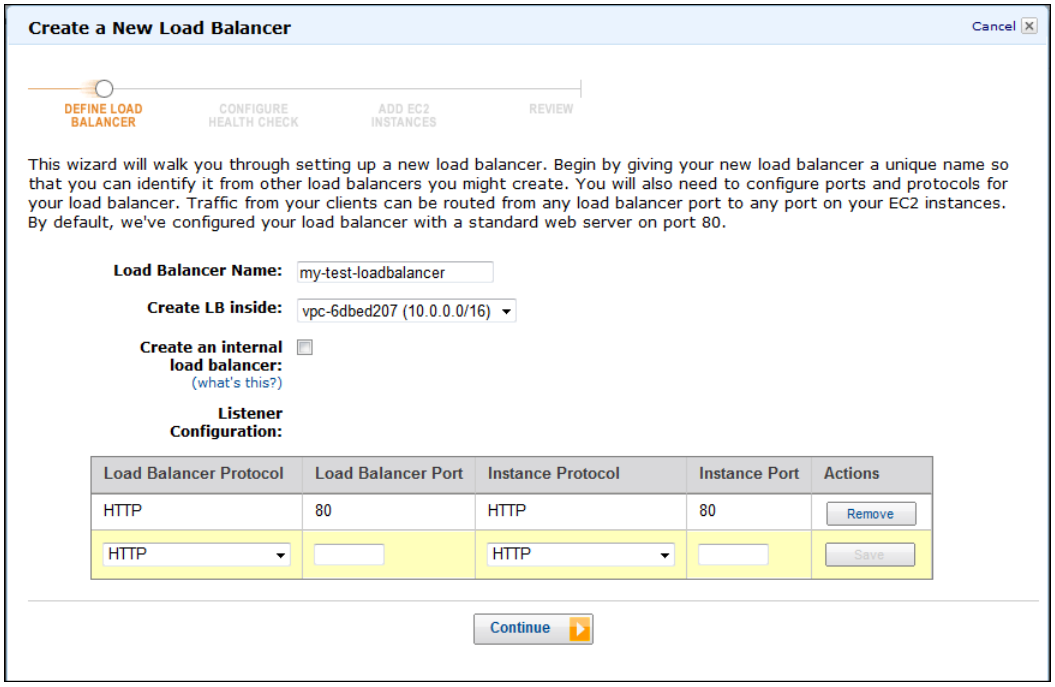
   b. Click the arrow in the **Create LB inside** drop-down box and select the VPC ID in which you want to create your load balancer.

   c. Leave the **Create an internal load balancer** box blank for this tutorial.

      Note
      In this tutorial, you are creating an Internet-facing load balancer with a publicly resolvable DNS name that resolves to public IP addresses. If you want to create an internal load balancer with a publicly resolvable DNS name that resolves to private IP addresses, see Deploy Elastic Load Balancing in Amazon VPC (p. 110)

   d. Leave **Listener Configuration** set to the default value for this tutorial.

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



4. Click **Continue** to configure health check for your Amazon EC2 instances.

Configure Health Check for Your Amazon EC2 Instances

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:

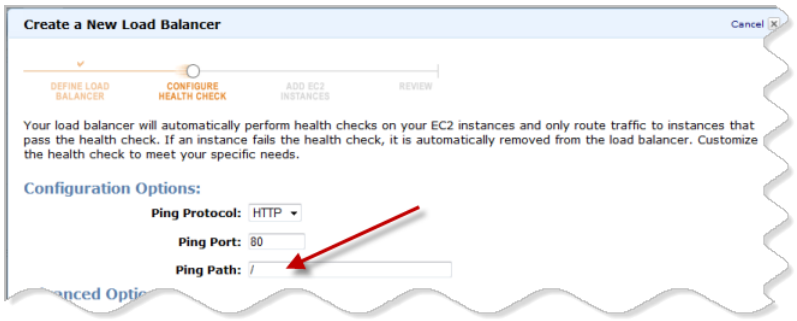
   a. Leave **Ping Protocol** set to its default value of `HTTP`.

   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.

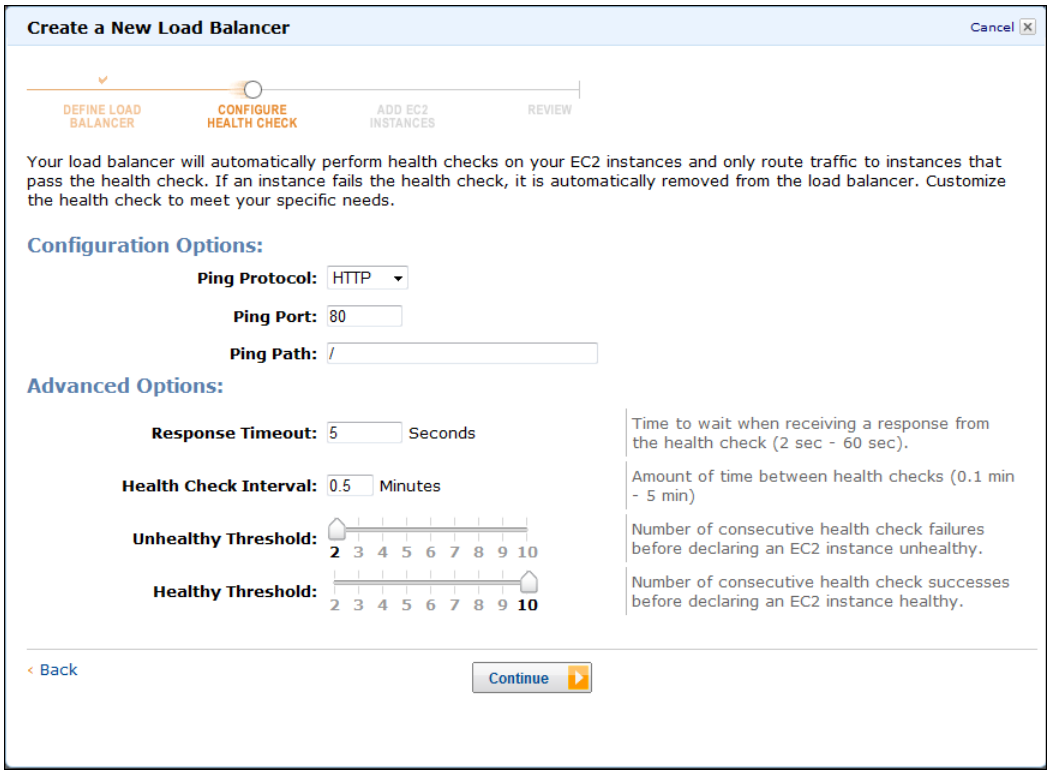
   Important
   Your Amazon EC2 instances must accept incoming traffic on the ping port. This example assumes that each of your instances has a working HTTP server that accepts incoming traffic on port 80.

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



   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.



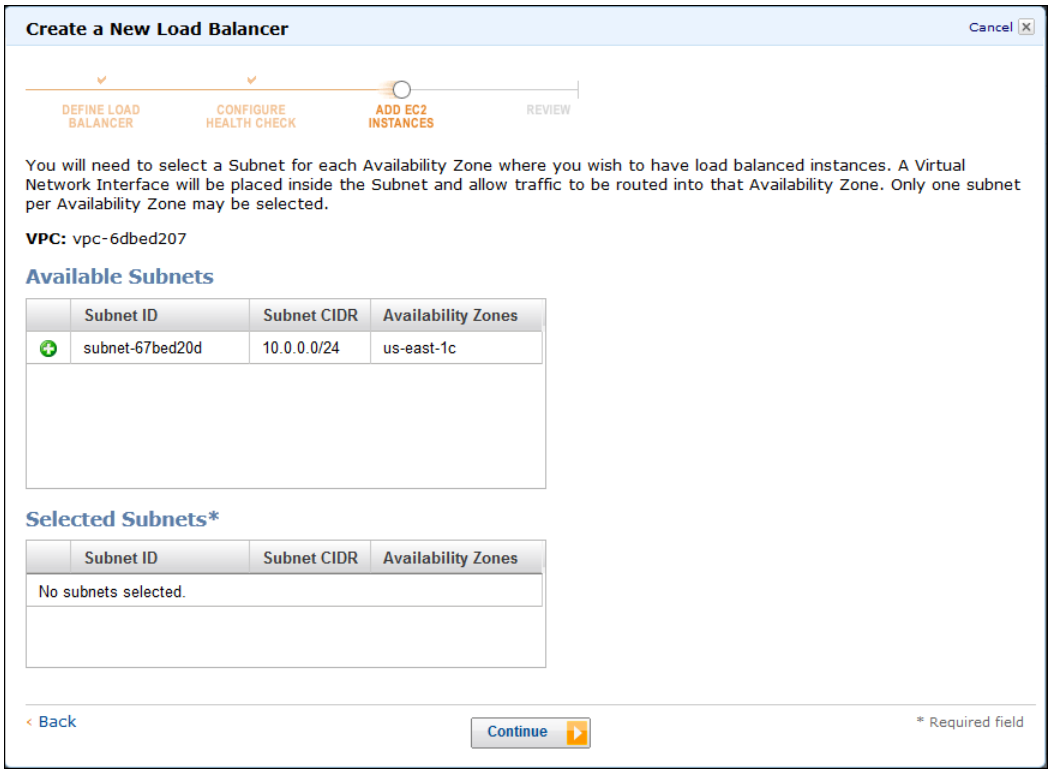
2. Click **Continue** to select the subnet in which you want to launch your load balancer instance.

Launch Your Load Balancer Instance in a Subnet

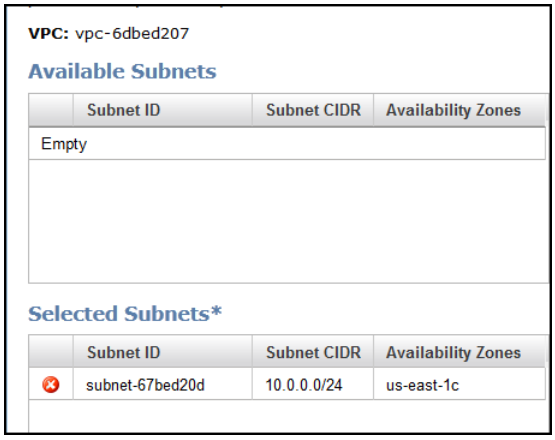
In this step, you select the subnet where you want to launch your load balancer instance.

To select your VPC subnet

1. On the **ADD EC2 Instances** page, in the **Available Subnets** table, click the green button to the left of the subnet into which you want to launch your load balancer instance.



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



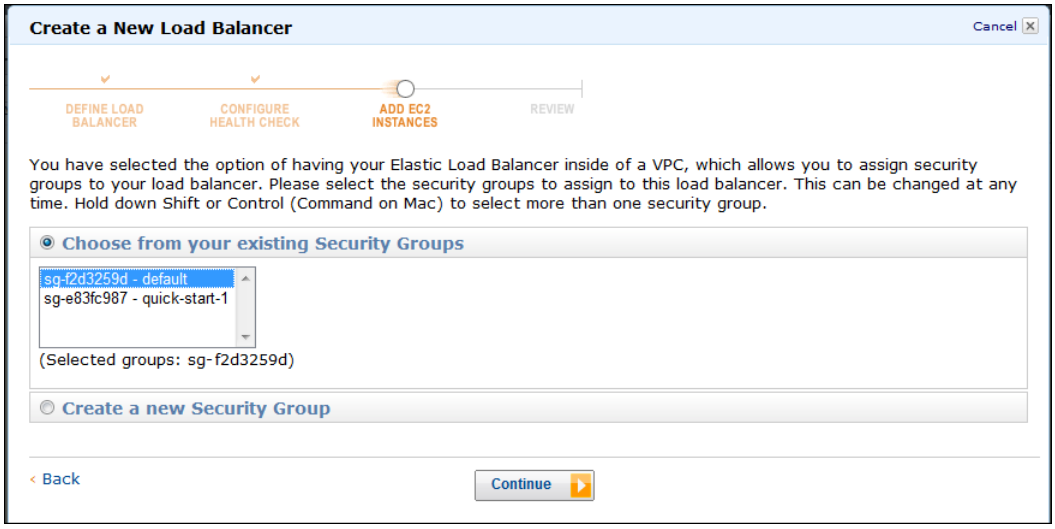
2. Click **Continue** to select security groups to assign to your load balancer.

Assign a Security Group to Your Load Balancer

To select security group for your load balancer

1. 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 tutorial uses the default security group associated with your virtual private cloud.

On the **ADD EC2 INSTANCES** page, select **Choose from your existing Security Groups**, and then select the default security group.



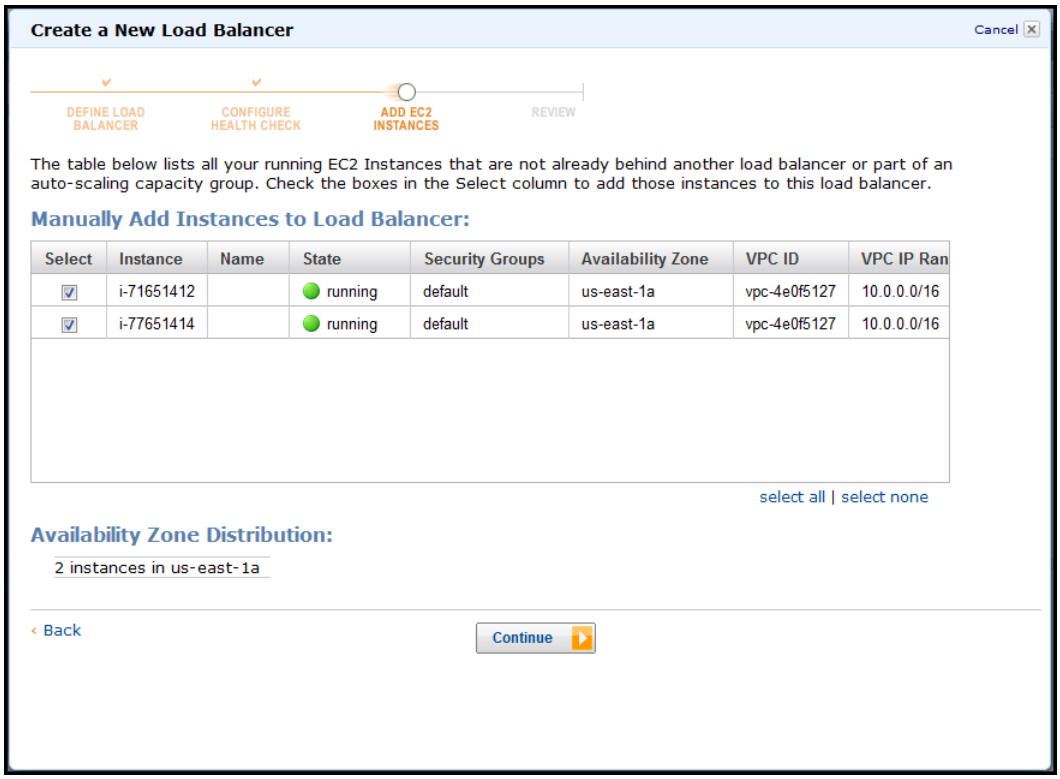
2. Click **Continue** to register EC2 instances with your load balancer.

Register Your Amazon EC2 Instances

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

To register your EC2 instances launched within VPC with your load balancer

1. On the **ADD EC2 INSTANCES** page, in the **Manually Add Instances to LoadBalancer** table, check the boxes in the **Select** column to add instances to your load balancer.



Note
When you register a multi-homed instance (an instance that has an elastic network interface (ENI) 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 ENIs, go to Elastic Network Interfaces.

2. Click **Continue** to review your settings and then create your load balancer.

Review Settings and Create Your Load Balancer

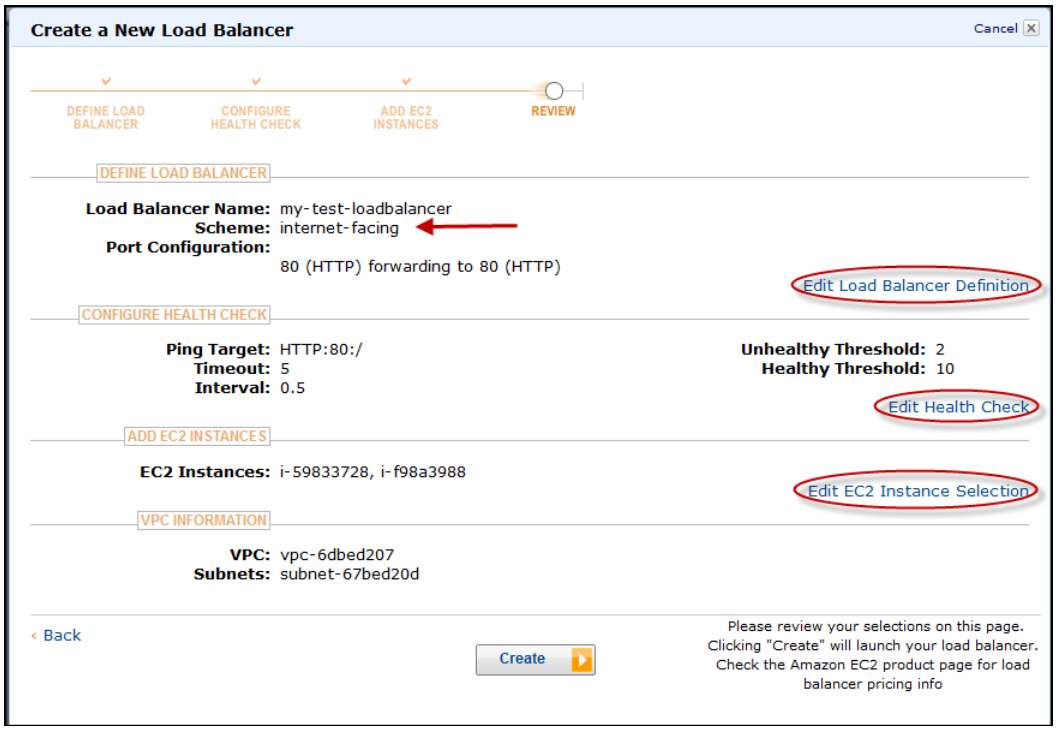
Now that you've registered your EC2 instances with your load balancer it's time to to review the settings you have selected.

To review your settings

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

   The description in the **Scheme** row indicates that the load balancer is `internet-facing`. This means that your load balancer has a publicly resolvable DNS name that resolves to public IP addresses.

   Note
   You can modify some of the settings even after you've created your load balancer. For example, you can modify the port configurations, health check configurations, and add or remove EC2 instances from your load balancer. For more information, see Using Elastic Load Balancing (p. 70).



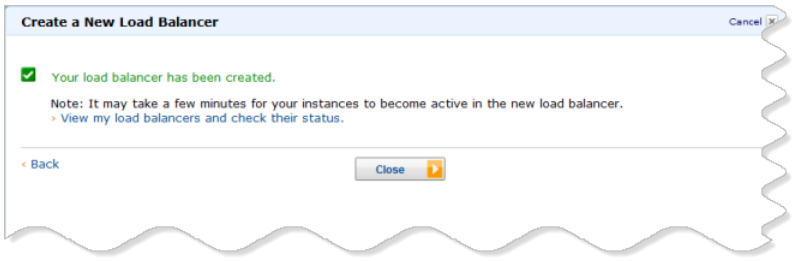
2. Click **Create** to create your load balancer.

Verify Creation of Your Load Balancer

Now that you've created your load balancer, you're ready to verify its settings.

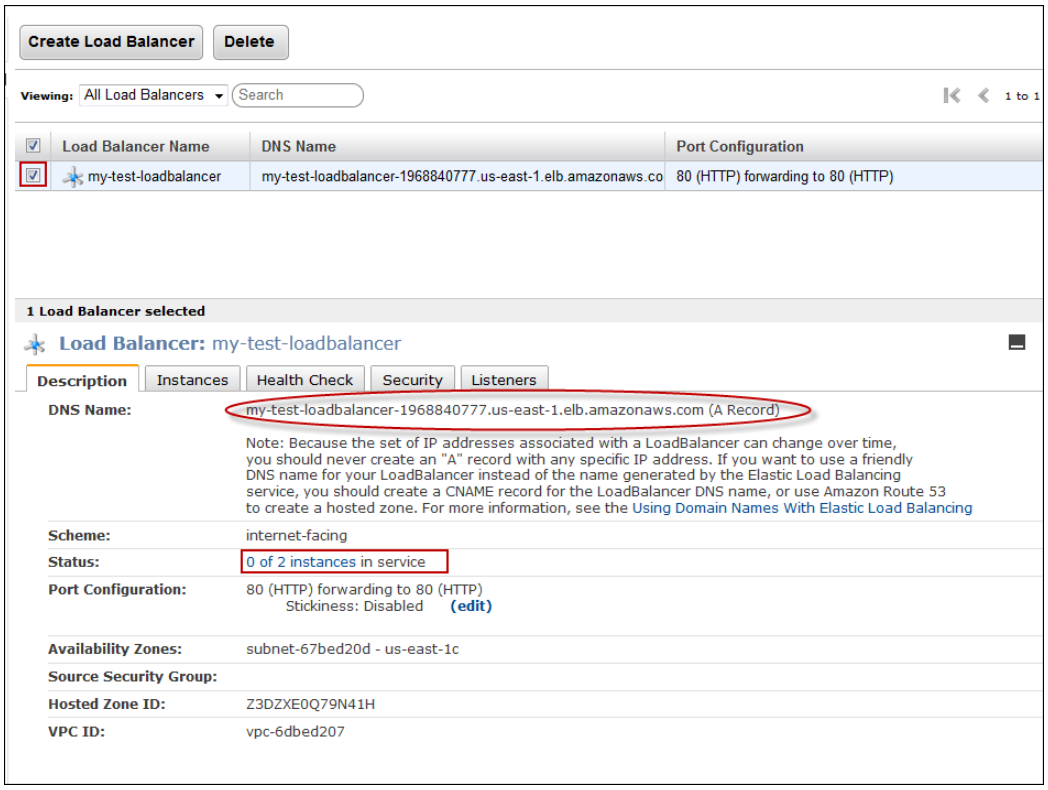
To verify creation of your load balancer

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



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

   Select the check box next to your load balancer.
3. The Load Balancer selected pane displays the description of your load balancer. Verify that the descriptions match your specifications.



If the description in the **Status** row indicates that some of your instances are not in service, its probably because your instances are still in the registration process. For more information, see Troubleshooting Elastic Load Balancing: Registering Instances (p. 175).

4. You can test your load balancer after you've verified that at least one of your EC2 instances is *InService*. 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 you can 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. 67).
- For information on some common Elastic Load Balancing user scenarios for Amazon VPC, and the tasks needed to accomplish efficient distribution of application loads among your Amazon EC2 instances in Amazon VPC, see Deploy Elastic Load Balancing in Amazon VPC (p. 110).
- For information about Elastic Load Balancing usage rates, go to the Elastic Load Balancing product page.

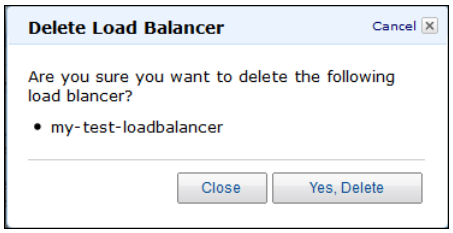
Delete Your Load Balancer

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 consoleGetting Started page, in the **EC2 Dashboard** pane, under **NETWORK & SECURITY**, click **Load Balancers**.
2. Select the check box next to the load balancer you want to delete, and then click **Delete**.

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



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.

Create a Basic Load Balancer in Default VPC

Topics

Amazon Web Services (AWS) supports defining a virtual networking environment in a private, isolated section of the AWS cloud. Within this virtual private cloud (VPC), you can launch Amazon EC2 instances that have a private IP address, a public IP address, a public DNS hostname, and a private DNS hostname. These instances can communicate with the Internet through an Internet gateway. An Internet gateway enables your instances to connect to the Internet through the Amazon EC2 network edge. You can use a load balancer to monitor and route traffic to your EC2 instances launched within this VPC.

An AWS virtual private cloud (Amazon VPC) spans one or more Availability Zones. Each Availability Zone can have multiple subnets. A subnet in Amazon VPC is a subdivision within an Availability Zone defined by a segment of the IP address range of the VPC. Using subnets you can group your instances based on your security and operational needs. A subnet resides entirely within the Availability Zone it was created in.

When you launch an Amazon EC2 instance within Amazon VPC, you associate it with one or more security groups. A security group acts as a firewall that controls the traffic allowed into an instance.

If your AWS account comes with a default VPC, your Amazon EC2 instances are launched within the default VPC, by default. For more information, see Detecting Your Supported Platforms and Whether You Have a Default VPC.

A default VPC combines the benefits of the advanced networking features provided by Amazon VPC platform (EC2-VPC) with the ease of use of the Amazon Elastic Compute Cloud platform (EC2-Classic).

Your default VPC automatically comes with a default subnet in each Availability Zone, an Internet gateway connected to your default VPC, and a default security group associated with your default VPC, among other default configurations. For more information on default VPCs and subnets, see Your Default VPC and Subnets.

When you launch your EC2 instances within a default VPC and you don't specify a subnet, they are automatically launched into a default subnet in your default VPC. By default, we select an Availability Zone for you and launch the instance into the corresponding subnet for that Availability Zone. Alternatively, you can select an Availability Zone for your instance by selecting its corresponding default subnet.

To load balance your EC2 instances launched in default VPC, you must create your load balancers within your default VPC. When you create a load balancer within default VPC, Elastic Load Balancing automatically creates a security group by defining the ports specified for the load balancer to be opened.

The following step-by-step instructions will help you create a basic load balancer in default VPC 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.

- Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
- Confirm that your AWS account supports the default VPC platform. For more information, see Detecting Your Supported Platforms and Whether You Have a Default VPC.

  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.
- Launch your Amazon EC2 instances into your default VPC using default subnets and default security group. For more information, see Launching an EC2 Instance into Your Default VPC.

The following steps outline how to create a basic load balancer in default VPC.

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

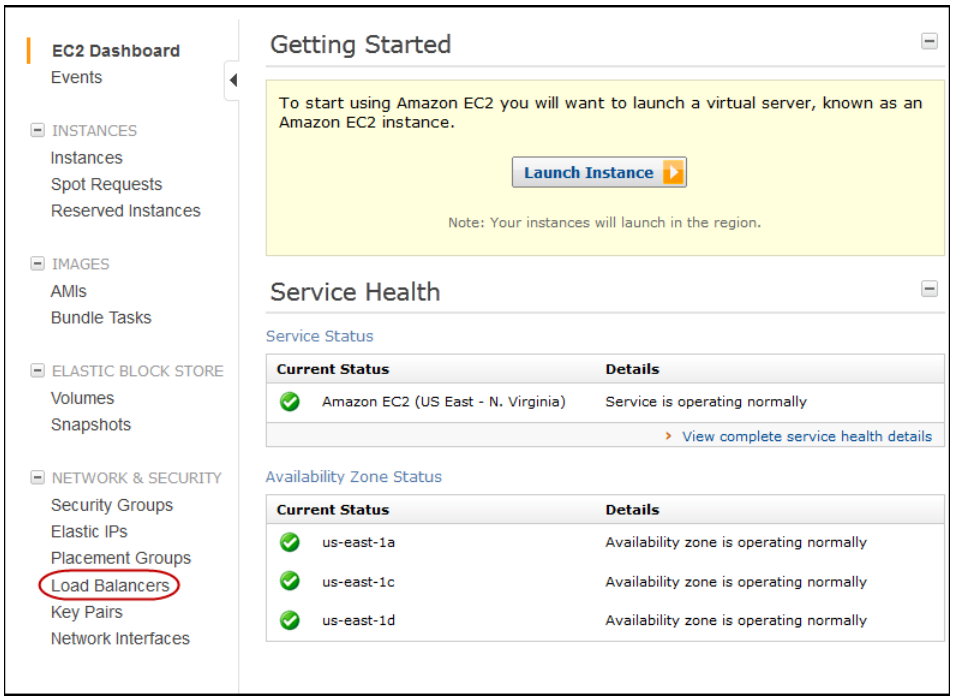
Important
The load balancer you're about to create will be live (and not running in a sandbox). If you are not signed up for the free usage tier, you will incur the standard Elastic Load Balancing usage fees for the load balancer until you terminate it. The total charges will be minimal (typically less 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.

Configure Listeners for Your Load Balancer

To configure listeners for 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. Start the **Create Load Balancer** wizard:

   a. On the Amazon EC2 console Getting Started page, in the **EC2 Dashboard** pane, under **NETWORK & SECURITY**, click **Load Balancers**.



b. Click **Create Load Balancer**.

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

   a. Enter a name for your load balancer (e.g., **`my-test-loadbalancer`**).

   b. Leave the entry in the **Create LB inside:** box set to the default VPC for this tutorial.

   c. Leave **Create an internal load balancer** box blank for this tutorial.

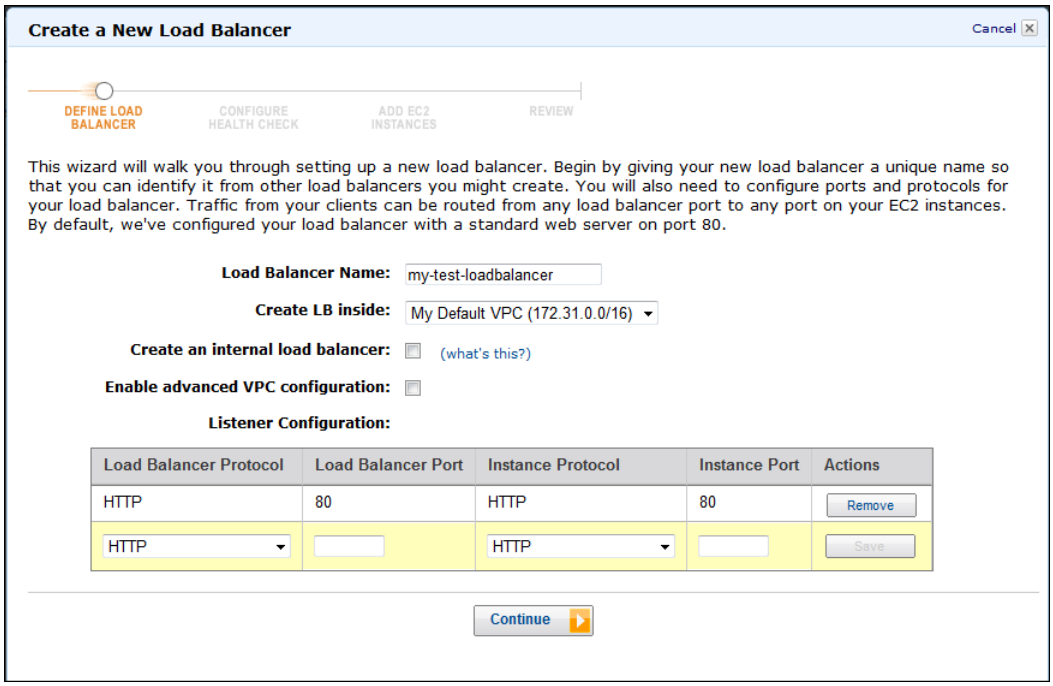
      Note
      In this tutorial, you are creating an Internet-facing load balancer with a publicly resolvable DNS name that resolves to public IP addresses. If you want to create an internal load balancer with a publicly resolvable DNS name that resolves to private IP addresses, see Deploy Elastic Load Balancing in Amazon VPC (p. 110)

   d. Leave the **Enable advanced VPC configuration:** box blank for this tutorial.

      Note
      Advanced VPC configuration option allows you to specify your own subnets. Select this option if you have created your own subnets and want to use them instead of the default subnets.

   e. Leave **Listener Configuration** set to the default value for this tutorial.

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



4. Click **Continue** to configure the health check for your instances.

Configure Health Check for Your Amazon EC2 Instances

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:

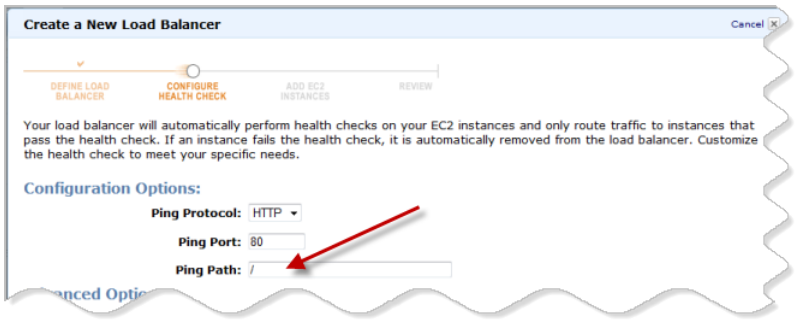
   a. Leave **Ping Protocol** set to its default value of `HTTP`.

   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.

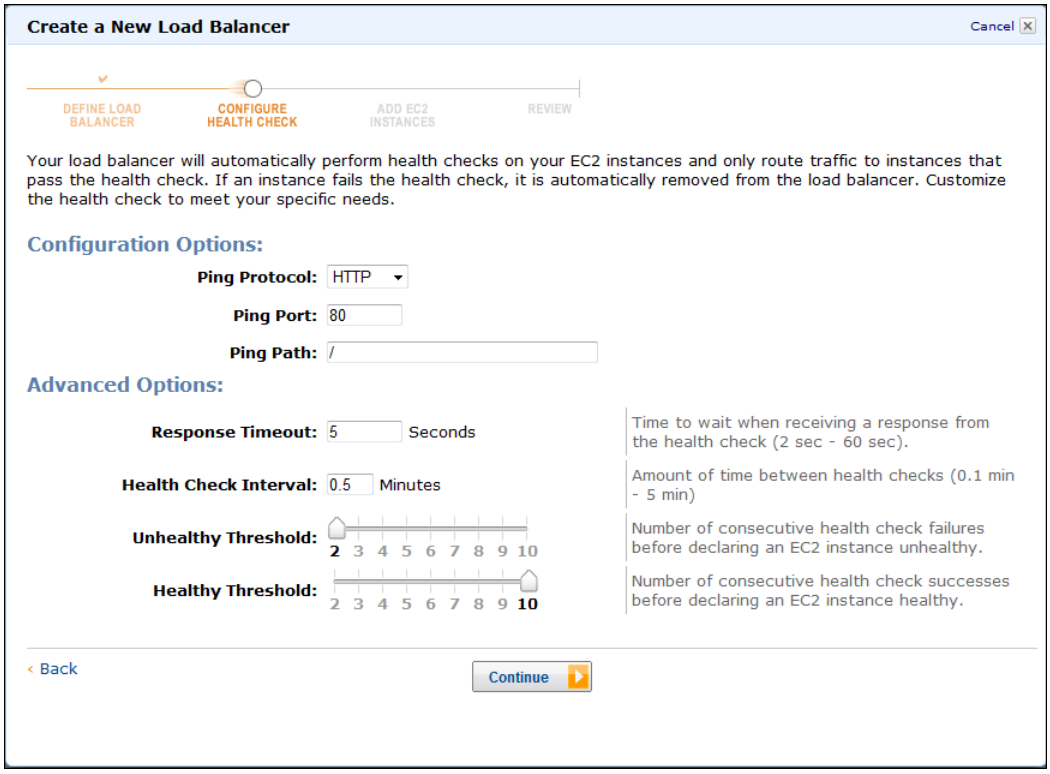
      Important
      Your Amazon EC2 instances must accept incoming traffic on the ping port. This example assumes that each of your instances has a working HTTP server that accepts incoming traffic on port 80.

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



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.



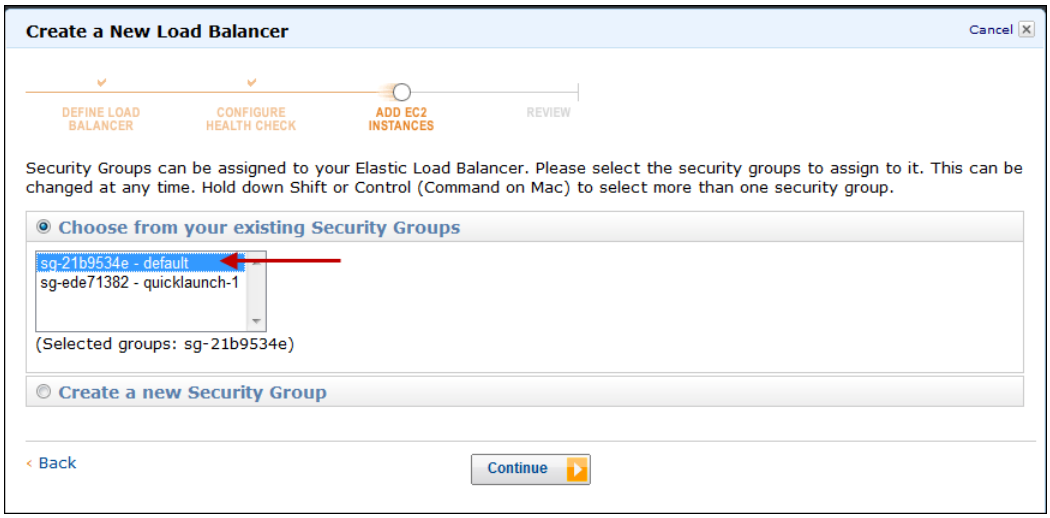
2. Click **Continue** to select security groups to assign to your load balancer.

Assign a Security Group to Your Load Balancer

To select a security group for your load balancer

1. This tutorial uses the default VPC security group associated with your default VPC.

   On the **ADD EC2 INSTANCES** page, select **Choose from your existing Security Groups** and then select the default security group.



If you choose a non-default 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.

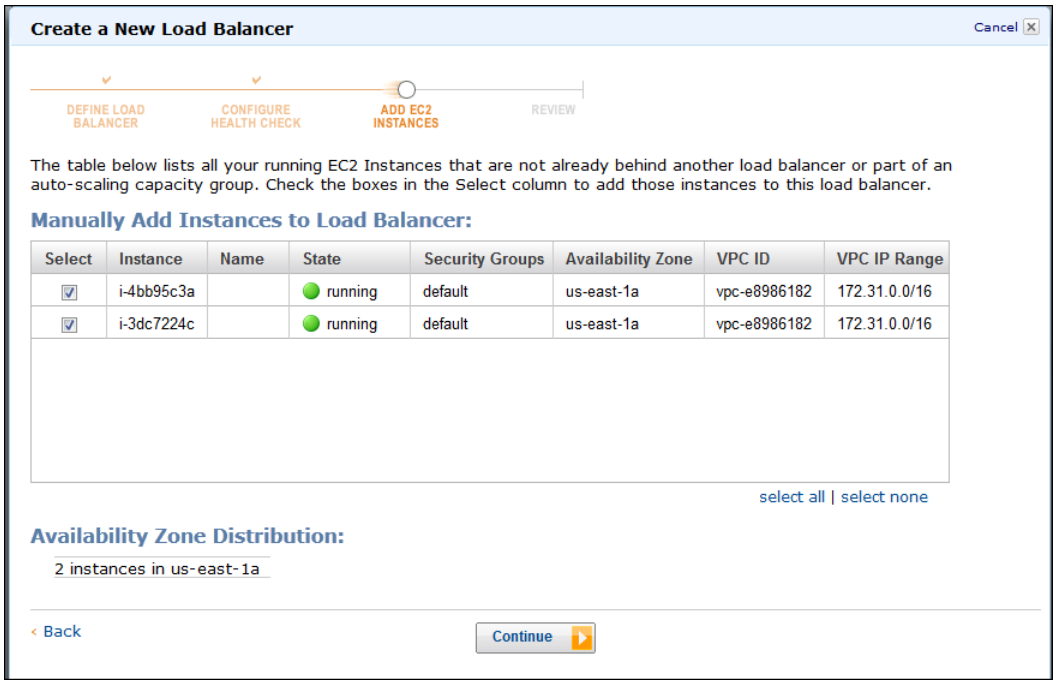
2. Click **Continue** to register your Amazon EC2 instances with your load balancer.

Register Your Amazon EC2 Instances

Now that you've made your configuration choices you're ready to register your EC2 instances with your load balancer.

To register your EC2 instances launched within VPC with your load balancer

1. On the **ADD EC2 Instances** page, in the **Manually Add Instances to LoadBalancer** table, check the boxes in the **Select** column to add instances to your load balancer.



Note
When you register a multi-homed instance (an instance that has an elastic network interface (ENI) 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 ENIs, go to Elastic Network Interfaces.

2. Click **Continue** to review your settings and then create your load balancer.

Review Settings and Create Your Load Balancer

Now that you've registered your EC2 instances with your load balancer it's time to review the settings you have selected.

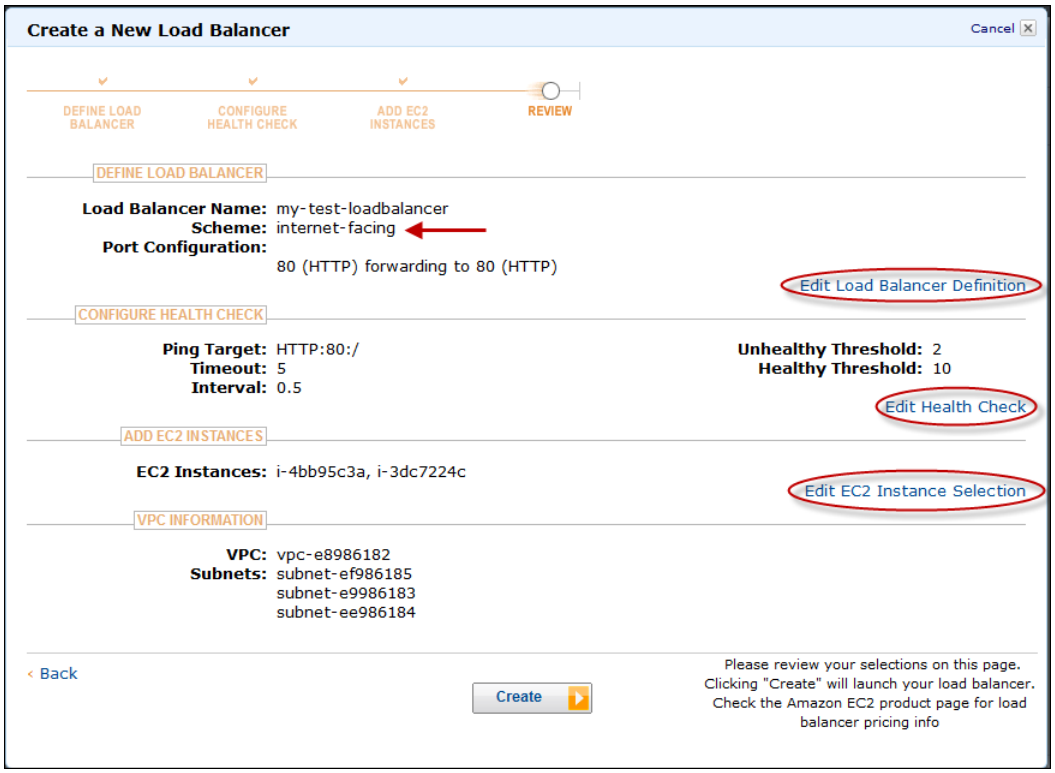
To review your settings

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

   The description in the **Scheme** row indicates that the load balancer is `internet-facing`. This means that your load balancer has a publicly resolvable DNS name that resolves to public IP addresses.

   The VPC and subnets listed under **VPC INFORMATION** are the default VPC and default subnets automatically created for your AWS account.

   Note
   You can modify some of the settings even after you've created your load balancer. For example, you can modify the port configurations, health check configurations, and add or remove EC2 instances from your load balancer. For more information, see Deploy Elastic Load Balancing in Amazon VPC (p. 110).



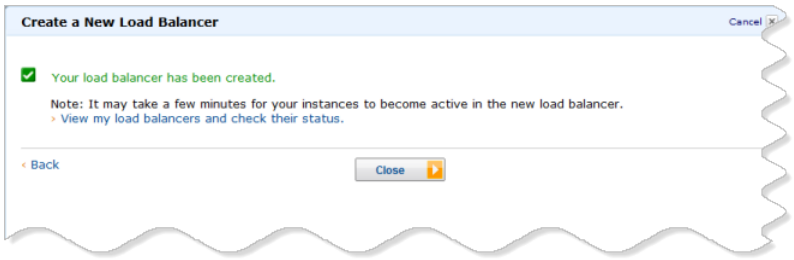
2. Click **Create** to create your load balancer.

Verify Creation of Your Load Balancer

Now that you've created your load balancer, you're ready to verify its settings.

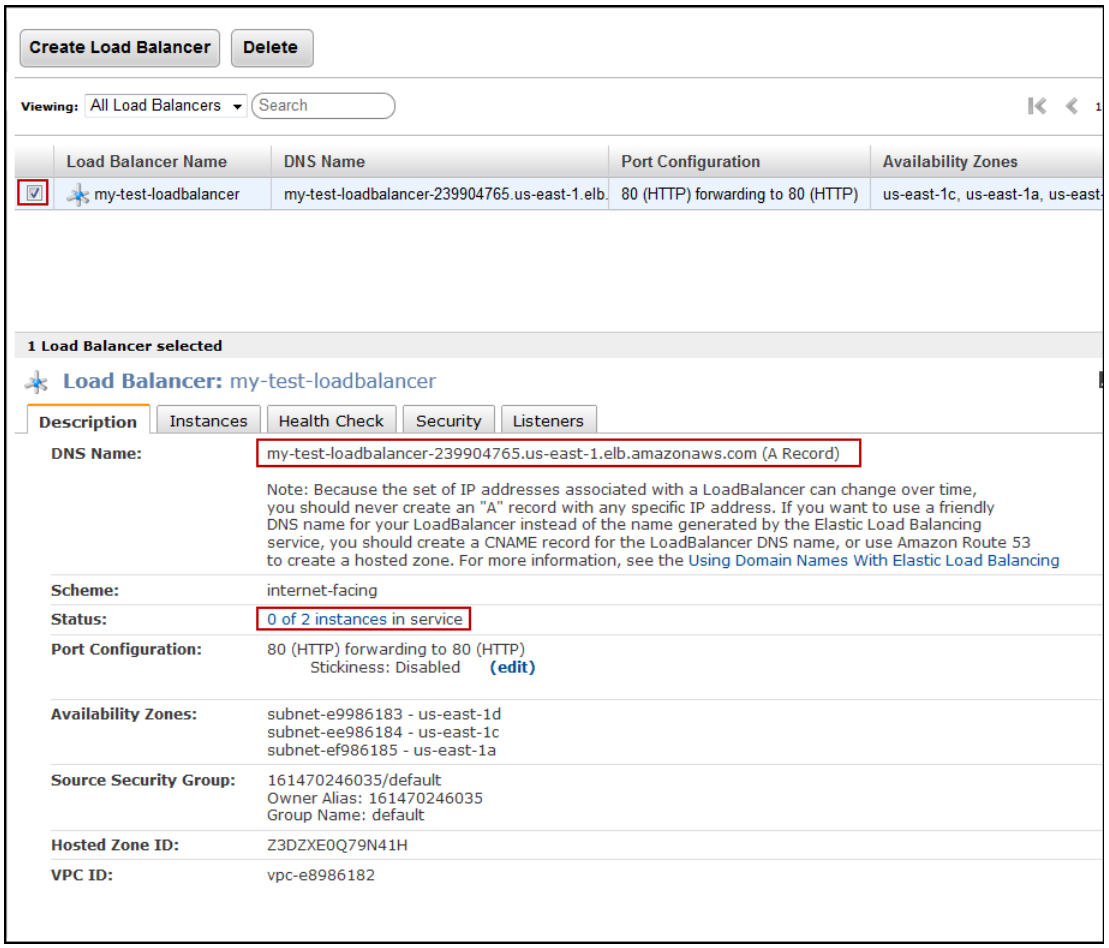
To verify your load balancer is created

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



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

   Select the check box next to your load balancer.
3. The Load Balancer selected pane displays the description of your load balancer. Verify that the descriptions match your specifications.



If the description in the **Status** row indicates that some of your instances are not in service, its probably because your instances are still in the registration process. For more information, see Troubleshooting Elastic Load Balancing: Registering Instances (p. 175).

4. You can test your load balancer after you've verified that at least one of your EC2 instances is *InService*. 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. 67).
- 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 in Amazon VPC, see Deploy Elastic Load Balancing in Amazon VPC (p. 110).

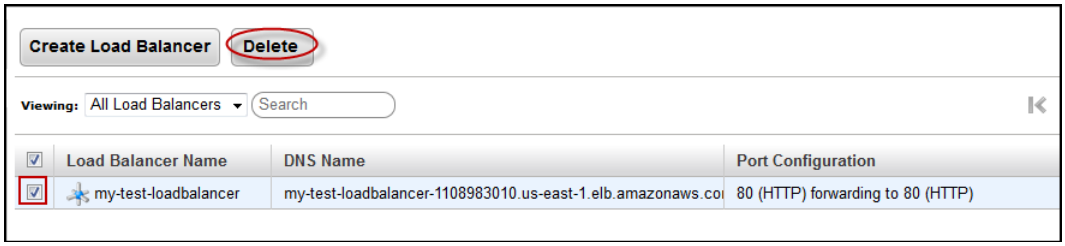
- For information about Elastic Load Balancing usage rates, go to the Elastic Load Balancing product page.

Delete Your Load Balancer

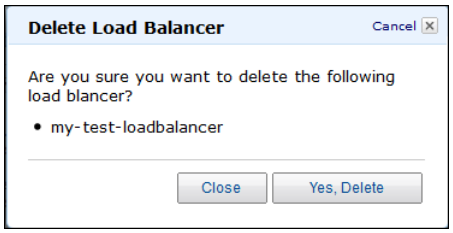
As soon as your load balancer becomes available, you're billed for each hour or partial hour that you keep the load balancer running. After you've decided that you no longer need the load balancer, you can delete it.

To delete your load balancer

1. On the Amazon EC2 console Getting Started page, in the **EC2 Dashboard** pane, under **NETWORK & SECURITY**, click **Load Balancers**.
2. Select the check box next to the load balancer you want to delete, and then click **Delete**.



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



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.

- **AWS Management Console**— The AWS Management Console provides a web-based interface for many AWS products. You can use the console to make requests 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/.
- **Command Line Interface**— 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. 45).
- **Query API**— 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 Use Query Requests to Call Elastic Load Balancing APIs (p. 51).
- **SOAP API**— 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. 60)

- **AWS SDKs**— 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. 63).

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

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=$JAVA_HOME/bin:$PATH
```

   On Windows the syntax is slightly different:

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

   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

When you sign up for an AWS account, we create access credentials for you so that you can make secure requests to AWS. Your access credentials are your AccessKeyId and SecretKey. You must provide these credentials to the command line tools so that they know that the commands that you issue come from your account.

The command line tool reads your 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 create security credentials file on your local system

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
```

Verify if Elastic Load Balancing Command Line Interface(CLI) is Installed

To verify the installation and configuration of Elastic Load Balancing CLI

1. On your Linux or Windows workstation, open a new command prompt.
2. Type the command **`elb-cmd`**.
3. You should see output similar to the following:

```
Command Name                                Description

------------                                -----------
elb-apply-security-groups-to-lb         Apply VPC security groups to Load
Balancer.

elb-associate-route53-hosted-zone       Associate LoadBalancer DNS... a
Route53 resource record.

elb-attach-lb-to-subnets                Attach existing LoadBalancer to
Subnets in VPC.

elb-configure-healthcheck               Configure the parameters f...tered
 with a LoadBalancer.

elb-create-app-cookie-stickiness-policy Create an application-generated
stickiness policy.

elb-create-lb                           Create a new LoadBalancer.

elb-create-lb-cookie-stickiness-policy  Create an LB-generated stickiness
policy.

elb-create-lb-listeners                Create a new LoadBalancer listener.

elb-create-lb-policy                    Create a LoadBalancer poli...ed
```

```
LoadBalancerPolicyType.

elb-delete-lb                              Deletes an existing LoadBalancer.

elb-delete-lb-listeners                    Deletes a listener on an existing
LoadBalancer.

elb-delete-lb-policy                       Delete a LoadBalancer policy.

elb-deregister-instances-from-lb           Deregisters instances from a Load
Balancer.

elb-describe-instance-health               Describes the state of instances.

elb-describe-lb-policies                  Describes the details of LoadBalancer
 Policies.

elb-describe-lb-policy-types              Describes the details of LoadBalancer
 PolicyTypes.

elb-describe-lbs                           Describes the state and properties
 of LoadBalancers.

elb-detach-lb-from-subnets                 Detach existing LoadBalancer from
Subnets in VPC.

elb-disable-zones-for-lb                   Remove availability zones from an
LoadBalancer.

elb-disassociate-route53-hosted-zone       Disassociate LoadBalancer ... a
Route53 resource record.

elb-enable-zones-for-lb                    Add availability zones to existing
 LoadBalancer.

elb-register-instances-with-lb            Registers instances to a LoadBalancer.

elb-set-lb-listener-ssl-cert               Set the SSL Certificate fo...ecified
 LoadBalancer port.

elb-set-lb-policies-for-backend-server  Set LoadBalancer policies for backend
 servers.

elb-set-lb-policies-of-listener            Set LoadBalancer policies for a
port.

version                                    Prints the version of the CLI tool
 and the API.

    For help on a specific command, type '<commandname> --help'
```

This completes your installation and configuration of the Elastic Load Balancing command line interface tool. 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.

Use Query Requests to Call Elastic Load Balancing APIs

Topics

Query requests are HTTP or HTTPS requests that use the HTTP verb GET or POST and a Query parameter named *Action* or *Operation* that specifies the API you are calling. Action is used throughout this documentation, although Operation is also supported for backward compatibility with other Amazon Web Services (AWS) Query APIs.

Calling the API using a Query request is the most direct way to access the web service, but requires that your application handle low-level details such as generating the hash to sign the request, and error handling. The benefit of calling the service using a Query request is that you are assured of having access to the complete functionality of the API.

> **Note**
> The Query interface used by AWS is similar to REST, but does not adhere completely to the REST principles.

Signing Query Requests

Query requests travel over the Internet using either HTTP or HTTPS, and are vulnerable to being intercepted and altered in transit. To prevent this and ensure that the incoming request is both from a valid AWS account and unaltered, AWS requires all requests to be signed.

To sign a Query request, you calculate a digital signature using a cryptographic hash function over the text of the request and your AWS secret key. A cryptographic hash is a one-way function that returns unique results based on the input.

When Elastic Load Balancing receives the request, it re-calculates the signature using the request text and the secret key that matches the AWS access key in the request. If the two signatures match, Auto Scaling knows that the query has not been altered and that the request originated from your account. This is one reason why it is important to safeguard your private key. Any malicious user who obtains it would be able to make AWS calls, and incur charges, on your account.

For additional security, you should transmit your requests using Secure Sockets Layer (SSL) by using HTTPS. SSL encrypts the transmission, protecting your request or the response from being viewed in transit. For more information about securing your Query requests, see Making Secure Requests to Amazon Web Services.

The signature format that AWS uses has been refined over time to increase security and ease of use. Elastic Load Balancing supports Signature Version 2 and Signature Version 4. If you are creating new applications that use Elastic Load Balancing, then we recommend using Signature Version 4 for signing your query requests.

For information on how to create the signature using Signature Version 4, see Signature Version 4 Signing Process in the *AWS General Reference*.

The following sections describe the steps needed to create a query request using Signature Version 2.

Structure of an Elastic Load Balancing Query Request

Elastic Load Balancing requires that each HTTP or HTTPS query request formatted for Signature Version 2 contain the following:

- **Endpoint—**Also known as the host part of an HTTP request. This is the DNS name of the computer to which you send the Query request. This is different for each AWS Region. For the complete list of endpoints supported by a web service, see Regions and Endpoints in the *Amazon Web Services General Reference*.
  The endpoint, `elasticloadbalancing.amazonaws.com`, shown in the example below, is the default endpoint and maps to the Region `us-east-1`.
- **Action—**Specifies the action that you want a web service to perform.
  This value determines the parameters that are used in the request. For descriptions of Elastic Load Balancing actions and their parameters, see Elastic Load Balancing API Reference.
  The action in the example below is `DescribeLoadBalancers`, which causes a web service to return details about one or more load balancers.
- **Required and optional parameters—**Each action in a web service has a set of required and optional parameters that define the API call. For a list of parameters that must be included in every a web service action, see Common Parameters. For details about the parameters for a specific action, see the entry for the specific Actions in the *Elastic Load Balancing API Reference*.

  The following common parameters are required to authenticate a query request:
  - **AccessKeyId—**A value distributed by AWS when you sign up for an AWS Account. For information on retrieving the AccessId for your AWS account, see Get Your Access Key ID and Secret Access Key
  - **Timestamp—**This is the time at which you make the request. Including this in the Query request helps prevent third parties from intercepting your request and re-submitting to a web service.
  - **SignatureVersion—**The version of the AWS signature protocol you're using. Elastic Load Balancing supports signature version 2 and signature version 4.
  - **SignatureMethod—**The hash-based protocol you are using to calculate the signature. This can be either HMAC-SHA1 or HMAC-SHA256 for version 2 AWS signatures.
  - **Signature—**This is a calculated value that ensures the signature is valid and has not been tampered with in transit.

Process for Signing Query Request Using Signature Verion 2

Before you can sign the Query request, you must put the request into a completely unambiguous format. This is needed because there are different—and yet correct—ways to format a Query request, but the different variations would result in different HMAC signatures. Putting the request into an unambiguous, canonical, format before signing it ensures that your application and a web service will calculate the same signature for a given request.

The unambigous string to sign is built up by concatenating the Query request components together as follows. As an example, let's generate the string to sign for the following call to the `DescribeLoadBalancers` API.

```
https://elasticloadbalancing.amazonaws.com?Action=DescribeLoadBalancers
&Version=2011-01-01
&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
&SignatureVersion=2
```

```
&SignatureMethod=HmacSHA256
&Timestamp=2011-10-03T15%3A19%3A30
```

To create the string to sign (Signature Version 2)

1. Start with the request method (either GET or POST), followed by a newline character. (In the following, for human readability, the newline character is represented as `\n`.)

   ```
   GET\n
   ```

2. Add the HTTP host header in lowercase, followed by a newline character. The port information is omitted if it is the standard port for the protocol (port 80 for HTTP and port 443 for HTTPS), but included if it is a non-standard port.

   ```
   elasticloadbalancing.amazonaws.com\n
   ```

3. Add the URL-encoded version of the absolute path component of the URI (this is everything between the HTTP host header to the question mark character (?) that begins the query string parameters) followed by a newline character. If the absolute path is empty, use a forward slash (/).

   ```
   /\n
   ```

4. Add the query string components (the name-value pairs, not including the initial question mark (?) as UTF-8 characters which are URL encoded per RFC 3986 (hexadecimal characters must be uppercased) and sorted using lexicographic byte ordering. Lexicographic byte ordering is case sensitive.

   Separate parameter names from their values with the equal sign character (=) (ASCII character 61), even if the value is empty. Separate pairs of parameter and values with the ampersand character (&) (ASCII code 38).

   Some API actions 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 contain only one parameter. For example, a query parameter list for load balancer names looks like this:

   ```
   &LoadBalancerNames.member.1=my-test-loadbalancer-1
   &LoadBalancerNames.member.2=my-test-loadbalancer-2
   ```

   All reserved characters *must* be escaped. All unreserved characters *must not* be escaped. Concatenate the parameters and their values to make one long string with no spaces between them. Spaces within a parameter value, are allowed, but must be URL encoded as %20. In the concatenated string, period characters (.) are not escaped. RFC 3986 considers the period character an unreserved character, and thus it is not URL encoded.

   Note
   RFC 3986 does not specify what happens with ASCII control characters, extended UTF-8 characters, and other characters reserved by RFC 1738. Since any values may be passed into a string value, these other characters should be percent encoded as %XY where X and Y are uppercase hex characters. Extended UTF-8 characters take the form %XY%ZA... (this handles multi-bytes). The space character should be represented as '%20'. Spaces should *not* be encoded as the plus sign (+) as this will cause as error.

The following example shows the query string components of a call to the Elastic Load Balancing API `DescribeLoadBalancers`, processed as described above.

```
AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Action=DescribeLoadBalancers&Signature
Method=HmacSHA256&SignatureVersion=2&Timestamp=2011-10-03T15%3A19%3A30&Ver
sion=2012-06-01
```

5. The string to sign for the call to `DescribeLoadBalancers` takes the following form:

```
GET\n
elasticloadbalancing.amazonaws.com%0A
/\n
AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Action=DescribeLoadBalancers&Signature
Method=HmacSHA256&SignatureVersion=2&Timestamp=2011-10-03T15%3A19%3A30&Ver
sion=2012-06-01
```

Calculating the AWS Signature Version 2

After you've created the canonical string as described in Process for Signing Query Request Using Signature Verion 2 (p. 52), you calculate the signature by creating a hash-based message authentication code (HMAC) using either the HMAC-SHA1 or HMAC-SHA256 protocols. The HMAC-SHA256 protocol is preferred.

You then add the value returned to the Query request as a signature parameter, as shown below. You can then use the signed request in an HTTP or HTTPS call. A web service will then return the results of the call formatted as a response.

```
https://elasticloadbalancing.amazonaws.com?AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Ac
tion=DescribeLoadBalancers&SignatureMethod=HmacSHA256&SignatureVer
sion=2&Timestamp=2011-10-03T15%3A19%3A30&Version=2011-01-01&Signa
ture=lptk88A0LEP2KfwM3ima33DUjY0e%2FyfF7YfitJ%2FQw6I%3D
```

The AWS SDKs offer functions to generate Query request signatures. To see an example using the AWS SDK for Java, go to Using the Java SDK to Sign a Query Request (p. 58).

Example Query Request and Response Structures

Example Query Request Structure

In the Elastic Load Balancing documentation, we leave the example GET requests unencoded to make them easier to read. To make the GET examples even easier to read, Elastic Load Balancing documentation presents them in the following parsed format.

```
https://elasticloadbalancing.amazonaws.com/?LoadBalancerNames.member.1=my-test-
loadbalancer-1
&MaxRecords=20
&Action=DescribeLoadBalancers
&AWSAccessKeyId=AccessKeyID
```

```
&Timestamp=2012-01-28T21%3A49%3A59.000Z
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Signature=calculated value
```

The first line represents the *endpoint* of the request. This is the resource the request acts on.

After the endpoint is a question mark (?), which separates the endpoint from the parameters. Each parameter is separated by an ampersand (&).

The Action parameter indicates the action to perform (for a list of the actions, see Elastic Load Balancing API Reference). For a list of the other parameters that are common to all Query requests, see Common Parameters in the *Elastic Load Balancing API Reference*.

In the example Query requests we present in the Elastic Load Balancing documentation, we omit the parameters related to authentication to make it easier to focus on the ones relevant to the particular action. We replace them with the following literal string to remind you that a real request includes the parameters: `&AUTHPARAMS`.

The above example without the authentication parameters listed will look similar to the following example:

```
https://elasticloadbalancing.amazonaws.com/?LoadBalancerNames.member.1=my-test-
loadbalancer
&LoadBalancerNames.member.2=www-example-com
&MaxRecords=20
&Action=DescribeLoadBalancers
&AUTHPARAMS
```

Example Query Response Structure

In response to a Query request, the Elastic Load Balancing returns an XML data structure that conforms to an XML schema defined as part of the Elastic Load Balancing WSDL. The structure of an XML response is specific to the associated request. In general, the response data types are named according to the operation performed and whether the data type is a container (can have children). Examples of containers include `<LoadBalancers>` (see the example that follows). Member elements are children of containers, and their contents vary according to the container's role.

In every response from AWS, you will see the element `ResponseMetadata`, which contains a string element called `RequestId`. This is simply a unique identifier that AWS assigns to this request for tracking and troubleshooting purposes.

If the query request is successful, you'll get a response similar to the following example:

```
<DescribeLoadBalancersResponse xmlns="http://elasticloadbalancing.amazon
aws.com/doc/2012-06-01/">
  <DescribeLoadBalancersResult>
    <LoadBalancerDescriptions>
      <member>
        <SecurityGroups/>
        <CreatedTime>2013-01-24T20:51:35.710Z</CreatedTime>
        <LoadBalancerName>my-test-loadbalancer</LoadBalancerName>
        <HealthCheck>
          <Interval>30</Interval>
          <Target>HTTP:80/</Target>
          <HealthyThreshold>10</HealthyThreshold>
          <Timeout>5</Timeout>
```

```
          <UnhealthyThreshold>2</UnhealthyThreshold>
        </HealthCheck>
        <ListenerDescriptions>
          <member>
            <PolicyNames/>
            <Listener>
              <Protocol>HTTP</Protocol>
              <LoadBalancerPort>80</LoadBalancerPort>
              <InstanceProtocol>HTTP</InstanceProtocol>
              <InstancePort>80</InstancePort>
            </Listener>
          </member>
          <member>
            <PolicyNames>
            <member>AWSConsole-SSLNegotiationPolicy-my-test-loadbalancer</mem
ber>
            </PolicyNames>
            <Listener>
              <Protocol>HTTPS</Protocol>
              <LoadBalancerPort>443</LoadBalancerPort>
              <InstanceProtocol>HTTP</InstanceProtocol>
              <SSLCertificateId>arn:aws:iam::803981987763:server-certific
ate/scert</SSLCertificateId>
              <InstancePort>80</InstancePort>
            </Listener>
          </member>
        </ListenerDescriptions>
        <Instances>
          <member>
            <InstanceId>i-48bb5d38</InstanceId>
          </member>
          <member>
            <InstanceId>i-78bc5a08</InstanceId>
          </member>
          <member>
            <InstanceId>i-98e204e8</InstanceId>
          </member>
          <member>
            <InstanceId>i-ccbb5dbc</InstanceId>
          </member>
        </Instances>
        <Policies>
          <AppCookieStickinessPolicies/>
          <OtherPolicies>
           <member>AWSConsole-SSLNegotiationPolicy-my-test-loadbalancer</member>

          </OtherPolicies>
          <LBCookieStickinessPolicies/>
        </Policies>
        <AvailabilityZones>
          <member>us-east-1e</member>
        </AvailabilityZones>
        <CanonicalHostedZoneName>my-test-loadbalancer-1086370712.us-east-
1.elb.amazonaws.com</CanonicalHostedZoneName>
        <CanonicalHostedZoneNameID>Z3DZXE0Q79N41H</CanonicalHostedZoneNameID>
        <Scheme>internet-facing</Scheme>
        <SourceSecurityGroup>
          <OwnerAlias>amazon-elb</OwnerAlias>
```

```
          <GroupName>amazon-elb-sg</GroupName>
        </SourceSecurityGroup>
       <DNSName>my-test-loadbalancer-1086370712.us-east-1.elb.amazonaws.com</DNS
Name>
        <BackendServerDescriptions/>
        <Subnets/>
      </member>
      <member>
        <SecurityGroups/>
        <CreatedTime>2013-02-08T00:17:08.810Z</CreatedTime>
        <LoadBalancerName>www-example-com</LoadBalancerName>
        <HealthCheck>
          <Interval>30</Interval>
          <Target>HTTP:80/index.html</Target>
          <HealthyThreshold>10</HealthyThreshold>
          <Timeout>5</Timeout>
          <UnhealthyThreshold>2</UnhealthyThreshold>
        </HealthCheck>
        <ListenerDescriptions>
          <member>
            <PolicyNames/>
            <Listener>
              <Protocol>HTTP</Protocol>
              <LoadBalancerPort>80</LoadBalancerPort>
              <InstanceProtocol>HTTP</InstanceProtocol>
              <InstancePort>80</InstancePort>
            </Listener>
          </member>
        </ListenerDescriptions>
        <Instances/>
        <Policies>
          <AppCookieStickinessPolicies/>
          <OtherPolicies/>
          <LBCookieStickinessPolicies/>
        </Policies>
        <AvailabilityZones>
          <member>us-east-1a</member>
        </AvailabilityZones>
        <CanonicalHostedZoneName>www-example-com-512613397.us-east-
1.elb.amazonaws.com</CanonicalHostedZoneName>
        <CanonicalHostedZoneNameID>Z3DZXE0Q79N41H</CanonicalHostedZoneNameID>
        <Scheme>internet-facing</Scheme>
        <SourceSecurityGroup>
          <OwnerAlias>amazon-elb</OwnerAlias>
          <GroupName>amazon-elb-sg</GroupName>
        </SourceSecurityGroup>
       <DNSName>www-example-com-512613397.us-east-1.elb.amazonaws.com</DNSName>

        <BackendServerDescriptions/>
        <Subnets/>
      </member>
    </LoadBalancerDescriptions>
  </DescribeLoadBalancersResult>
  <ResponseMetadata>
    <RequestId>6fbe234c-80f9-11e2-a17f-9148EXAMPLE</RequestId>
  </ResponseMetadata>
</DescribeLoadBalancersResponse>
```

Troubleshooting Request Signatures Version 2

This section describes some error codes you might see when you are initially developing code to generate the signature to sign Query requests.

SignatureDoesNotMatch Signing Error in a web service

The following error response is returned when a web service attempts to validate the request signature by recalculating the signature value and generates a value that does not match the signature you appended to the request. This can occur because the request was altered between the time you sent it and the time it reached a web service endpoint (this is the case the signature is designed to detect) or because the signature was calculated improperly. A common cause of the error message below is not properly creating the string to sign, such as forgetting to URL encode characters such as the colon (:) and the forward slash (/) in Amazon S3 bucket names.

```
<ErrorResponse xmlns="http://elasticmapreduce.amazonaws.com/doc/2009-03-31">
  <Error>
    <Type>Sender</Type>
    <Code>SignatureDoesNotMatch</Code>
    <Message>The request signature we calculated does not match the signature
you provided.
    Check your AWS Secret Access Key and signing method.
    Consult the service documentation for details.</Message>
  </Error>
  <RequestId>7589637b-e4b0-11e0-95d9-639f87241c66</RequestId>
</ErrorResponse>
```

IncompleteSignature Signing Error in a web service

The following error indicates that signature is missing information or has been improperly formed.

```
<ErrorResponse xmlns="http://elasticmapreduce.amazonaws.com/doc/2009-03-31">
  <Error>
    <Type>Sender</Type>
    <Code>IncompleteSignature</Code>
   <Message>Request must contain a signature that conforms to AWS standards</Mes
sage>
  </Error>
  <RequestId>7146d0dd-e48e-11e0-a276-bd10ea0cbb74</RequestId>
</ErrorResponse>
```

Using the Java SDK to Sign a Query Request

The following example uses the amazon.webservices.common package of the AWS SDK for Java to generate an AWS Signature Version 2 Query request signature. To do so, it creates an RFC 2104-compliant HMAC signature. For more information about HMAC, go to HMAC: Keyed-Hashing for Message Authentication.

Note

Java is used in this case as a sample implementation. You can use the programming language of your choice to implement the HMAC algorithm to sign Query requests.

```
import java.security.SignatureException;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import com.amazonaws.util.*;

/**
* This class defines common routines for generating
* authentication signatures for AWS Platform requests.
*/
public class Signature {
    private static final String HMAC_SHA256_ALGORITHM = "HmacSHA256";

    /**
     * Computes RFC 2104-compliant HMAC signature.
     * * @param data
     * The signed data.
     * @param key
     * The signing key.
     * @return
     * The Base64-encoded RFC 2104-compliant HMAC signature.
     * @throws
     * java.security.SignatureException when signature generation fails
     */
    public static String calculateRFC2104HMAC(String data, String key)
    throws java.security.SignatureException
    {
        String result;
        try {

            // Get an hmac_sha256 key from the raw key bytes.
            SecretKeySpec signingKey = new SecretKeySpec(key.getBytes("UTF8"),
 HMAC_SHA256_ALGORITHM);

            // Get an hmac_sha256 Mac instance and initialize with the signing
 key.
            Mac mac = Mac.getInstance(HMAC_SHA256_ALGORITHM);
            mac.init(signingKey);

            // Compute the hmac on input data bytes.
            byte[] rawHmac = mac.doFinal(data.getBytes("UTF8"));

            // Base64-encode the hmac by using the utiliy in the SDK
            result = BinaryUtils.toBase64(rawHmac);


        } catch (Exception e) {
            throw new SignatureException("Failed to generate HMAC : " + e.get
Message());
        }
        return result;
    }
}
```

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. 63).

> **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

Configure Listeners for Your Load Balancer

A typical web application communication goes through layers of hardware and software. Each layer provides a specific communication function. The control over the communication function is passed from one layer to the next, in sequence. The Open System Interconnection (OSI) defines a model framework for implementing a standard format for communication, called a *protocol*, in these layers. For detailed information about the layers of the OSI model, go to http://en.wikipedia.org/wiki/OSI_model.

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

Elastic Load Balancing supports the load balancing of applications using HTTP, HTTPS , TCP, and Secure Sockets Layer (SSL) 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 choose HTTP for your front-end connection, the back-end connection to the instance can be HTTP or HTTPS.

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

Before you start using Elastic Load Balancing, you'll have to configure the *listeners* for your load balancer. Listener is a process that listens for client connection requests. It is configured with a protocol and a port number for front-end (Load Balancer) and back-end (Back-end instance) connections.

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 (Layer 4) with Elastic Load Balancing

When you use TCP for both front-end and back-end connections, your load balancer will forward the request to the back-end instances without modification to the headers. This configuration will also not insert cookies for session stickiness or the X-Forwarded-* headers.

If you use SSL (secure TCP) for your front-end connection, 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 (Layer 7) with Elastic Load Balancing

When you use HTTP (layer 7) for both front-end and back-end connections, your load balancer parses the headers in the request and terminates the connection before re-sending the request to the registered instance(s). This is the default configuration provided by Elastic Load Balancing.

If you use HTTPS (secure HTTP) for your front-end listener, you must install an SSL server certificate on your load balancer. The load balancer uses the certificate to terminate and then decrypt requests before sending them to the back-end instances. Additionally, when you use HTTPS, the load balancer inserts or updates the X-Forwarded-* headers and can insert or update cookies for sticky sessions. For more information on X-Forwarded-* headers, see X-Forwarded-For (p. 9). 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. 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 Creating a HTTPS/SSL Load Balancer (p. 73).

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. 138).

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. Depending on the number of nodes, your ciphers and protocols should take less than 30 seconds to take effect.

For information on how to configure the cipher settings, see Creating a HTTPS/SSL Load Balancer (p. 73).

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.

For a quick reference to the listener configurations supported by Elastic Load Balancing, see Elastic Load Balancing Listener Configurations Quick Reference (p. 67).

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 (Layer 7)

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 (p. 8). • Supports X-Forwarded-For (p. 9) header. • Requires SSL server certificate installed on the load balancer.
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 (p. 8). • Supports X-Forwarded-For (p. 9) header. • 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 (Layer 4)

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.

Use Case	Front-End Protocol	Front-End Optional Configuration	Back-End Protocol	Back-End Optional Configuration	Notes
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 and on the registered instances.

Where Do I Go From Here?

For configuring listeners:

- 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. 14).
- For information on how to set up an HTTPS/SSL load balancer with cipher settings and back-end authentication, see Create a Load Balancer with SSL Cipher Settings and Back-End Server Authentication (p. 73).
- For information on how to set up a basic HTTP/TCP load balancer within Amazon VPC, see Deploy Elastic Load Balancing in Amazon VPC (p. 110).
- For information on adding a listener to an existing load balancer, see Adding a Listener to your Load Balancer (p. 128).
- For information on deleting a listener from an existing load balancer, see Delete a Listener from Your Load Balancer (p. 133).

For performing Elastic Load Balancing tasks:

- 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. 44).
- For information on using the various features supported by Elastic Load Balancing, see Using Elastic Load Balancing (p. 70).
- 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:

- Determined the region in which your EC2 instances will run.

  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. 45). 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. 67).
- 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 Deploy Elastic Load Balancing in Amazon EC2-Classic (p. 72).

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 Deploy Elastic Load Balancing in Amazon VPC (p. 110).

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-Register and Register Amazon EC2 Instances (p. 136).

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. 14).
- For information on creating HTTPS/SSL load balancer, see Creating a HTTPS/SSL Load Balancer (p. 73).
- For information on adding listener to your existing load balancer, see Adding a Listener to your Load Balancer (p. 128).
- For information on deleting a Listener from your load balancer, see Delete a Listener from Your Load Balancer (p. 133).

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 Update an SSL Certificate for a Load Balancer (p. 138).

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 Configure Custom Domain Name for Your Load Balancer (p. 143).

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 Enable Duration-Based Session Stickiness (p. 153).

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 Enable Application-Controlled Session Stickiness (p. 156).

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 Monitor Your Load Balancer Using Amazon CloudWatch (p. 159).

For information on deleting an existing load balancer, see Delete Your Load Balancer (p. 165).

Deploy Elastic Load Balancing in Amazon EC2-Classic

Topics

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 Deploy Elastic Load Balancing in Amazon VPC (p. 110).

Elastic Load Balancing provides Secure Sockets Layer (SSL) support for connections between clients on the front-end 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 your back-end instance, see Configure Listeners for Your Load Balancer (p. 65). And for detailed instructions on creating a load balancer in Amazon EC2 with pre-defined cipher settings and enabling the back-end authentication, see Create a Load Balancer with SSL Cipher Settings and Back-End Server Authentication (p. 73).

As the traffic to your instances increases, you might consider expanding your EC2 instances to run in an additional Availability Zone. For detailed instructions on expanding to additional Availability Zones, see Expand a Load Balanced Application to an Additional Availability Zone (p. 94).

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 Disable an Availability Zone from a Load-Balanced Application (p. 97).

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 information on locking the incoming traffic between your load balancer and your back-end instances , see Manage Security Groups in Amazon EC2-Classic (p. 99).

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

Clients can connect to your load balancer using either IPv4 or IPv6 (in regions where it is available). 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 Use IPv6 with Elastic Load Balancing (p. 108).

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

Create a Load Balancer with SSL Cipher Settings and Back-End Server Authentication

Topics

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 interface 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. 44).
- For this example, we use Availability Zone us-east-1a. 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.
- To enable HTTPS support for your listeners, you must create and then install an SSL server certificate on your load balancer. The load balancer uses the certificate to terminate and then decrypt requests before sending them to the back-end instances. Elastic Load Balancing uses AWS Identity and Access Management (IAM) to upload your certificate to your load balancer. For information on how to create an SSL certificate, go to Creating and Uploading Server Certificates in *Using AWS Identity and Access Management*.

  All your SSL server certificates are managed by IAM. By default, IAM allows 10 SSL server certificates per AWS account. If you try to upload a new server certificate after reaching this limit, you'll get an error. You can request for more certificates using this form - IAM Limit Increase Contact Us Form.

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 interface, or the Query API.

Using AWS Management Console

Topics

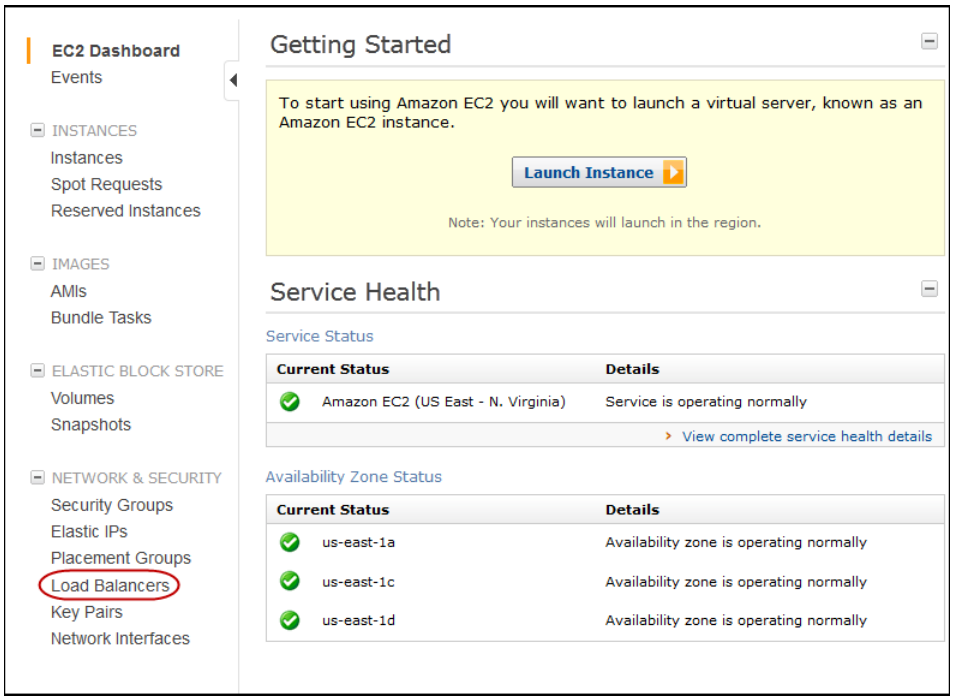
Configuring Listeners

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 80 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

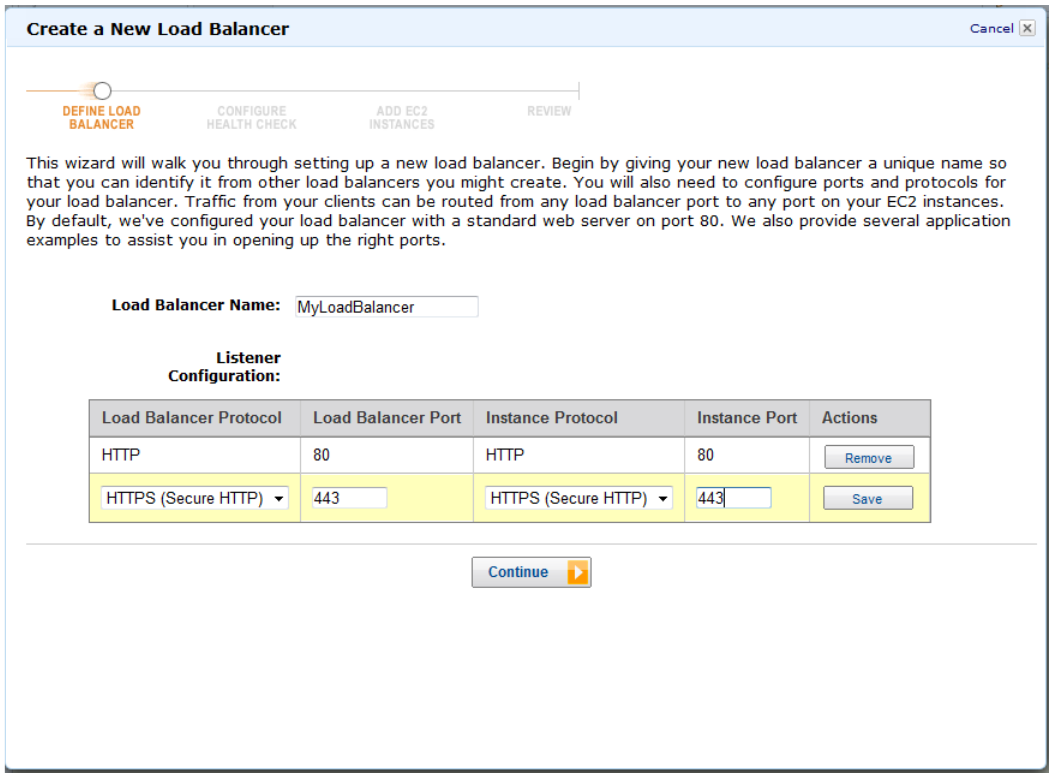
1. Start the **Create Load Balancer** wizard:

   a. On the Amazon EC2 Getting Started page, in the **EC2 Dashboard** pane, under **NETWORK & SECURITY**, click **Load Balancers**.



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

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.



5. Click **Save**, then click **Continue** to upload your SSL certificate.

   Note
   If you do not have an SSL certificate, create an SSL certificate and upload the certificate using AWS Identity and Access Management (IAM). For instructions on creating and uploading the SSL certificate, go to Creating and Uploading Server Certificates in the *Using AWS Identity and Access Management*.

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** if you have a signed SSL certificate and want to upload it.

   Before you upload, check if your certificate meets the following criteria:

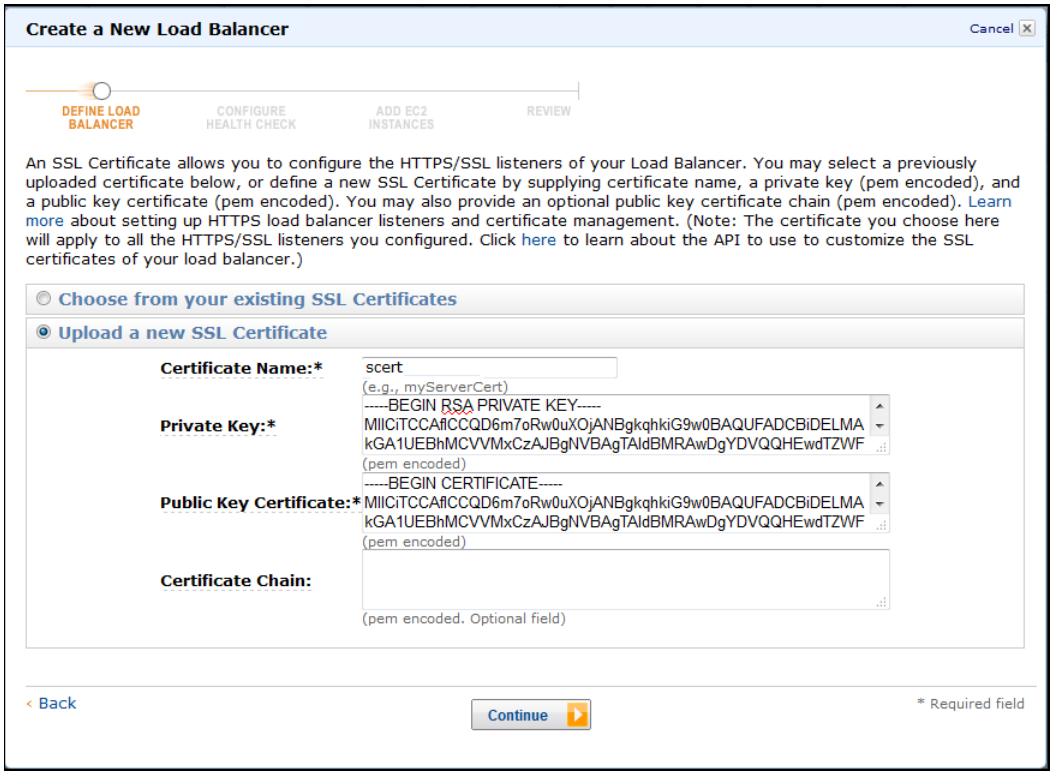
   - Certificates must follow the X.509 PEM format.
   - The current date must be between the certificate's start and end date.
   - Public and private certificate files must contain only a single certificate.
   - The private key must match the public key that is in the digital server certificate.
   - The private key must be an RSA private key in PEM format, where the PEM header is BEGIN RSA PRIVATE KEY and the footer is END RSA PRIVATE KEY.
   - The private key cannot be encrypted with a password.
   - A certificate chain starts with the immediate signing certificate and is then followed by any intermediaries in order. Intermediaries that are not involved in the trust path must not be included. The trusted root certificate can be optionally included as the last certificate.

   If your certificate does not meet the criteria listed in step 7, you might get an error when you upload it. Create a new SSL certificate and upload the certificate using IAM. For instructions on creating and

uploading the SSL certificate, go to Creating and Uploading Server Certificates in *Using AWS Identity and Access Management*.

If your certificate meets the criteria, step through the following instructions to continue uploading your 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.



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

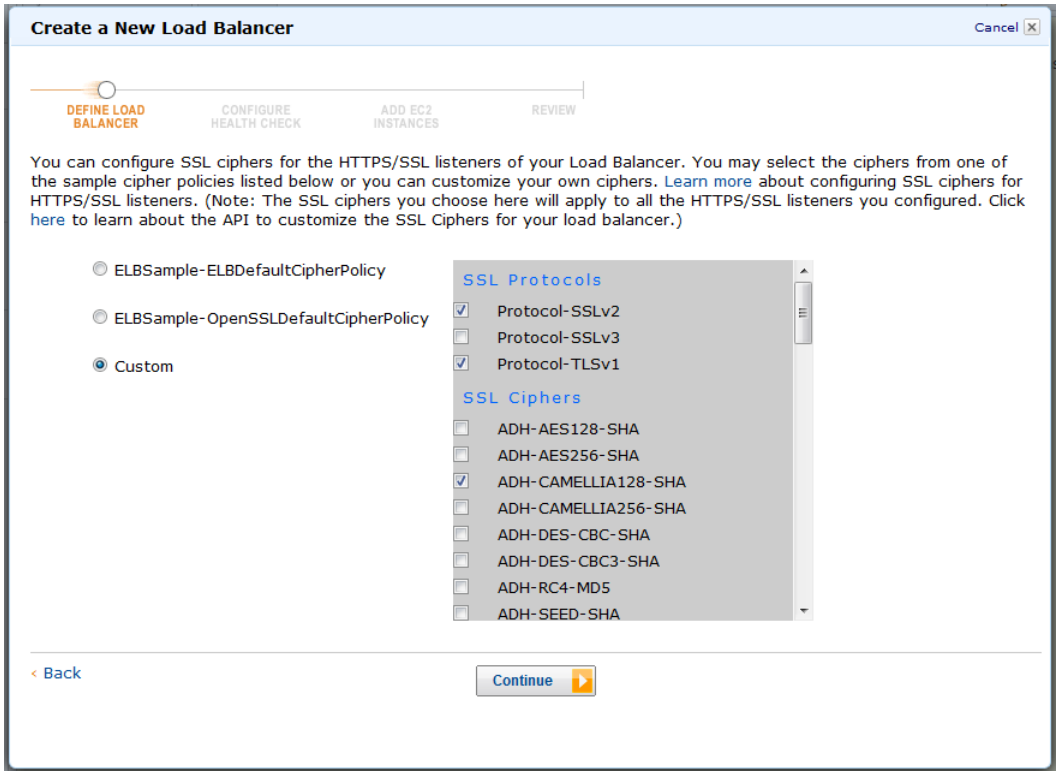
Configure SSL Ciphers

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.

To customize the SSL ciphers,

1. 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.



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

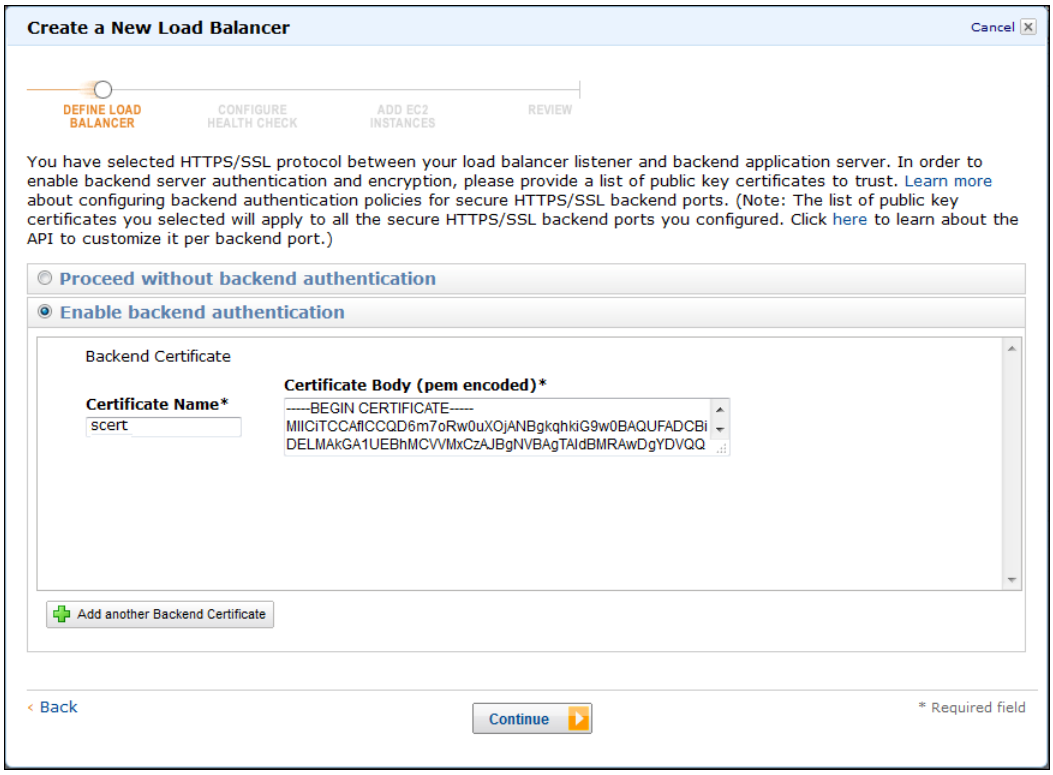
Configure Back-end Server Authentication

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. Select **Proceed without backend authentication** if you do not want to enable authentication for your back-end server.

To configure back-end server authentication

1. Select **Enable backend 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.
   b. Click **Add another Backend Certificate** to add multiple certificates.



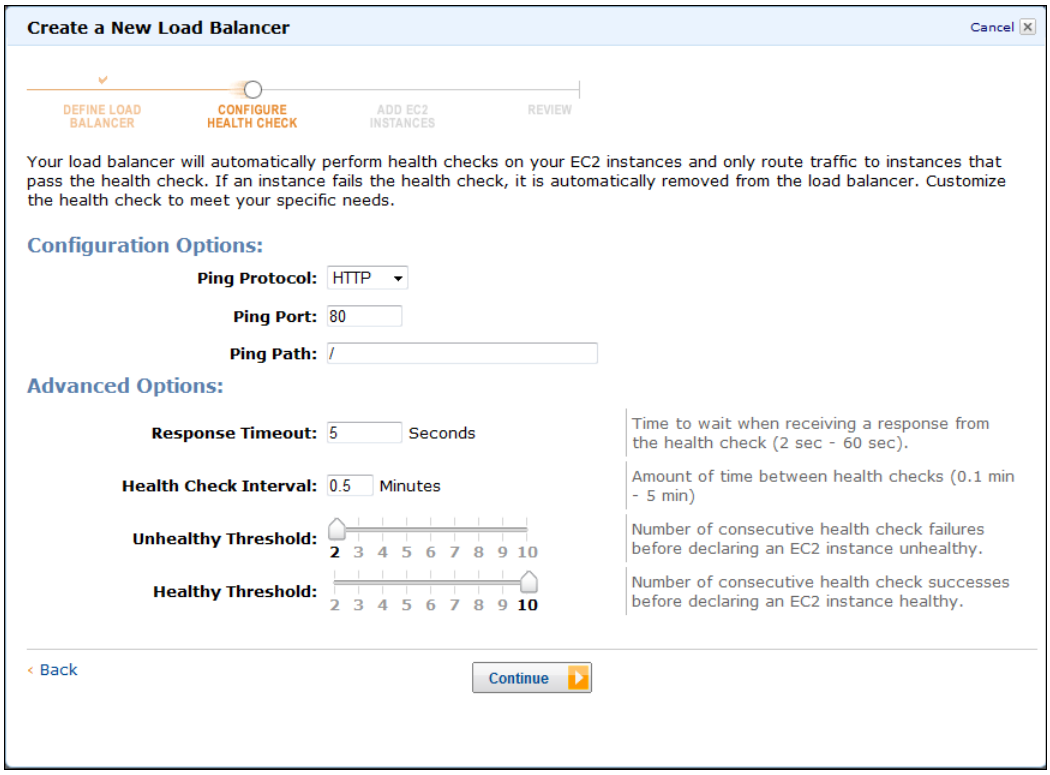
2. Click **Continue** to configure health check for your back-end server.

Configure 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.



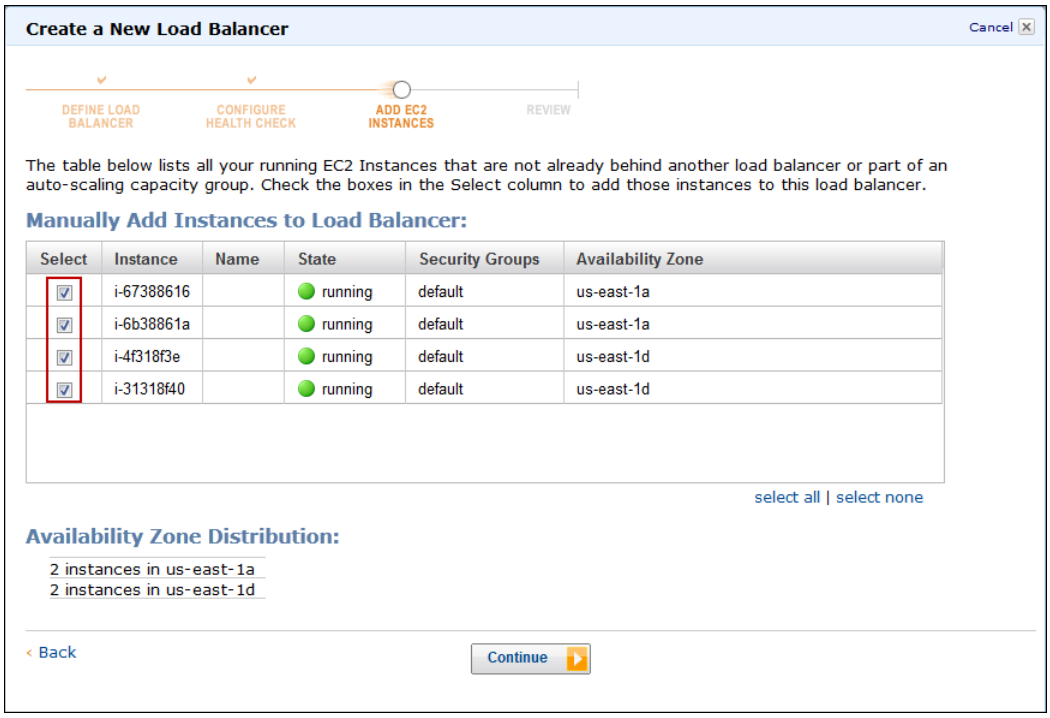
2. Click **Continue** to add your Amazon EC2 instances.

Add Amazon EC2 Instances

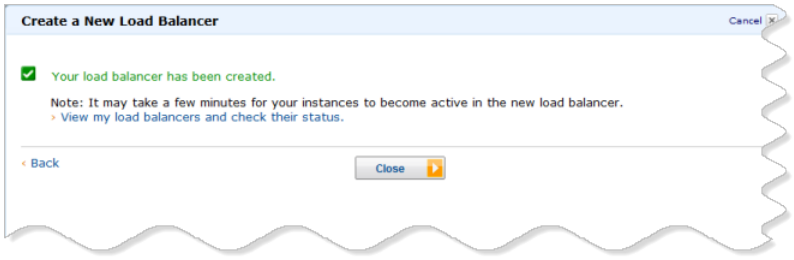
Next the wizard takes through the steps for adding Amazon EC2 instances to your load balancer.

To add Amazon EC2 instances

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



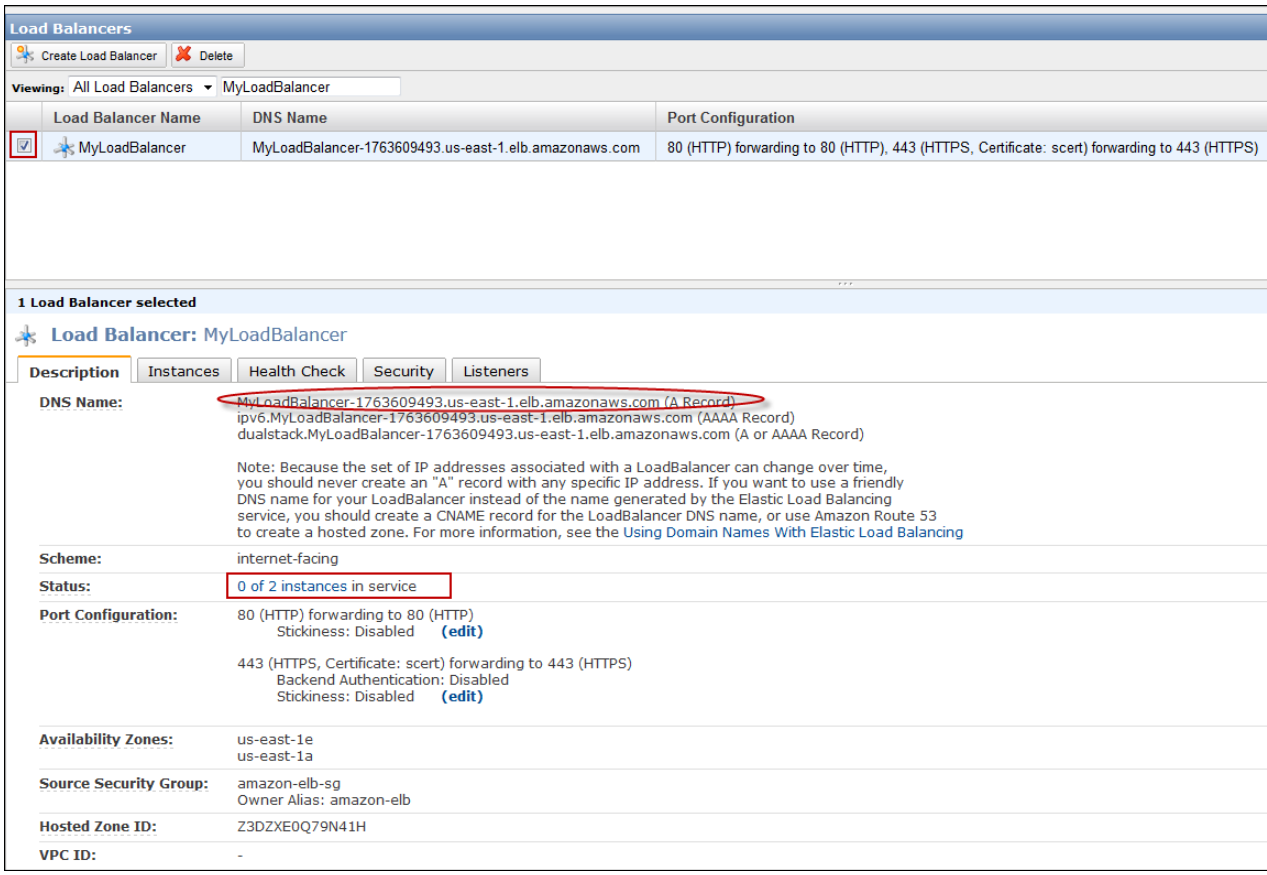
2. Click **Continue** to review your configuration. On the REVIEW page, click **Create** to create your load balancer.
3. After you click **Create** button in the **REVIEW** page, a confirmation window opens. Click **Close**.



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

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

   The bottom pane displays the description of your load balancer. Verify that the descriptions match your specifications.



If the description in the **Status** row indicates that some of your instances are not in service, it's probably because your instances are still registering. For more information, see Troubleshooting Elastic Load Balancing: Registering Instances (p. 175).

6. You can test your load balancer after you've verified that at least one of your EC2 instances is *InService*. 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.

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 change. Your load balancer cannot recognize the new IP addresses, which prevents it from routing traffic to your instance. 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 procedures associated with de-registering and then registering your instances with your load balancer, see De-Register and Register Amazon EC2 Instances (p. 136).

Using Query API

Topics

- Add Amazon EC2 Instances (p. 87)
- Check the state of your newly registered Amazon EC2 Instances (p. 88)

Configure 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 80 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. **Create and upload your certificate**

   To enable HTTPS support for your listeners, you must install an SSL server certificate on your load balancer. The load balancer uses the certificate to terminate and then decrypt requests before sending them to the back-end instances. Elastic Load Balancing uses AWS Identity and Access Management to upload your certificate to your load balancer.

   To create an SSL certificate and upload the certificate using AWS Identity and Access Management (IAM), go to Creating and Uploading Server Certificates in *Using AWS Identity and Access Management*.

   If you have an SSL certificate and have uploaded it using IAM, go to step 2.

   If you have an SSL certificate and want to upload it using IAM, step through the following instructions:

   a. Check if your certificate meets the following criteria:

      - Certificates must follow the X.509 PEM format.
      - The current date must be between the certificate's start and end date.
      - Public and private certificate files must contain only a single certificate.
      - The private key must match the public key that is in the digital server certificate.
      - The private key must be an RSA private key in PEM format, where the PEM header is BEGIN RSA PRIVATE KEY and the footer is END RSA PRIVATE KEY.
      - The private key cannot be encrypted with a password.
      - A certificate chain starts with the immediate signing certificate and is then followed by any intermediaries in order. Intermediaries that are not involved in the trust path must not be included. The trusted root certificate can be optionally included as the last certificate.

      If your SSL certificate does not meet the criteria listed in this step, you might get an error when you upload it. Create a new SSL certificate and upload the certificate using AWS Identity and Access Management (IAM). For instructions on creating and uploading the SSL certificate, go to Creating and Uploading Server Certificates in the *Using AWS Identity and Access Management*.

      If your certificate meets the criteria, step through the following instructions to continue uploading your certificate.

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

      - *`ServerCertificateName`* = `testCert`

- *CertificateBody* = *<encoded certificate body>*
- *PrivateKey* = *<encoded private key>*
- [Optional] *CertificateChain* = *<concatenation of the encoded public key certificates>*
- [Optional] *Path* = /

  Note
  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*.

c. The response includes the Amazon Resource Name (ARN) of the server certificate. Copy the Amazon Resource Name (ARN) of the certificate for the next step.

2. Call CreateLoadBalancer with the following parameters:

   - *AvailabilityZones* = us-east-1a
   - *Listener*
     - *Protocol* = HTTP
     - *InstanceProtocol* = HTTP
     - *InstancePort* = 80
     - *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. Copy the DNS name in a safe place. You'll be using the DNS name to connect to the load balancer later in the procedure. You can also later map any other domain name (such as www.example.com) to your load balancer's DNS name using CNAME or some other technique.

Configure 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`

Configure 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

Configure 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:80/ping`

    Note
    Make sure your instances respond to a ping on port 80 with an HTTP 200 status code.
  - *`Interval`* = `30`
  - *`Timeout`* = `3`
  - *`HealthyThreshold`* = `2`
  - *`UnhealthyThreshold`* = `2`

Add 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 Deploy Elastic Load Balancing in Amazon VPC (p. 110)

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 Manage Security Groups in Amazon EC2 (p. 99).

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 change. Your load balancer cannot recognize the new IP address, which prevents it from routing traffic to your instance. We recommend that 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-Register and Register Amazon EC2 Instances (p. 136).

Check the state of your newly registered Amazon EC2 Instances

In this example, you check the state of your newly registered Amazon EC2 instances. Your load balancer is usable as soon as any one of your registered EC2 instance is in *InService* state.

To check the state of your newly registered Amazon EC2 instances

1. Call `DescribeInstanceHealth` with the following parameters:
   - *`LoadBalancerName`* = `MyLoadBalancer`
   - *`Instances.member.N`* = [`i-4f8cf126` ,`i-0bb7ca62`]
2. If the **State** field of some or all of your instances displays *OutOfService*, it's probably because your instances are still registering. For more information, see Troubleshooting Elastic Load Balancing: Registering Instances (p. 175).
3. You can test your load balancer after you have verified that at least one of your EC2 instances is *InService*. To test your load balancer, copy the **DNS Name** of your load balancer that you received after you completed the Configure Listeners (p. 83) task, 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.

Using the Command Line Interface

Topics

Configure 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 80 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. **Create and upload your certificate**

   To enable HTTPS support for your listeners, you must install an SSL server certificate on your load balancer. The load balancer uses the certificate to terminate and then decrypt requests before sending them to the back-end instances. Elastic Load Balancing uses AWS Identity and Access Management (IAM) to upload your certificate to your load balancer.

   To create an SSL certificate and upload the certificate using IAM, go to Creating and Uploading Server Certificates in the *Using AWS Identity and Access Management*.

If you already have an SSL certificate and have uploaded it using IAM, go to step 2.

If you already have an SSL certificate and want to upload it using IAM, step through the following instructions.

a. Check if your signed certificate meets the following criteria:

- Certificates must follow the X.509 PEM format.
- The current date must be between the certificate's start and end date.
- Public and private certificate files must contain only a single certificate.
- The private key must match the public key that is in the digital server certificate.
- The private key must be an RSA private key in PEM format, where the PEM header is BEGIN RSA PRIVATE KEY and the footer is END RSA PRIVATE KEY.
- The private key cannot be encrypted with a password.
- A certificate chain starts with the immediate signing certificate and is then followed by any intermediaries in order. Intermediaries that are not involved in the trust path must not be included. The trusted root certificate can be optionally included as the last certificate.

If your certificate does not meet the criteria listed in this step, you might get an error when you upload it. Create a new SSL certificate and upload the certificate using AWS Identity and Access Management (IAM). For instructions on creating and uploading the SSL certificate, go to Creating and Uploading Server Certificates in the *Using AWS Identity and Access Management*.

If your certificate meets the criteria, step through the following instructions to continue uploading your certificate.

b. Use the IAM command `iam-servercertupload` in verbose mode, as in the following example, to upload your SSL certificate to the AWS IAM service.

Note
You have to download and install the AWS Identity and Access Management command line interface (IAM CLI) to use this command. For more information about installing the IAM CLI, go to Getting the Command Line Tools in the *AWS Identity and Access Management CLI Reference*.

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

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

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

2. The first line is the ARN and the second line is the GUID. Copy the ARN of the certificate 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=80,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 DNS name of your load balancer.

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

4. Copy the DNS name in a safe place. You'll be using the DNS name to connect to the load balancer later in the procedure. You can also later map any other domain name (such as www.example.com) to your load balancer's DNS name using CNAME or some other technique.

Configure SSL Ciphers

Your Elastic Load Balancer is first 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}
```

Configure 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
```

Configure 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:80/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:80/ping  30     3      2       2
```

Add 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 Deploy Elastic Load Balancing in Amazon VPC (p. 110).

- **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-Register and Register Amazon EC2 Instances (p. 136).

Check the state of your newly registered Amazon EC2 Instances

In this example, you check the state of your newly registered Amazon EC2 instances. Your load balancer is usable as soon as any one of your registered EC2 instance is in *InService* state.

1. **To check the state of your newly registered Amazon EC2 instances**

   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_ID  INSTANCE_ID  STATE         DESCRIPTION
                         REASON-CODE
INSTANCE_ID  i-4f8cf126   OutOfService  Instance has failed at least the
UnhealthyThreshold number of health checks consecutively.  Instance
INSTANCE_ID  i-0bb7ca62   InService     N/A
                     N/A
```

2. If the **State** field of some or all of your instances display *OutOfService*, it's probably because your instances are still registering. For more information, see Troubleshooting Elastic Load Balancing: Registering Instances (p. 175).
3. You can test your load balancer after you have verified that at least one of your EC2 instances is *InService*. To test your load balancer, copy the **DNS Name** of your load balancer that you got after you completed the Configure Listeners (p. 88) task, 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.

Expand 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.

Preconditions:

- You have set up an HTTP load balancer in Availability Zone us-east-1a as in Creating a HTTPS/SSL Load Balancer (p. 73).
- 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
```

Disable 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 there are no healthy instances in an enabled Availability Zone, the request will be forwarded to healthy instances in another enabled Availability Zone. If you do not plan to have instances in a zone, you must disable it by calling the `DisableAvailabilityZonesForLoadBalancer` action on 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
```

Manage Security Groups in Amazon EC2-Classic

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. For each seucurity group, you can add rules that govern the allowed inbound traffic to instances in the 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. For information on Amazon EC2 security groups, go to Using Security Groups.

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. To lock down traffic between your load balancer and your back-end instances, add or modify a rule to your back-end security group that limits ingress traffic so that it can come only from the Amazon EC2 source security group provided by the Elastic load Balancing.

Locking down traffic between Elastic Load Balancing and back-end Amazon EC2 instance

Topics

In this section, we will walk you through the process for locking down the traffic between your load balancer and your back-end Amazon EC2 instances.

Before you continue, be sure you've read Security Group Concepts section in the *Amazon EC2 User Guide*.

The following table outlines the steps for locking down the traffic between your load balancer and the back-end instances. This will be followed by instructions for locking down traffic between your load balancer and your back-end instances using the AWS Management Console, the command line interface (CLI), or the Query API.

If you are planning on using the command line interface (CLI), be sure that you've installed the Amazon EC2 command line interface tool. For information on downloading and installing the tool, go to Setting Up the Amazon EC2 Command Line Tools section in the *Amazon EC2 User Guide*.

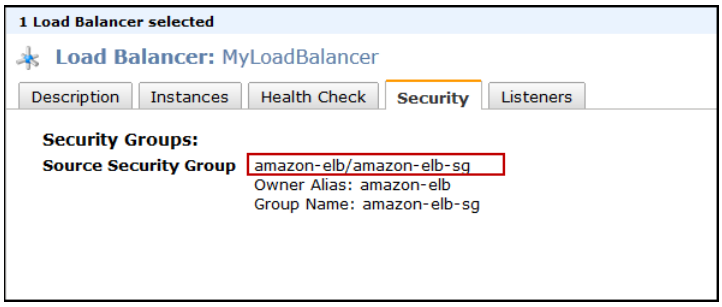
Steps for locking down the traffic between your load balancer and your back-end instance

1	Get the name of the source security group provided by Elastic Load Balancing for your load balancer.
2	Add a new rule in the security group associated with your Amazon EC2 back-end instances to allow inbound traffic from your load balancer.
3	Verify that the new rule has been added.
4	[optional] Remove the rules in your back-end instance's security group that are less restrictive.

Using the AWS Management Console

1. **To get the name of the source security group provided by Elastic Load Balancing for your load balancer,**

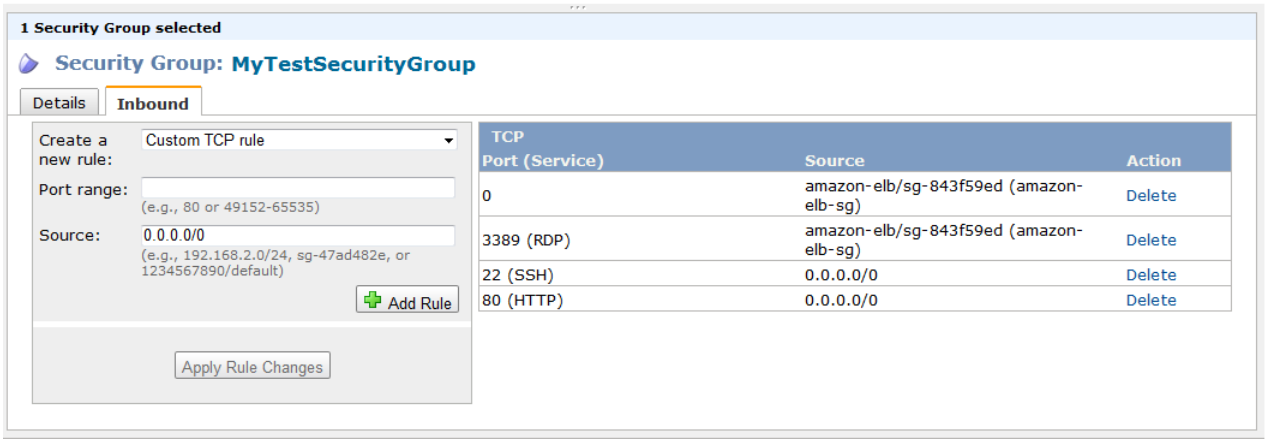
   a. On the Amazon EC2 Getting Started page, in the **EC2 Dashboard** pane, under **NETWORK & SECURITY**, click **Load Balancers**.
   b. On the **Load Balancers** page, select your load balancer.
   c. The bottom pane displays the details of your load balancer.
   d. Click the **Security** tab.
   e. Copy the name displayed in the **Source Security Group** row. This is the source security group provided by Elastic Load Balancing for your load balancer.



2. **To add a rule in the security group of your Amazon EC2 back-end instances to allow incoming traffic from the Elastic Load Balancing source security group,**

   a. On the **Load Balancers** page, click **Instances** in the **Navigation** pane.
   b. On the **My Instances** page, select an instance registered with your load balancer.
   c. The bottom pane displays details of your instance. Make sure that the **Description** tab is selected.
   d. Make a note of the name of the security group displayed in the **Security Groups:** field. This is the security group associated with your back-end instance.
   e. Click **Security Groups** in the Navigation pane.
   f. On the **Security Groups** page, select the security group associated with your back-end instance.
   g. The bottom pane displays details of the security group.
   h. Click the **Inbound** tab. The right pane of the **Inbound** tab displays the rules that belong to the security group.
   i. In the **Port range:** field, specify the port or port range for accessing your Amazon EC2 instances.
   j. In the **Source:** field, type the source security group name of your load balancer. This should be the same name you used in step 1. In our example it's **amazon-elb/amazon-elb-sg**.
   k. Click **Add Rule** and then click **Apply Rule Changes**.

3. The right-side pane of the **Inbound** tab displays the new rule. Verify that the new rule is added to the security group of your back-end instance.



4. [optional] If your security group has rules that are less restrictive than the rule you added in step 2, remove the less restrictive rule by clicking **Delete** in the row that has the less restrictive rule.

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

Using the command line interface (CLI)

1. **To get the name of the source security group provided by Elastic Load Balancing for your load balancer,**
   Enter the `elb-describe-lbs` command. You must include the *`--show-long`* parameter to display the source security group 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 `MyLoadBalancer` load balancer.

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

   The following is an example response.

```
LOAD_BALANCER,NAME,DNS_NAME,CANONICAL_HOSTED_ZONE_NAME,CANONICAL_HOS
TED_ZONE_NAME_ID,
HEALTH_CHECK,AVAILABILITY_ZONES,SUBNETS,VPC_ID,INSTANCE_ID,LISTENER_DESCRIP
TIONS,
BACKEND_SERVER_DESCRIPTIONS,OTHER_POLICIES,SOURCE_SECURITY_GROUP,SECUR
ITY_GROUPS,CREATED_TIME,SCHEME
LOAD_BALANCER,MyLoadBalancer,MyLoadBalancer-91948427.us-east-1.elb.amazon
aws.com
,MyLoadBalancer-91948427.us-east-1.elb.amazonaws.com,Z3DZXE0Q79N41H,"{inter
val=6
,target=HTTP:80/,timeout=5,healthy-threshold=2,unhealthy-threshold=2}",us-
east-1
e,(nil),(nil),"i-f5ab7988, i-fbab7986","{protocol=HTTP,lb-port=80,instance-
proto
col=HTTP,instance-port=80,policies=},{protocol=HTTPS,lb-port=443,instance-
protoc
ol=HTTP,instance-port=80,cert-id=arn:aws:iam::803981987763:server-certific
ate/sc
```

```
ert,policies=AWSConsole-SSLNegotiationPolicy-MyLoadBalancer}",(nil),AWSCon
sole-S
SLNegotiationPolicy-MyLoadBalancer,"{owner-alias=example-elb,group-name=ex
ample-elb-sg}"
,(nil),2012-09-28T17:57:27.580Z,internet-facing
```

The response element includes a source security group data structure composed of an `owner-alias` and `group-name`. This is the source security group associated with your load balancer. Make a note of the `owner-alias` and `group-name`. You'll use it in the next step.

2. **To add a rule in the security group of your Amazon EC2 back-end instances to allow incoming traffic from the Elastic Load Balancing source security group,**

   You will be using the Amazon EC2 CLI commands for this step.

   a. [optional] If you do not know the name of the security group of your back-end instances, enter the `ec2-describe-group` command as in the following example.

      ```
      ec2-describe-group --headers
      ```

      The response element includes the details of all the security groups in your account. Make a note of the security group name associated with the back-end instance that you want to modify.

   b. Enter the `ec2-authorize` command to add a rule to the security group associated with your back-end instance.

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

      ```
      ec2-authorize MyTestSecurityGroup -u amazon-elb -o amazon-elb-sg
      ```

      For specifying additional parameters for ports and protocols, go to ec2-authorize in the *Amazon EC2 CLI Reference*.

3. **To verify that the security group associated with your back-end instance has the new rule that allows incoming traffic from the Elastic Load Balancing source security group,**

   Enter `ec2-describe-group` command as in the following example:

   ```
   ec2-describe-group MyTestSecurityGroup  --headers
   ```

   The following is the example response that lists the newly added rule:

   ```
   GROUP   Id      Owner   Name    Description
   GROUP   sg-9cd8a3f4     000011112222    MyTestSecurityGroup  This is an ex
   ample
   PERMISSION      000011112222    MyTestSecurityGroup     ALLOWS  tcp     0
         0
   FROM    USER    amazon-elb      NAME amazon-elb-sg      ID sg-843f59ed  in
   gress
   PERMISSION      000011112222    MyTestSecurityGroup     ALLOWS  tcp     3389
       3389
   FROM    USER    amazon-elb      NAME amazon-elb-sg      ID sg-843f59ed  in
   gress
   PERMISSION      000011112222    MyTestSecurityGroup     ALLOWS  tcp     22
   ```

```
      22
FROM    CIDR    0.0.0.0/0       ingress
PERMISSION      000011112222    MyTestSecurityGroup     ALLOWS  tcp     80
      80
FROM    CIDR    0.0.0.0/0       ingress
```

4. [optional] 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 `MyTestSecurityGroup`.

```
ec2-revoke MyTestSecurityGroup -p 80 -s 0.0.0.0/0
```

   Note
   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 the Query API

1. **To get the name of the source security group provided by Elastic Load Balancing for your load balancer,**

   a. Call the `DescribeLoadBalancers` action with the following parameter:

   - `LoadBalanacerNames` = `MyLoadBalancer`

   The response should include the details of your load balancer. The information you get should be similar to the following example:

```
<DescribeLoadBalancersResponse xmlns="http://elasticloadbalancing.amazon
aws.com/doc/2012-06-01/">
  <DescribeLoadBalancersResult>
    <LoadBalancerDescriptions>
      <member>
        <SecurityGroups/>
        <LoadBalancerName>MyLoadBalancer</LoadBalancerName>
        <CreatedTime>2012-09-28T17:57:27.580Z</CreatedTime>
        <HealthCheck>
          <Interval>6</Interval>
          <Target>HTTP:80/</Target>
          <HealthyThreshold>2</HealthyThreshold>
          <Timeout>5</Timeout>
          <UnhealthyThreshold>2</UnhealthyThreshold>
        </HealthCheck>
        <ListenerDescriptions>
          <member>
            <PolicyNames/>
            <Listener>
              <Protocol>HTTP</Protocol>
              <LoadBalancerPort>80</LoadBalancerPort>
              <InstanceProtocol>HTTP</InstanceProtocol>
```

```
              <InstancePort>80</InstancePort>
            </Listener>
          </member>
          <member>
            <PolicyNames>
            <member>AWSConsole-SSLNegotiationPolicy-MyLoadBalancer</mem
ber>
            </PolicyNames>
            <Listener>
              <Protocol>HTTPS</Protocol>
              <LoadBalancerPort>443</LoadBalancerPort>
              <InstanceProtocol>HTTP</InstanceProtocol>
              <SSLCertificateId>arn:aws:iam::000011112222:server-certi
ficate/scert</SSLCertificateId>
              <InstancePort>80</InstancePort>
            </Listener>
          </member>
        </ListenerDescriptions>
        <Instances>
          <member>
            <InstanceId>i-f5ab7988</InstanceId>
          </member>
          <member>
            <InstanceId>i-fbab7986</InstanceId>
          </member>
        </Instances>
        <Policies>
          <AppCookieStickinessPolicies/>
          <OtherPolicies>
           <member>AWSConsole-SSLNegotiationPolicy-MyLoadBalancer</mem
ber>
          </OtherPolicies>
          <LBCookieStickinessPolicies/>
        </Policies>
        <AvailabilityZones>
          <member>us-east-1e</member>
        </AvailabilityZones>
        <CanonicalHostedZoneName>MyLoadBalancer-91948427.us-east-
1.elb.amazonaws
.com</CanonicalHostedZoneName>
        <CanonicalHostedZoneNameID>Z3DZXE0Q79N41H</CanonicalHostedZone
NameID>
        <Scheme>internet-facing</Scheme>
        <SourceSecurityGroup>
          <OwnerAlias>amazon-elb</OwnerAlias>
          <GroupName>amazon-elb-sg</GroupName>
        </SourceSecurityGroup>
       <DNSName>MyLoadBalancer-91948427.us-east-1.elb.amazonaws.com</DNS
Name>
        <BackendServerDescriptions/>
        <Subnets/>
      </member>
    </LoadBalancerDescriptions>
  </DescribeLoadBalancersResult>
  <ResponseMetadata>
    <RequestId>0d7f1256-0b10-11e2-b66c-d3e9bEXAMPLE</RequestId>
  </ResponseMetadata>
```

```
</DescribeLoadBalancersResponse>
```

b. The response element includes a `SourceSecurityGroup` data structure composed of an `OwnerAlias` and `GroupName`. This is the security group provided by Elastic Load Balancing for your load balancer. Make a note of the details in the `SourceSecurityGroup` data structure. You'll use it in the next step.

2. **To add a rule in the security group of your Amazon EC2 back-end instances to allow incoming traffic from the Elastic Load Balancing source security group,**

   a. [optional] If you do not know the name of the security group of your back-end instances, call the `DescribeSecurityGroup` action to find out the name of the security group associated with your back-end instance.

      The response includes the details of all the security groups in your account. Make a note of the security group name associated with the back-end instance that you want to modify.

   b. Call the `AuthorizeSecurityGroupIngress` action with the following parameters :

      - `GroupName` = `MyTestSecurityGroup`
      - `IpPermissions.1.Groups.1.UserId` = `amazon-elb`
      - `IpPermissions.1.Groups.1.GroupName` = `amazon-elb-sg`

      For information on specifying additional parameters for the ports and protocols, go to AuthorizeSecurityGroupIngress in the *Amazon EC2 API Reference*.

3. Verify that the security group associated with your back-end instance has the new rule that allows incoming traffic from the Elastic Load Balancing source security group.

   Call `DescribeSecurityGroups` with the following parameters:

   - `GroupName` = `MyTestSecurityGroup`

   The response should include the details of the security group including the newly added rule. The information you get should be similar to the following example:

```
<DescribeSecurityGroupsResponse xmlns="http://ec2.amazonaws.com/doc/2012-
03-01/">
  <requestId>a8a2955f-61db-4e5e-b138-e2be7EXAMPLE</requestId>
      <securityGroupInfo>
        <item>
          <ownerId>000011112222</ownerId>
          <groupId>sg-9cd8a3f4</groupId>
          <groupName>MyTestSecurityGroup </groupName>
          <groupDescription>This is an example </groupDescription>
          <ipPermissions>
            <item>
              <ipProtocol>tcp</ipProtocol>
              <fromPort>0</fromPort>
              <toPort>0</toPort>
              <groups>
                <item>
                  <userId>amazon-elb</userId>
```

```
                <groupId>sg-843f59ed</groupId>
                <groupName>amazon-elb-sg</groupName>
              </item>
            </groups>
            <ipRanges/>
          </item>
          <item>
            <ipProtocol>tcp</ipProtocol>
            <fromPort>3389</fromPort>
            <toPort>3389</toPort>
            <groups>
              <item>
                <userId>amazon-elb</userId>
                <groupId>sg-843f59ed</groupId>
                <groupName>amazon-elb-sg</groupName>
              </item>
            </groups>
            <ipRanges/>
          </item>
          <item>
            <ipProtocol>tcp</ipProtocol>
            <fromPort>22</fromPort>
            <toPort>22</toPort>
            <groups/>
            <ipRanges>
              <item>
                <cidrIp>0.0.0.0/0</cidrIp>
              </item>
            </ipRanges>
          </item>
          <item>
            <ipProtocol>tcp</ipProtocol>
            <fromPort>80</fromPort>
            <toPort>80</toPort>
            <groups/>
            <ipRanges>
              <item>
                <cidrIp>0.0.0.0/0</cidrIp>
              </item>
            </ipRanges>
          </item>
    </securityGroupInfo>
  </DescribeSecurityGroupsResponse>
```

4. [optional] If your security group has rules that are less restrictive than the rule you added in the previous step, use the `RevokeSecurityGroupIngress` action 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 calls the `RevokeSecurityGroupIngress` action with the following parameters to remove a rule that allows HTTP traffic from all IPv4 addresses from a security group named `MyTestSecurityGroup`.

   - `GroupName` = `MyTestSecurityGroup`
   - `IpPermissions.1.FromPort` = `80`
   - `IpPermissions.1.IpRanges.1.CidrIp` = `0.0.0.0/0`

Note

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).

Use 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. 5).

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 Option 2: Create an Amazon Route 53 Hosted Zone (p. 145).

Deploy Elastic Load Balancing in Amazon VPC

Topics

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

Within this virtual private cloud, you can launch Amazon EC2 instances that have private (RFC 1918) IP addresses in the classless inter-domain routing (CIDR) range of your choice (for example, 10.0.0.0/16). You can use a load balancer to monitor and route traffic to your EC2 instances launched within a VPC.

You can create a VPC that spans multiple Availability Zones then add one or more subnets in each Availability Zone. A subnet in Amazon VPC is a subdivision within an Availability Zone defined by a segment of the IP address range of the VPC. Using subnets you can group your instances based on your security and operational needs. A subnet resides entirely within the Availability Zone it was created in.

To enable communication between the Internet and the instances in your subnets, you must create and attach an Internet gateway (IGW) to your VPC. An IGW connects the instances within your subnets to the Internet. The subnets that interact directly with the Internet must contain public instances (instances with public IP addresses).

For more information on Amazon VPC, see What is Amazon VPC in the *Amazon Virtual Private Cloud User Guide*. For information on Amazon VPC subnets, see Your VPC and Subnets.

To load balance your EC2 instances in a VPC, when you register load balancers in multiple Availability Zones, specify one subnet in each Availability Zone to which you want to attach the load balancer. Because a subnet is created for an Availability Zone, specifying the subnet ensures that the load balancer is configured to listen to requests in the corresponding Availability Zone.

When you attach your load balancer to a subnet, this defines the subnet that traffic must enter 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. To ensure that your load balancer can scale properly, specify that the CIDR block of the subnet to which you attach the load balancer has at least a /27 bitmask (e.g., 10.0.0.0/27). You should also have at least 20 free IP addresses in the subnet where you attach the load balancer.

Elastic Load Balancing on Amazon VPC works essentially the same way as it does on 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. In Amazon EC2, 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. You cannot modify the source security group. Within Amazon VPC, you have control over the security groups assigned to your load balancer. Having control over your load balancer's security groups allows you to choose the ports and protocols to accept. For example, in Amazon 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.
Dedicated tenancy VPCs are not currently supported by Elastic Load Balancing.

Default VPC

If your AWS account comes with a default virtual private cloud (default VPC), your instances are launched within the default VPC, by default, unless you specify a subnet from a nondefault VPC. For more information, see Detecting Your Supported Platforms and Whether You Have a Default VPC.

A default VPC combines the benefits of the advanced networking features provided by Amazon VPC platform (EC2-VPC) with the ease of use of the Amazon Elastic Compute Cloud platform (EC2-Classic).

Your default VPC automatically comes with a default subnet in each Availability Zone, an Internet Gateway connected to your default VPC, and a default security group associated with your default VPC, among other default configurations. For more information on default VPC and Subnets, see Your Default VPC and Subnets.

When you launch your EC2 instances within default VPC without specifying a subnet, it is automatically launched into a default subnet in your default VPC. By default, we select an Availability Zone for you and launch the instance into the corresponding subnet for that Availability Zone. Alternatively, you can select the Availability Zone for your instance by selecting its corresponding default subnet.

To load balance your EC2 instances launched in default VPC, you have to create your load balancers within your default VPC. When you create a load balancer within default VPC, Elastic Load Balancing automatically creates a security group by defining the ports specified for the load balancer to be opened.

If you are using the AWS Management Console to create your load balancer with default VPC, you can choose the default VPC security group associated with your default VPC, choose any other pre-existing security group associated with your AWS account, or modify the existing security group to create a new one. If you choose a non-default pre-existing security group, ensure that it allows ingress to the ports that you configured the load balancer to use. If you choose to create a security group, the console will define these ports to be open for you.

The security group you create using the AWS Management Console is associated with your load balancer. You cannot delete the security group as long as it is associated with a load balancer.

If you are using the Elastic Load Balancing command line interface (CLI) or the Query API to create your load balancer within your default VPC, a default security group; default_elb_*special number* will be created for 0.0.0.0/0 with the ports specified for the load balancer. Currently, Elastic Load Balancing does not support the option to choose a security group to associate with your load balancer when you use either the CLI or the Query API.

A security group created for your load balancer within default VPC using either the CLI or the Query API is associated with your AWS account. Only one security group will be created per AWS account. Subsequent calls to create new load balancers will re-use the same security group. If you add listeners to an existing load balancer, you will need to review the security groups that are applied to the load balancer; otherwise, traffic may not reach the load balancer. If you create a load balancer that uses this default security group and the new load balancer has a different listener configuration, you will need to review and update the security groups that are applied to the load balancer.

If you are using the CLI to create a load balancer for your EC2 instances in default VPC, you can see the security group associated with your load balancer by using the `-show-long` option with `elb-describe-lbs` command. The load balancer security group is listed as `SOURCE_SECURITY_GROUP` in the CLI output and the query response. This security group can be used to limit ingress to your instance from only the selected security group.

Whether you are using the console, CLI, or the Query API, if you update the listeners of your load balancer within default VPC, the security groups associated with the load balancer will not change. If you delete your load balancer, the security group will not be deleted automatically.

For information on creating a basic load balancer for your EC2 instances within default VPC, see Create a Basic Load Balancer in Default VPC (p. 34).

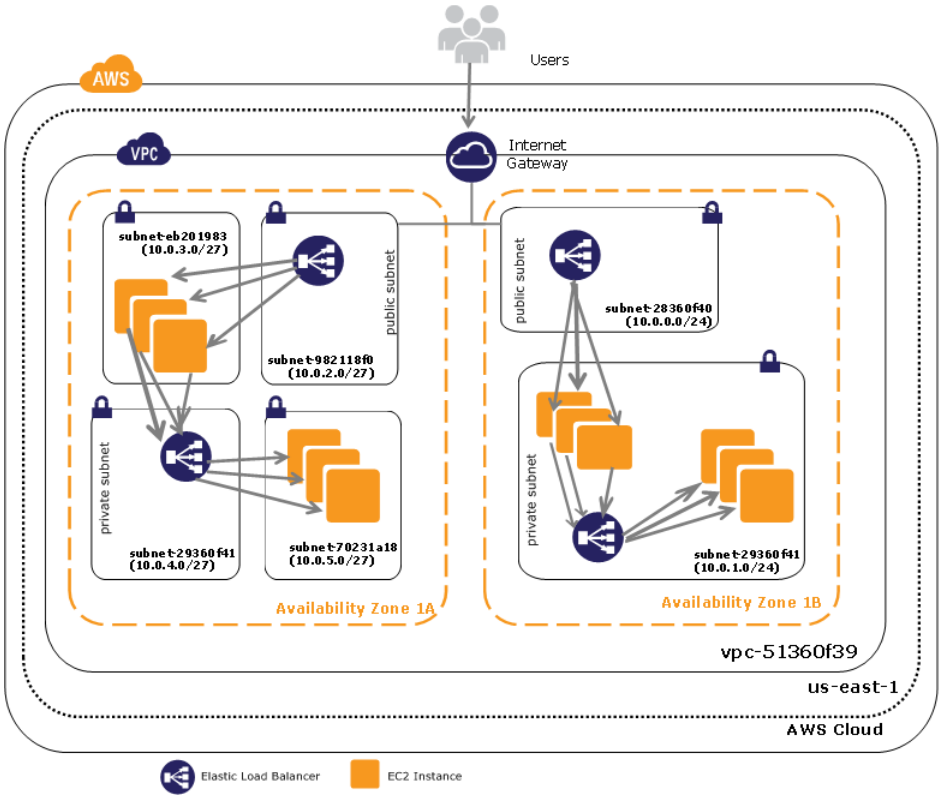
Internet-facing and Internal Load Balancers

You can make your load balancer internal (private) or Internet-facing (public) when creating it within a VPC. When you make your load balancer internal, a DNS name will be created, and it will contain the private IP address of the load balancer. Internal load balancer is not exposed to the internet. When you make your load balancer Internet-facing, a DNS name will be created with the public IP address. The DNS records are publicly resolvable in both cases.

By combining both internal and internet-facing load balancers, you can balance requests between multiple tiers of your application. For example, let us say you have web servers at your front-end that takes requests from the internet and passes it on to your back-end application instances. You can create an internal load balancer in your VPC and then place your back-end application instances behind the internal load balancer. You can create an internet-facing load balancer with the DNS name and public IP address and place it in front of your web server. Your web server will take requests coming from the internet-facing load balancer and will make requests to the internal load balancer, using private IP addresses that are resolved from the internal load balancer's DNS name. The internal load balancer will route requests to the back-end application instances, which are also using private IP addresses and only accept requests from the internal load balancer. With this multi-tier architecture, all your infrastructure can use private IP addresses and security groups so that the only part of your architecture that has public IP address is the internet-facing load balancer.

For a load balancer to be Internet-facing, the load balancer must reside in a public subnet (with public IP addresses). The application instances behind the load balancer do not need to be in the public subnets.

The following diagram shows Elastic Load Balancing within Amazon VPC combining both Internet-facing and internal load balancers.



Create Amazon VPC for Elastic Load Balancing

To create and use Elastic Load Balancing load balancers within Amazon VPC, you have to first configure your VPC environment. This section provides you with steps you can use to configure your VPC. If you are using default VPC, your VPC environment is automatically configured for you. You can skip this step.

- Create your Amazon VPC with an Internet gateway and create subnets in each Availability Zone in which you want to load balance your instances. For information on creating an Amazon VPC and subnets, see Get Started with Amazon VPC.
- If you are creating an Internet-facing load balancer, be sure to create a VPC with public subnet. Your Internet-facing load balancer must be in a public subnet to establish connections with the Internet.
- 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 within Amazon VPC, see Launch an Instance.
- The examples in this guide assume that your Amazon VPCs are in the US East (Northern Virginia) Region. You can set up your VPCs in other AWS regions. For example, if you want to launch your instances in a VPC in a region 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. 45). For information about this product's regions and endpoints, go to Regions and Endpoints in the Amazon Web Services General Reference.

You've now completed creating your VPC environment for your load balancers. Next, click any one of the following sections to create and manage your load balancer within your Amazon VPC.

- Create a Basic Load Balancer in EC2-VPC (p. 23)
- Create a Basic Load Balancer in Default VPC (p. 34)
- Create a Basic Internal Load Balancer in Amazon VPC (p. 113)
- Attach Your Load Balancer to a Subnet (p. 122)
- Detach Your Load Balancer from a Subnet (p. 124)
- Manage Security Groups in Amazon VPC (p. 126)

Create a Basic Internal Load Balancer in Amazon VPC

Topics

This topic uses an example to walk you through the process for creating a basic internal load balancer within your VPC and registering your EC2 instances with the newly created internal load balancer. This example uses default configurations for security group, listener protocols and ports, and for the health check. If you want to create an internet-facing load balancer, see Create a Basic Load Balancer in EC2-VPC (p. 23).

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

Creating a Basic Internal Load Balancer in 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 in which to launch your load balancer.
4	Select security groups to assign to your load balancer.
5	Add Amazon EC2 instances to your load balancer.
6	Review settings.
7	Create your load balancer.

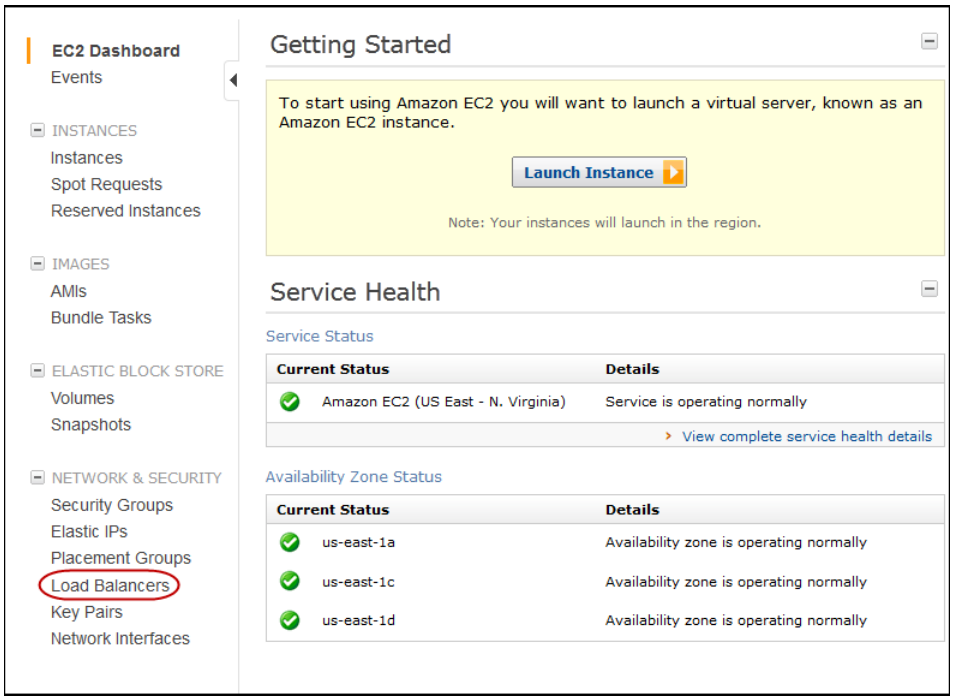
You can choose to create your load balancer in EC2-VPC using the AWS Management Console, the command line interface, or the Query API. If want to use the command line interface, be sure to install the interface. For more information, see Get Set Up with Elastic Load Balancing Interfaces (p. 44).

Using the AWS Management Console

To create a basic internal load balancer in VPC

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

   a. On the Amazon EC2 Getting Started page, in the **EC2 Dashboard** pane, under **NETWORK & SECURITY**, click **Load Balancers**.

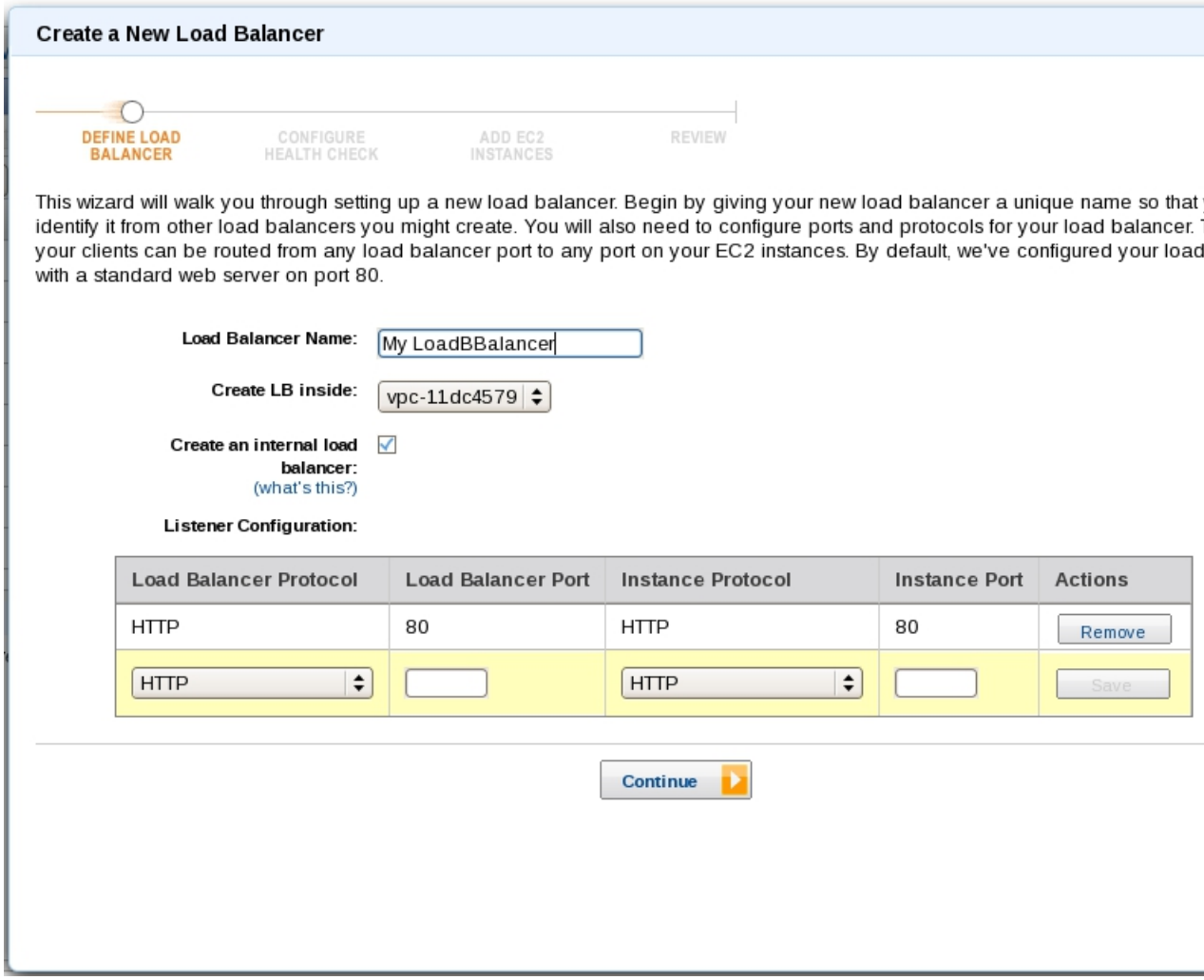


b. On the **Load Balancers** page, click **Create Load Balancer**.

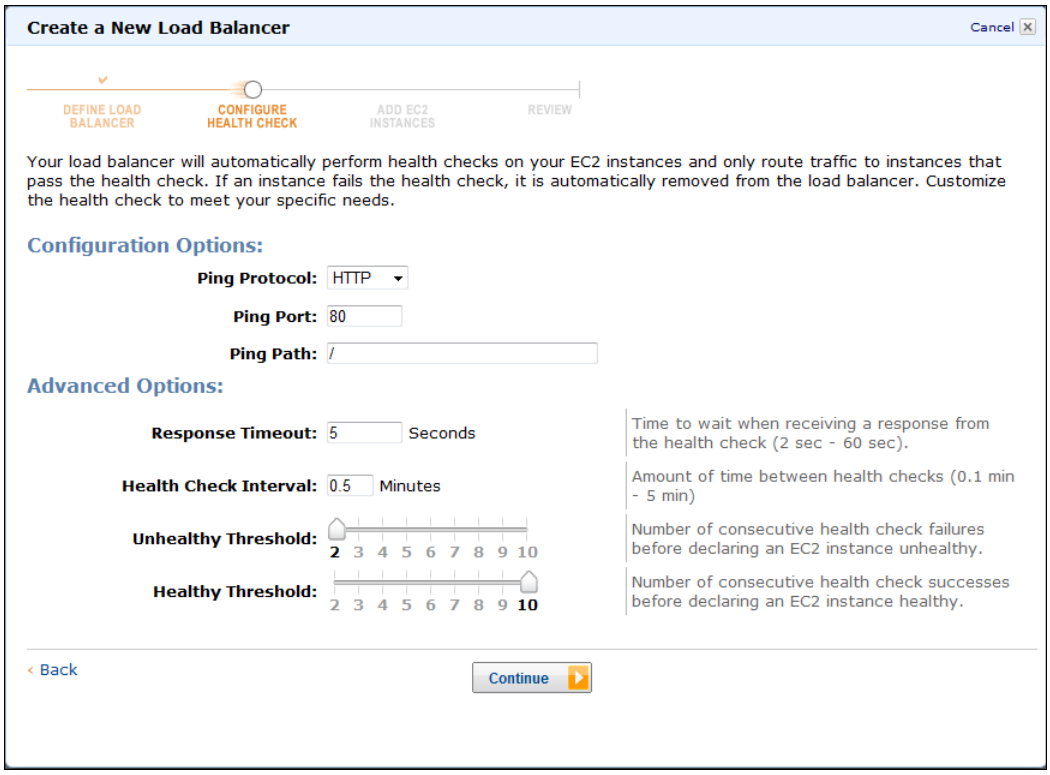
2. On the **DEFINE LOAD BALANCER** page, enter a name for your Amazon VPC load balancer (e.g., MyVPCLoadBalancer).
3. Click the arrow in the **Create LB inside** box and select the Amazon VPC in which you want to create your load balancer.
4. By default, Elastic Load Balancing creates an Internet-facing load balancer with a publicly resolvable DNS name that resolves to public IP addresses. In this example, you'll create an internal load balancer with a publicly resolvable DNS name that resolves to private IP addresses.

   Click **Create an internal load balancer** box.

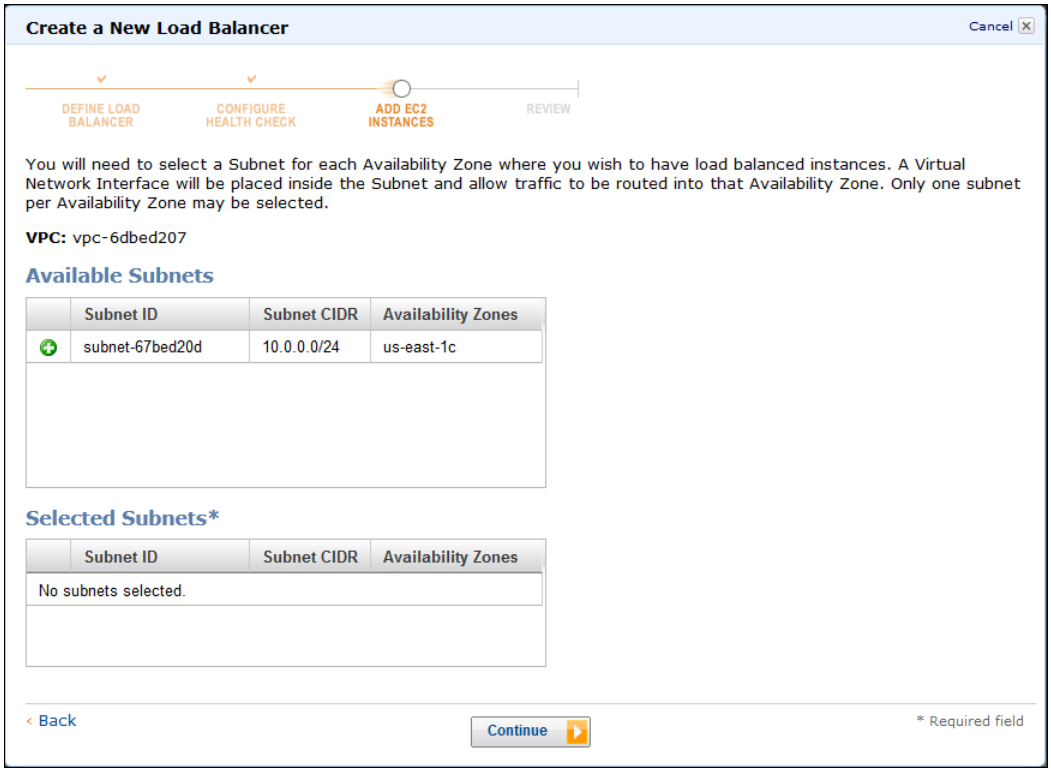
   Note
   Do not select **Create an internal load balancer** if you want the DNS name of your load balancer to resolve to public IP addresses.
5. Leave the **Listener Configuration** set to the default value.



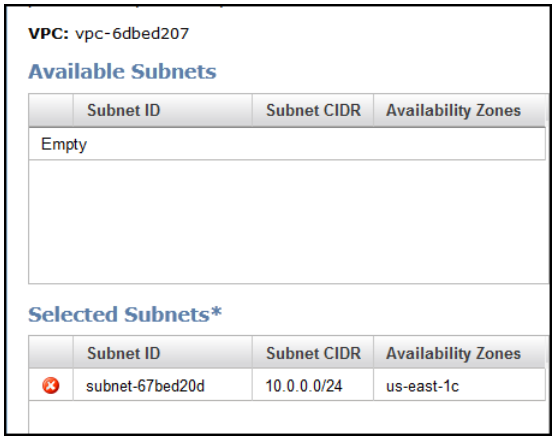
6. Click **Continue** to configure the health check for your instances.
7. Configure the health check settings that your application requires.



8. Click **Continue** to select the subnet in which you want to launch your load balancer instance.
9. 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.

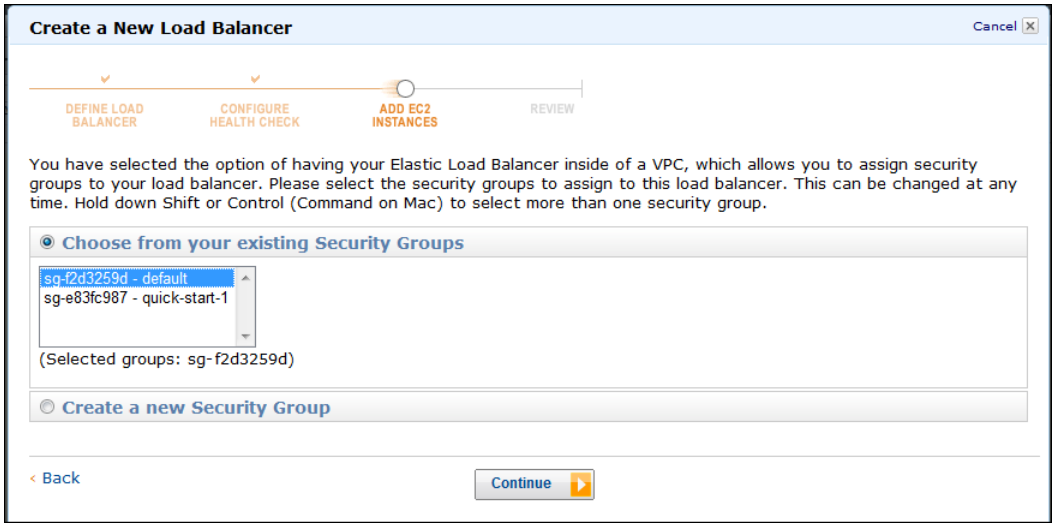


Your selected subnets are displayed in the **Selected Subnets** table.

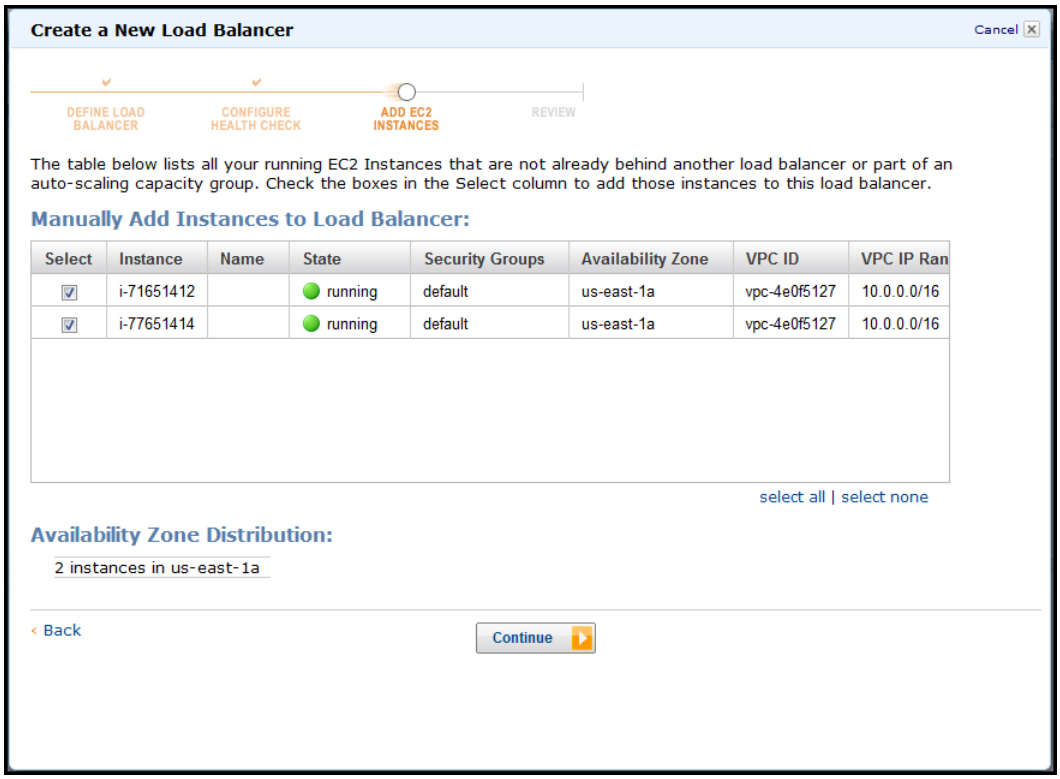


10. Click **Continue** to select security groups to assign to your load balancer.
11. If you use a pre-existing security group, ensure that it allows ingress to the ports that you configured the load balancer to use. If you create a security group in this step, the console will define these ports to 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.



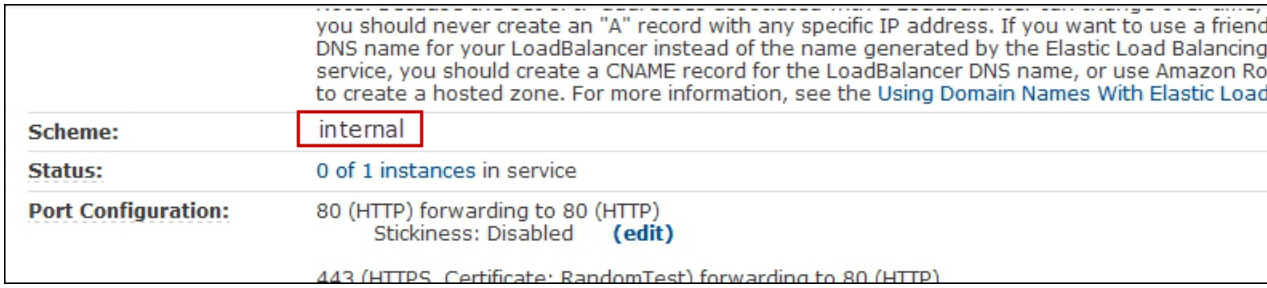
12. Click **Continue** to add running EC2 instances to your load balancer.
13. In the **Manually Add Instances to LoadBalancer** table, check the boxes in the **Select** column to add instances to your load balancer.



Note

When you register a multi-homed instance (an instance that has an elastic network interface (ENI) 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 ENIs, go to Elastic Network Interfaces.

14. Click **Continue** to review your configuration. On the **REVIEW** page, click **Create** to create your load balancer.
15. A confirmation window opens. Click **Close**.
16. When the confirmation window closes, the Load Balancers page opens. Your new load balancer now appears in the list. Select the check box next to your load balancer.
17. A set of tabs opens with details about your new load balancer. Check the description in the row titled **Scheme**. It shows that your newly created load balancer is **internal**.



Using the Query API

By default, Elastic Load Balancing creates an Internet-facing load balancer with a publicly resolvable DNS name that resolves to public IP addresses. You can choose to create an internal load balancer with a DNS name that resolves to private IP addresses.

This example walks you through the process for creating a basic HTTP internal 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 internal load balancer in EC2-VPC

1. Call `CreateLoadBalancer` using the following parameters:

   - *`Subnets`* = `subnet-450f512c`
   - **[Optional]** Use this parameter to create an internal load balancer. You need not specify this parameter if you're creating an Internet-facing load balancer.

     `Scheme` = `internal`
   - *`Listener`*
     - *`Protocol`* = `HTTP`
     - *`InstanceProtocol`* = `HTTP`
     - *`InstancePort`* = `80`
     - *`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 an Amazon 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 (ENI) 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 ENIs, go to Elastic Network Interfaces.

To verify that an internal load balancer was created

1. Call `DescribeLoadBalancers` with the following parameter:

   - *`LoadBalancerName`* = `MyVPCLoadBalancer`

2. The operation returns the description of your load balancer. The description in the `Scheme` field indicates that your newly created load balancer is **internal**.

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

Using the Command Line Interface

By default, Elastic Load Balancing creates an Internet-facing load balancer with a publicly resolvable DNS name that resolves to public IP addresses. You can choose to create an internal load balancer with a DNS name that resolves to private IP addresses.

This example walks you through the process for creating a basic HTTP internal 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 internal load balancer in EC2-VPC

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

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

Note
Use `--scheme` option to create an internal load balancer. You need not specify this option if you're creating an Internet-facing (public) load balancer.

2. Elastic Load Balancing returns the following:

```
DNS-NAME  DNS-NAME
DNS-NAME  internal-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 an Amazon 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 (ENI) 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 ENIs, go to Elastic Network Interfaces.

To verify that an internal load balancer was created

1. Use the `elb-describe-lbs` command as in the following example.

```
PROMPT> elb-describe-lbs  MyVPCLoadBalancer
```

2. Elastic Load Balancing returns the following:

```
LOAD_BALANCER   MyVPCLoadBalancer  internal-MyVPCLoadBalancer-2111276808.us-
east-1a.elb.amazonaws.com
2012-06-04T02:33:20 450Z internal
```

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

Attach Your Load Balancer to a Subnet

Topics

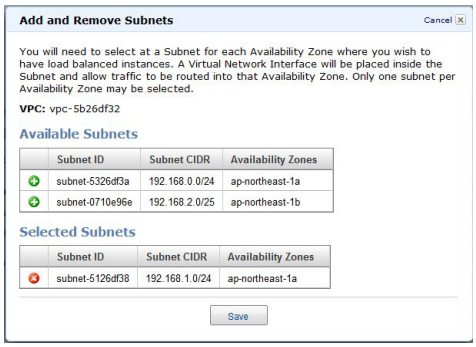
This example walks you through the process of attaching a subnet to an existing load balancer using either the AWS Management Console, the Query API, or the command line interface (CLI).

Before you get started, be sure you've created your Amazon VPC with an Internet gateway and created subnets in each Availability Zone in which you want to load balance your instances. For information on creating an Amazon VPC and subnets for Elastic Load Balancing, see Deploy Elastic Load Balancing in Amazon VPC (p. 110).

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. On the Amazon EC2 Getting Started page, in the **EC2 Dashboard** pane, under **NETWORK & SECURITY**, click **Load Balancers**.
3. On the **Load Balancers** page, select the load balancer that you created for your Amazon 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 to attach a subnet.
7. The **Add and Remove Subnets** page is displayed.



8. In the **Available Subnets** table, click the plus sign in the green circle 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.

Detach Your Load Balancer from a Subnet

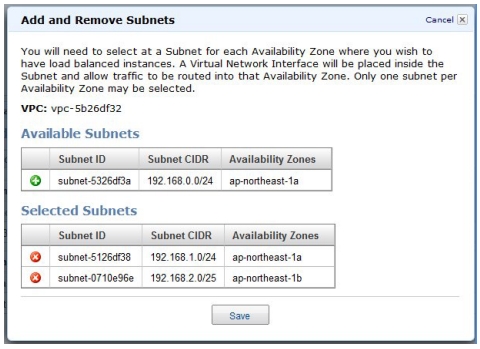
Topics

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 interface (CLI).

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. On the Amazon EC2 Getting Started page, in the **EC2 Dashboard** pane, under **NETWORK & SECURITY**, click **Load Balancers**.
3. On the **Load Balancers** page, select the load balancer that you created for your Amazon 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 to detach a subnet.
7. The **Add and Remove Subnets** page is displayed.



8. In the **Selected Subnets** table, click the x in the red circle 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 a 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 a 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.

Manage Security Groups in Amazon VPC

Topics

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.

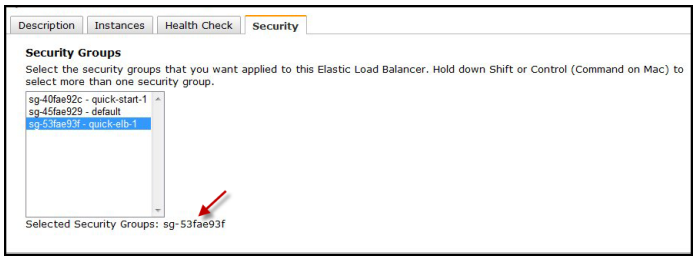
The security groups you've created for Amazon EC2 (i.e., EC2 security groups) are not available to use in your VPC. You must create a separate set of security groups to use in your Amazon VPC (i.e., VPC security groups). The rules you create for a VPC security group can't reference a EC2 security group 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. On the Amazon EC2 Getting Started page, in the **EC2 Dashboard** pane, under **NETWORK & SECURITY**, click **Load Balancers**.
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.



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.

Add a Listener to Your Load Balancer

Topics

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 for both the connections. The Elastic Load Balancing Listener Configurations Quick Reference (p. 67) 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:

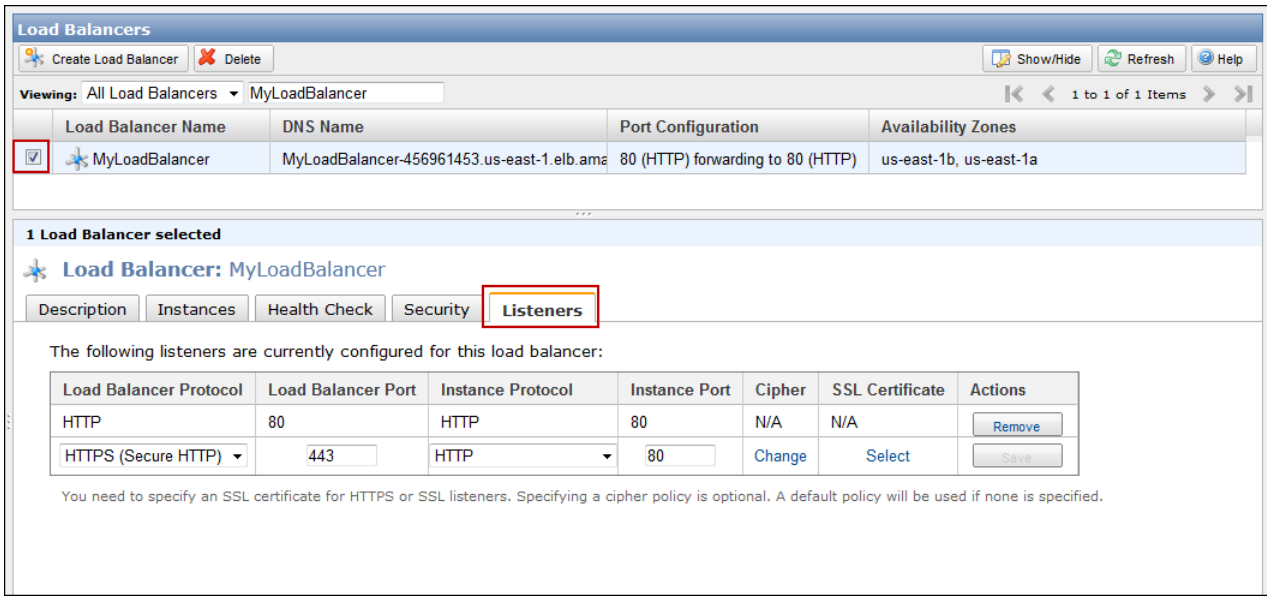
- Created a load balancer with Elastic Load Balancing. For information on how to set up an HTTPS/SSL load balancer with the AWS Management Console, command line interfaces (CLI), or Query API, see Create a Load Balancer with SSL Cipher Settings and Back-End Server Authentication (p. 73). To learn how to set up a HTTP/TCP load balancer with the AWS Management Console, go to the Get Started with Elastic Load Balancing (p. 14).
- 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 AWS Management Console, command line interface (CLI), or the Query API. For information on installing the CLI or the Query API, see Get Set Up with Elastic Load Balancing Interfaces (p. 44).
  - 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 AWS Management Console, 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 AWS Management Console

To add a new listener to 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. On the Amazon EC2 Getting Started page, in the **EC2 Dashboard** pane, under **NETWORK & SECURITY**, click **Load Balancers**.
3. On the **Load Balancers** page, select your load balancer.
4. The bottom pane displays the details of your load balancer.
5. Click the **Listeners** tab.
6. In the **Listeners** table **Load Balancer Protocol** column, select **HTTPS (Secure HTTP)** from the drop-down box. This populates the box in the **Load Balancer Port** column. In the **Instance Protocol** column, select **HTTP** from the drop-down box, then enter port number 80 for the instance port in the **Instance Port** box.

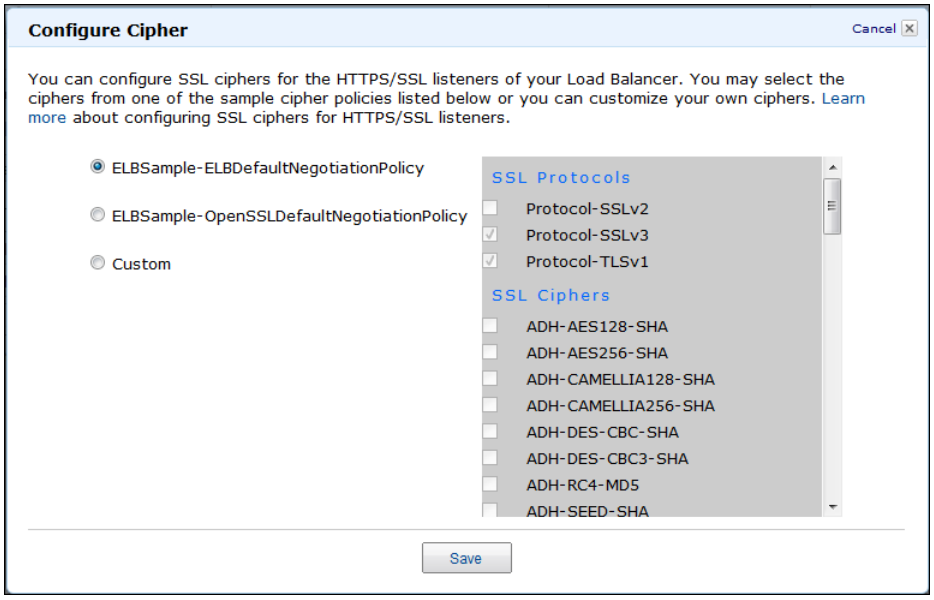


7. The Elastic Load Balancing service provides you with sample cipher policies, **ELBSample-ELBDefaultCipherPolicy** and **ELBSample-OpenSSLDefaultCipherPolicy**. You can select one of the sample policies or customize your own ciphers. A default policy will be used if none is specified.

   a. To specify a cipher policy, select **Change** in the **Cipher** box.
   b. On the **Configure Cipher** page, select one of the sample policies provided.
   c. Or, select **Custom** to customize your own ciphers, 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.

   d. Click **Save**.



8. Click **Select** in the **SSL Certificate** box to specify an SSL certificate to install on your load balancer.

   Note
   To enable HTTPS support for your listeners, you must install an SSL server certificate on your load balancer. The load balancer uses the certificate to terminate and then decrypt requests before sending them to the back-end instances. Elastic Load Balancing uses AWS Identity and Access Management (IAM) to upload your certificate to your load balancer. If you do not have an SSL certificate, go to Creating and Uploading Server Certificates in *Using AWS Identity and Access Management* for instructions on creating and uploading SSL certificates.

9. In the **Configure SSL Certificate** page, select **Choose from your existing SSL Certificates** to use the previously uploaded SSL certificate and select the certificate from the drop-down box.
10. Alternatively, select **Upload a new SSL Certificate** if you have an SSL certificate and want to upload using AWS Identity and Access Management.

    Check if your certificate meets the following criteria:

    - Certificates must follow the X.509 PEM format.
    - The current date must be between the certificate's start and end date.
    - Public and private certificate files must contain only a single certificate.
    - The private key must match the public key that is in the digital server certificate.
    - The private key must be an RSA private key in PEM format, where the PEM header is BEGIN RSA PRIVATE KEY and the footer is END RSA PRIVATE KEY.
    - The private key cannot be encrypted with a password.
    - A certificate chain starts with the immediate signing certificate and is then followed by any intermediaries in order. Intermediaries that are not involved in the trust path must not be included. The trusted root certificate can be optionally included as the last certificate.

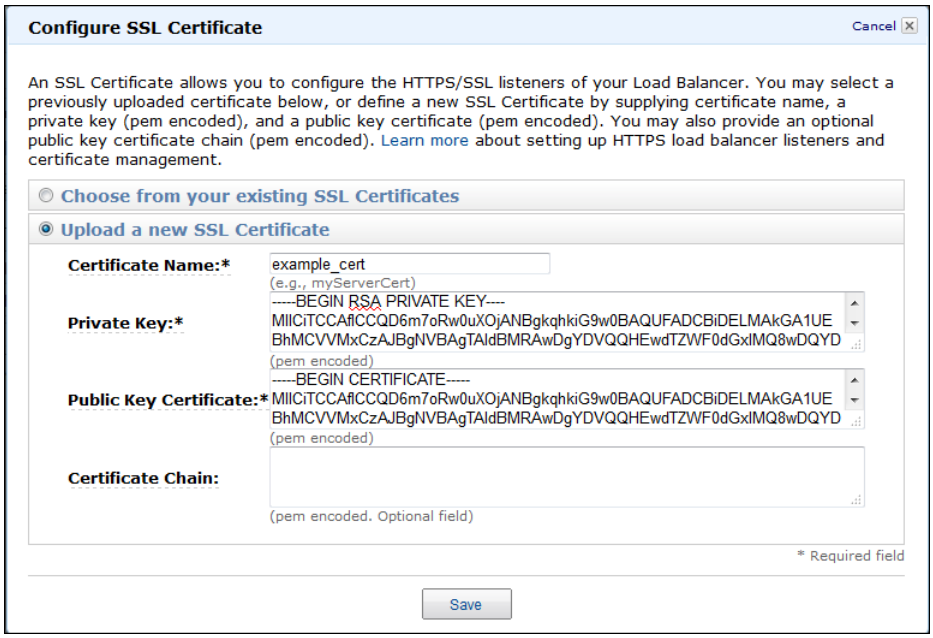
    If your certificate does not meet the criteria, you might get an error when you upload it. Create a new SSL certificate and upload the certificate using AWS Identity and Access Management (IAM). For instructions on creating and uploading the SSL certificate, go to Creating and Uploading Server Certificates in *Using AWS Identity and Access Management*.

    If your certificate meets the criteria, step through the following instructions to continue uploading your 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.

    e. Click **Save**.



11. Click **Save** in the **Actions** box.

Using the Command Line Interface

To enable HTTPS support for your listeners, you must install an SSL server certificate on your load balancer. The load balancer uses the certificate to terminate and then decrypt requests before sending them to the back-end instances. Elastic Load Balancing uses AWS Identity and Access Management (IAM) to upload your certificate to your load balancer. If you do not have an SSL certificate, go to Creating and Uploading Server Certificates in *Using AWS Identity and Access Management* for instructions on creating and uploading SSL certificates.

To add a new listener to your load balancer

1. Get the Amazon Resource Name (ARN) of your SSL certificate.
2. 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"
```

3. 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 enable HTTPS support for your listeners, you must install an SSL server certificate on your load balancer. The load balancer uses the certificate to terminate and then decrypt requests before sending them to the back-end instances. Elastic Load Balancing uses AWS Identity and Access Management (IAM) to upload your certificate to your load balancer. If you do not have an SSL certificate, go to Creating and Uploading Server Certificates in *Using AWS Identity and Access Management* for instructions on creating and uploading SSL certificates.

To add a new listener to your load balancer

1. Get the Amazon Resource Name (ARN) of your SSL certificate.
2. 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`**

3. 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.

Delete a Listener from Your Load Balancer

Topics

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

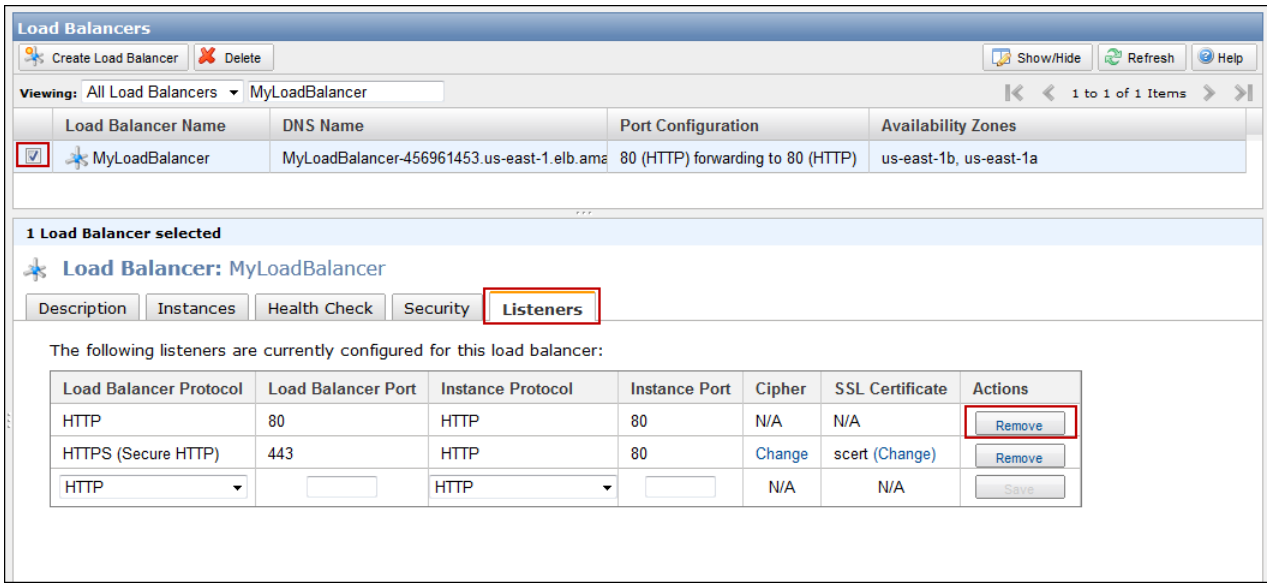
- Created a load balancer with Elastic Load Balancing. For information on how to set up a HTTPS/SSL load balancer with the AWS Management Console, command line interface (CLI), or Query API, see Create a Load Balancer with SSL Cipher Settings and Back-End Server Authentication (p. 73). To learn how to set up a HTTP/TCP load balancer with the AWS Management Console, see Get Started with Elastic Load Balancing (p. 14).
- Installed the Elastic Load Balancing tool that you plan to use to perform load balancing tasks. You can add or delete listeners on your existing load balancer using the AWS Management Console, the command line interface (CLI), or the Query API. For information on installing the CLI, or the Query API, see Get Set Up with Elastic Load Balancing Interfaces (p. 44).
  - 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 a listener from the specified port of your existing load balancer using the AWS Management Console, command line interface (CLI), or the Query API. In this example, you delete a listener from port 80 of your load balancer `MyLoadBalancer`.

Using the AWS Management Console

To delete a listener 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. On the Amazon EC2 Getting Started page, in the **EC2 Dashboard** pane, under **NETWORK & SECURITY**, click **Load Balancers**.
3. On the **Load Balancers** page, select your load balancer.
4. The bottom pane displays the details of your load balancer.
5. Click the **Listeners** tab.
6. Select **Remove** in the **Actions** box of the listener you want to delete.



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-Register and Register 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. On the Amazon EC2 Getting Started page, in the **EC2 Dashboard** pane, under **NETWORK & SECURITY**, click **Load Balancers**.
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. On the Amazon EC2 Getting Started page, in the **EC2 Dashboard** pane, under **NETWORK & SECURITY**, click **Load Balancers**.
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.

Update an SSL Certificate for a Load Balancer

If you are using HTTPS/SSL protocol for your listeners, you might have an SSL server certificate installed on your load balancer. The SSL certificate has to be updated periodically. This section describes how to update an SSL certificate for your HTTPS/SSL load balancer. Before you get started, be sure you've done the following:

- Created a HTTPS/SSL load balancer with Elastic Load Balancing. For information on how to set up an HTTPS/SSL load balancer with the AWS Management Console, the command line interfaces (CLI), or the Query API, see Create a Load Balancer with SSL Cipher Settings and Back-End Server Authentication (p. 73).
- Created a new SSL server certificate to replace the expired server certificate and have uploaded it using the AWS Identity and Access Management (IAM). For information on how to create and upload a SSL certificate, go to Creating and Uploading Server Certificates in *Using AWS Identity and Access Management*.

  All your SSL server certificates are managed by AWS Identity and Access management (IAM). By default, IAM allows 10 server certificates per AWS account. If you try to upload a new server certificate after reaching this limit, you'll get an error. You can request for more certificates using this form - IAM Limit Increase Contact Us Form.
- Installed the Elastic Load Balancing tools that you plan to use to perform load balancing tasks. You can update your SSL certificate installed on your HTTPS/SSL load balancer using the AWS Management Console, command line interface (CLI), or the Query API. For information on installing the CLI, or the Query API, see Get Set Up with Elastic Load Balancing Interfaces (p. 44).
- If you plan to use the command line interface (CLI), you'll also have to 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*.
  - 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.
  - For information on using AWS Identity and Access Management APIs and command line interfaces, go to Using AWS Identity and Access Management.

The following sections include instructions for updating an SSL certificate using the AWS Management Console, the command line interface (CLI), or the Query API.

Using the AWS Management Console

To update an SSL certificate for an HTTPS load balancer

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the Amazon EC2 Getting Started page, in the **EC2 Dashboard** pane, under **NETWORK & SECURITY**, click **Load Balancers**.
3. On the **Load Balancers** page, select your load balancer.
4. The bottom pane displays the details of your load balancer.
5. Click the **Listeners** tab.
6. Click **Change** in the **SSL Certificate** column of the certificate you want to update.
7. On the **Configure SSL Certificate** page, select **Choose from your existing SSL Certificates** to use previously uploaded SSL certificate and select the certificate from the drop-down box.
8. Or, select **Upload a new SSL Certificate** if you have a SSL certificate and want to uploaded it.

   Before you upload, ensure that your certificate meets the following citeria:

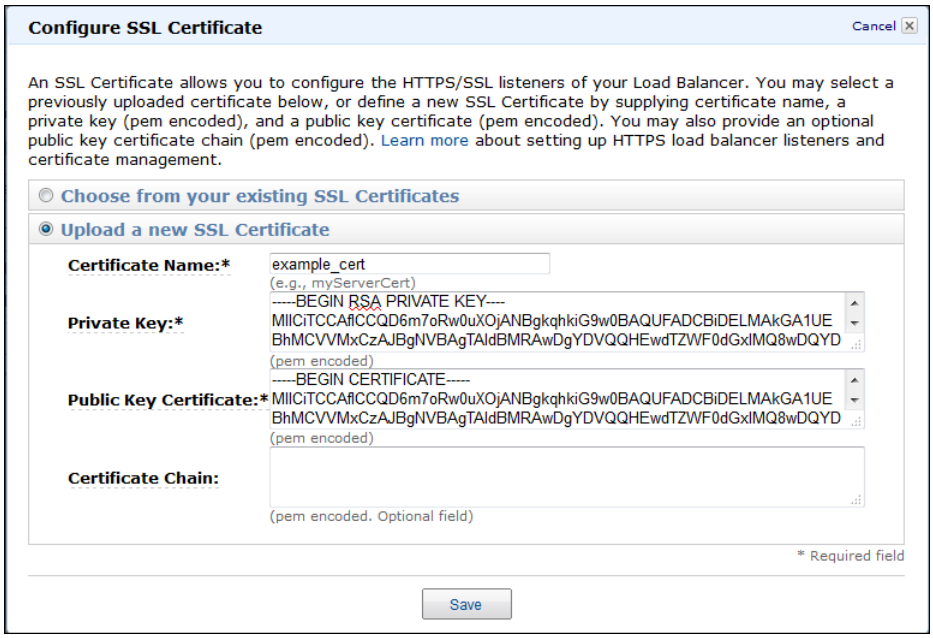
   - Certificates must follow the X.509 PEM format.
   - The current date must be between the certificate's start and end date.
   - Public and private certificate files must contain only a single certificate.
   - The private key must match the public key that is in the digital server certificate.
   - The private key must be an RSA private key in PEM format, where the PEM header is BEGIN RSA PRIVATE KEY and the footer is END RSA PRIVATE KEY.
   - The private key cannot be encrypted with a password.
   - A certificate chain starts with the immediate signing certificate and is then followed by any intermediaries in order. Intermediaries that are not involved in the trust path must not be included. The trusted root certificate can be optionally included as the last certificate.

   If your certificate does not meet the criteria listed in this step, you might get an error when you upload it. Create a new SSL certificate and upload the certificate using AWS Identity and Access Management (IAM). For instructions on creating and uploading the SSL certificate, go to Creating and Uploading Server Certificates in *Using AWS Identity and Access Management*.

   Step through the following instructions to continue uploading your 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.



9. Click **Save**.

Using the Query API

To update an SSL certificate for an HTTPS load balancer

1. If you have a SSL certificate and have uploaded it using the AWS Identity and Access Management (IAM), get the Amazon Resource Name (ARN) of your certificate and go to step 3.
2. If you have a SSL certificate and want to upload it, step through the following instructions:

   a. Check if your certificate meets the following criteria:

   - Certificates must follow the X.509 PEM format.
   - The current date must be between the certificate's start and end date.
   - Public and private certificate files must contain only a single certificate.
   - The private key must match the public key that is in the digital server certificate.
   - The private key must be an RSA private key in PEM format, where the PEM header is BEGIN RSA PRIVATE KEY and the footer is END RSA PRIVATE KEY.
   - The private key cannot be encrypted with a password.
   - A certificate chain starts with the immediate signing certificate and is then followed by any intermediaries in order. Intermediaries that are not involved in the trust path must not be included. The trusted root certificate can be optionally included as the last certificate.

   If your certificate does not meet the criteria, you might get an error when you upload it. Create a new SSL certificate and upload the certificate using AWS Identity and Access Management (IAM). For instructions on creating and uploading the SSL certificate, go to Creating and Uploading Server Certificates in the *Using AWS Identity and Access Management*.

   If your certificate meets the criteria, step through the following instructions to continue uploading your SSL certificate.

b. 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.

3. 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. If you have a SSL certificate and have uploaded it using the AWS Identity and Access Management (IAM), get the Amazon Resource Name (ARN) of your certificate and go to step 3.
2. If you have a SSL certificate and want to upload it using AWS Identity and Access management (IAM), step through the following instructions:

   a. Check if your certificate meets the following criteria:

   - Certificates must follow the X.509 PEM format.
   - The current date must be between the certificate's start and end date.
   - Public and private certificate files must contain only a single certificate.
   - The private key must match the public key that is in the digital server certificate.
   - The private key must be an RSA private key in PEM format, where the PEM header is BEGIN RSA PRIVATE KEY and the footer is END RSA PRIVATE KEY.
   - The private key cannot be encrypted with a password.

- A certificate chain starts with the immediate signing certificate and is then followed by any intermediaries in order. Intermediaries that are not involved in the trust path must not be included. The trusted root certificate can be optionally included as the last certificate.

If your certificate does not meet the criteria, you might get an error when you upload it. Create a new SSL certificate and upload the certificate using AWS Identity and Access Management (IAM). For instructions on creating and uploading the SSL certificate, go to Creating and Uploading Server Certificates in *Using AWS Identity and Access Management*.

If your certificate meets the criteria, step through the following instructions to continue uploading your SSL certificate.

b. Enter the IAM 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/testCertASCACexampleKEZUQ4K
```

c. 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
```

Configure Custom Domain Name for Your Load Balancer

Topics

Each Elastic Load Balancing load balancer that you create has an automatically created Domain Name System (DNS) name. Typically, the DNS name includes the name of the AWS region in which the load balancer is created. For example, if you create a load balancer named 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 just have to paste the DNS name generated by Elastic Load Balancing into the address field of an Internet-connected web browser to connect to your load balancer.

If you'd rather use a user-friendly domain name, such as www.example.com, instead of the load balancer DNS name, you can create a custom domain name and then associate the custom domain name with the load balancer DNS name. When a request is placed to your load balancer using the custom domain name that you created, it resolves to the load balancer DNS name.

To use a custom domain name for your load balancer instance, you'll have to first register your domain name with a DNS service provider.

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 as your custom domain name, you can create subdomain names such as foo.bar.example.com, foo.myLB.example.com, and so on. This set of a domain and its subdomain names is called a zone. A domain name that you reserve, such as example.com, is called the zone apex because it sits at the top of the zone's hierarchy.

Associating Your Custom Domain Name with Your Load Balancer Name

After you register your custom domain name, you have two ways to associate your custom domain name with the load balancer DNS name.

Option 1: Create a canonical name (CNAME) record for your zone with your existing domain name provider. A CNAME record specifies that a domain name is an alias of another CNAME domain name. For example, the following CNAME record associates an alias, www.foo.example.com, with a canonical name, the DNS name of an Elastic Load Balancing instance.

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

For more information on CNAME records, go to the Wikipedia article http://en.wikipedia.org/wiki/CNAME_record.

The creation of CNAME records is a simple process. Many domain name registrars provide self-service tools that you can use to create the CNAME record yourself. However, 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 foo.example.com subdomain name, but not for the example.com zone apex.

You can use the next option if you want to associate a zone apex with your load balancer DNS name.

Option 2: Create a domain using Amazon Route 53 as the DNS service. Amazon Route 53 stores information about your domain in 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 your domain name, including the subdomain names within the domain and mappings between names and IP addresses. For more information about Amazon Route 53, go to What is Route 53 and How Does it Work?

Use this option to associate a zone apex with your load balancer DNS name. You'll use Amazon Route 53 to create a hosted zone for your domain (for example, example.com), and then create alias resource record sets. An alias resource record set contains a pointer to a resource record set that contains your DNS resource records. For example, an alias resource record set for your domain, *example.com*, can point to the DNS name of your Elastic Load Balancing load balancer instance *myLB-1234567890.us-east-1.elb.amazonaws.com*. After you create a hosted zone, you can also create alias resource record sets to associate subdomain names with your Elastic Load Balancing instance.

DNS Failover: When you use Route 53 to create and associate a custom domain name with your load balancer you have an option to enable DNS failover for your load balancer. If DNS failover is enabled, Route 53 responds to the queries to your alias record set, *example.com* based on the health checks of the associated primary and secondary load balancer instances.

If you plan to enable DNS failover for your load balancers, see Configure DNS Failover for Your Load Balancer (p. 149).

This section describes how to associate your Elastic Load Balancing instance with a custom domain name using Option 1 or Option 2.

Prerequisite

Before you start associating a load balancer DNS name with a custom domain name, you first need to create a load balancer. For information on creating a basic load balancer, see Get Started with Elastic Load Balancing (p. 14) For information on creating a load balancer with custom settings, see Create a Load Balancer with SSL Cipher Settings and Back-End Server Authentication (p. 73).

> **Note**
> The time-to-live (TTL) for an Elastic Load Balancing DNS entry is set to 60 seconds. This setting ensures that IP addresses can be re-mapped quickly to respond to events that cause Elastic Load Balancing to scale up or down.

Option 1: Create a CNAME Record for your subdomain and load balancer

1. Register your custom domain name with your DNS provider. For a list of registrar websites you can use to register your domain name, go to ICANN. Wait for your registrar to notify you that your domain name is successfully registered.
2. Create a subdomain name to associate with your load balancer DNS name. For example, if your custom domain name is example.com, you can create a subdomain name such as foo.mylb.example.com.
3. Retrieve the public DNS name of your load balancer. You can use AWS Management Console, the Elastic Load Balancing Query API, or the Elastic Load Balancing command line interface (CLI) to retrieve your load balancer DNS name.

   To use the AWS Management Console to retrieve the public DNS name of your load balancer

   a. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
   b. On the Amazon EC2 Getting Started page, in the **EC2 Dashboard** pane, under **NETWORK & SECURITY**, click **Load Balancers**.

c. On the **Load Balancers** page, select your load balancer.
d. The bottom pane displays the details of your load balancer.
e. Make a note of the DNS name of your load balancer.

4. **To use the Elastic Load Balancing Query API to retrieve the public DNS name of your load balancer**

   a. Call DescribeLoadBalancers with the following parameter:

   ```
   LoadBalancerName = your load balancer name
   ```

   The operation returns the DNS name of your load balancer along with other details.
   b. Make a note of the DNS name.

5. **To use the Elastic Load Balancing CLI to retrieve the public DNS name of your load balancer**

   a. You'll need to install the Elastic Load Balancing CLI tool before you can use the following command. If you haven't yet installed the tool, go to Installing the Command Line Interface (p. 45) for instructions on downloading and installing the CLI tool.
   b. Enter the command `elb-describe-lbs`, as in the following example.

   ```
   PROMPT> elb-describe-lbs your load balancer name
   ```

   The command returns the DNS name of your load balancer along with other details.
   c. Make a note of the DNS name.

6. Create a CNAME record to associate your custom subdomain name with your load balancer DNS name, as in the following example:

   Note
   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.

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

Option 2: Create an Amazon Route 53 hosted zone for your load balancer

1. Register your custom domain name with your DNS provider. For a list of registrar websites you can use to register your domain name, go to ICANN.org. Wait for your registrar to notify you that your domain name is successfully registered.
2. Create a hosted zone for your custom domain name using Amazon Route 53. For detailed instructions, go to Creating a Hosted Zone in *Amazon Route 53 Developer Guide*.

   As with other AWS products, there are no contracts or minimum commitments for using Amazon Route 53—you pay only for the hosted zones that you configure and the number of queries that Route 53 answers. For more information, see Route 53 Pricing.

3. For each hosted zone you create, Route 53 automatically creates four name server (NS) records. These four name servers are called the delegation set. Before the Domain Name System will start to route queries for your domain to Route 53 name servers, you must update your registrar's or your DNS service's name server records, to point to the Route 53 name servers.

   Follow the instructions in Getting the Name Servers for a Hosted Zone to get the four name servers assigned to your hosted zone.

   Follow the instructions in Name Server (NS) Records to update your registrar's or your DNS service's name server records, to point to the Route 53 assigned name servers.

4. Create an alias resource record set you will use to associate your zone apex with your Elastic Load Balancing instance DNS name. If you want to enable DNS failover for your load balancers, see Create Alias Resource Records Sets and Enable DNS Failover (p. 150).

   To create the alias resource record set you can use the Amazon Route 53 console, Amazon Route 53 API, or the Elastic Load Balancing command line interface (CLI).

   Note
   Use either the Route 53 console or the Route 53 API if you want to enable DNS failover for your load balancer. Elastic Load Balancing CLI does not support DNS failover options at this time.

   To use Amazon Route 53 console or Amazon Route 53 Query API

   For detailed instructions on using the Amazon Route 53 console or the Amazon Route 53 Query API to create an alias resource record set, go to Creating Alias Resource Record Sets in the *Amazon Route 53 Developer Guide*.

   To use the Elastic Load Balancing command line interface

   a. Before you get started, be sure you've installed the Elastic Load Balancing CLI tools. For more information, see Installing the Command Line Interface (p. 45) for instructions on downloading and installing the CLI tools. For detailed descriptions of the Elastic Load Balancing commands, see the Elastic Load Balancing Quick Reference Card.

   b. Enter the `elb-associate-route53-hosted-zone` command, as in the following example, to associate a zone apex with an Elastic Load Balancing instance.

      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
      ```

      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. For instructions on getting the hosted zone ID of your custom domain name, go to Listing the Hosted Zones for an AWS Account.

      For more information about the *`weight`* parameter, go to Setting Up Weighted Resource Record Sets in the *Amazon Route 53 API Reference Guide*.

      Note
      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*.

   c. 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 foo.bar.example.com your Amazon Route 53 hosted zone with ID number Z123456789.

```
elb-associate-route53-hosted-zone myLoadBalancer --rr-name foo.bar.ex
ample.com --hosted-zone-id Z123456789 --weight 100
```

Note
The `elb-associate-route53-hosted-zone` command works only with AWS secret key authentication. Unlike other Elastic Load Balancing CLI commands, this Elastic Load Balancing command does not work with X.509 certificate and RSA private key credentials.

Disassociating Your Custom Domain Name From Your Load Balancer Name

You can disassociate your custom domain name from a load balancer instance by first deleting the resource record sets in your hosted zone and then deleting the hosted zone.

1. **Delete the alias resource record sets in your Amazon Route 53 hosted zone**

   You can use Amazon Route 53 console, Amazon Route 53 API, or the Elastic Load Balancing command line interface (CLI) to delete alias resource record sets in your hosted zone.

   ### Using the Amazon Route 53 Console

   - For information on using Amazon Route 53 console, go to Creating, Changing, and Deleting Resource Record Sets in the *Amazon Route 53 Developer Guide*.

   ### Using the Amazon Route 53 Query API

   - For information on using Amazon Route 53 Query API, scroll down to the **Creating, Changing, and Deleting Resource Record Sets Using the Route 53 API** section in the Creating, Changing, and Deleting Resource Record Sets topic in the *Amazon Route 53 Developer Guide*.

   ### Using the Elastic Load Balancing command line interface

   - 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*.

Note
The `elb-disassociate-route53-hosted-zone` command works only with AWS secret key authentication. Unlike other Elastic Load Balancing CLI commands, this new Elastic Load Balancing command does not work with X.509 certificate and RSA private key credentials.

2. **Delete the hosted zone associated with your load balancer DNS name**

   You can use the Amazon Route 53 console or the Amazon Route 53 Query API to delete the hosted zone associated with your load balancer DNS name. For more information, go to Deleting a Hosted Zone in the *Amazon Route 53 Developer Guide*.

Configure DNS Failover for Your Load Balancer

You can provide high availability and redundancy for your applications running behind Elastic Load Balancing by enabling Amazon Route 53 Domain Name System (DNS) failover for your load balancers. Amazon Route 53 is a DNS service that provides reliable routing to your infrastructure. When DNS failover is enabled, Route 53 uses a health-checking feature to determine the availability of the applications running behind the load balancers. If there are no healthy EC2 instances registered with the load balancer or if the load balancer itself is unhealthy, Route 53 will route traffic away from the unhealthy load balancer, and to the other available healthy load balancers.

For information on Amazon Route 53, see What is Route 53 and How Does it Work?.

We will use an example to illustrate how Route 53 DNS failover works with your load balancers. Suppose you have a web application for www.example.com, and you want to have redundant applications running behind two load balancers residing in two different regions, US East and US West. You want the traffic for www.example.com to be primarily routed to the load balancer in US East, and use the load balancer in US West as a backup during failures. In this example, if you use Route 53 DNS failover, you can specify how you want the DNS service to route queries to your domain name, www.example.com. When you enable DNS failover, you must specify your primary and your secondary (backup) load balancers. Enabling the failover routing mechanism tells Route 53 to direct traffic to a primary load balancer, if available, or otherwise reroute the traffic to a secondary load balancer.

To configure DNS failover, you will need to associate your domain name, www.example.com, with your load balancers. Each load balancer has an automatically generated DNS name, such as myLB-1234567890.us-east-1.elb.amazonaws.com. Using Route 53, you will create records for your domain name, www.example.com, pointing to the load balancer DNS name, myLB-1234567890.us-east-1.elb.amazonaws.com.

In this section we walk you through the process of enabling Amazon Route 53 DNS failover for your load balancers running in two different regions.

Prerequisites

Before you can enable DNS failover for your load balancers, you first need to create your load balancers. Be sure to create a primary and a secondary load balancer, preferably in two separate regions. For information on creating a basic load balancer, see Get Started with Elastic Load Balancing (p. 14).

If you do not already have a custom domain name, register your domain name with your DNS provider. For a list of registrar websites you can use to register your domain name, go to ICANN.org. Wait for your registrar to notify you that your domain name is successfully registered.

Use the Console to Configure DNS Failover for Your Load Balancers

The following steps outline how to configure DNS failover for your load balancers using the Route 53 console.

1. Create a hosted zone for your custom domain name.
2. Update the DNS Name Server Records to Point to Route 53 Name Servers.
3. Create alias record sets for your primary and secondary load balancers.

Create an Amazon Route 53 Hosted Zone for Your Custom Domain Name

The hosted zone contains information about your domain name, including the subdomain names and mappings.

Follow the instructions in Creating a Hosted Zone in the *Amazon Route 53 Developer Guide* to create a hosted zone for your custom domain name name, www.example.com.

As with other AWS products, there are no contracts or minimum commitments for using Amazon Route 53—you pay only for the hosted zones that you configure and the number of queries that Route 53 answers. For more information, see Route 53 Pricing.

Update the DNS Name Server Records to Point to Route 53 Name Servers

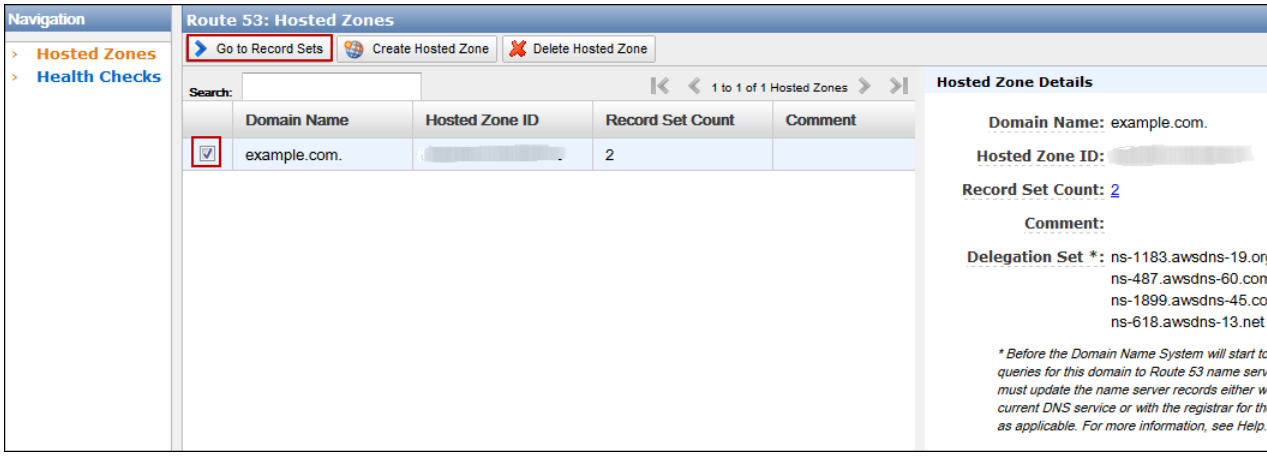
For each hosted zone you create, Route 53 automatically creates four name server (NS) records. Before the DNS will start to route queries for your domain to Route 53 name servers, you must update your registrar's or your DNS service's name server records, to point to the Route 53 name servers.

1. Follow the instructions in Getting the Name Servers for a Hosted Zone to get the name servers assigned to your hosted zone.
2. Follow the instructions in Name Server (NS) Records to update your registrar's or your DNS service's name server records to point to the Route 53 assigned name servers.

Create Alias Resource Records Sets and Enable DNS Failover

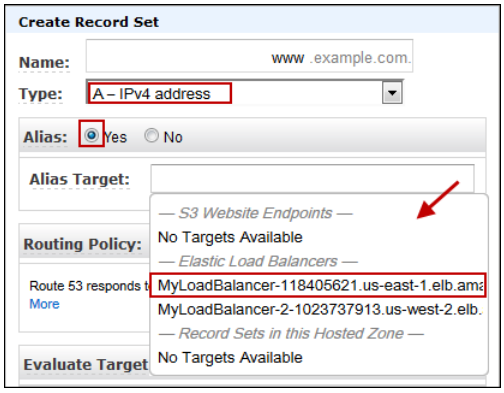
After you have created a hosted zone for your domain, you need to create primary and secondary alias resource record sets to tell the DNS how you want traffic to be routed for your domain.

1. Open the Amazon Route 53 console at https://console.aws.amazon.com/route53/.
2. Click the row of your hosted zone, and click **Go to Record Sets**.

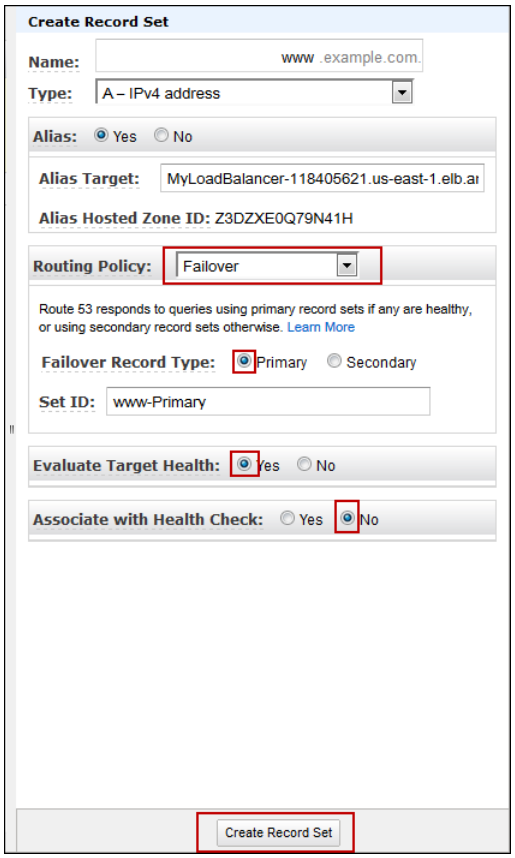


3. Click **Create Record Set**.
4. In the **Name** box, type the domain name for this record set, or use the default value.
5. In the **Type** box, select **A - Ipv4 address**.
6. For **Alias**, click **Yes**

7. Click the **Alias Target** field, and choose your primary load balancer from the list.



8. The value of **Alias Hosted Zone ID** appears automatically based on the value that you selected or entered for **Alias Target**.
9. In the **Routing Policy:** drop-down box, select **Failover**.
10. For **Failover Record Type:**, select **Primary**.
11. For **Set ID:** enter an ID for the record set or use the default value.
12. For **Evaluate Target Health**, select **Yes**.
13. For **Associate with Health Check**, select **No**.
14. Click **Create Record Set**.



15. Follow the same steps to create another alias record set for your secondary load balancer, with the following exceptions:

    - In the **Alias Target** field, choose your secondary load balancer from the list.
    - For **Failover Record Type:**, select **Secondary**.
    - For **Evaluate Target Health**, select **Yes** if you want Route 53 to evaluate the health of the secondary load balancer. If the secondary load balancer is unhealthy, Route 53 will route back to the primary load balancer. If **Evaluate Target Health** is set to **No**, Route 53 will assume that the secondary load balancer is healthy and route traffic to the secondary load balancer whenever the primary load balancer is unhealthy.

For information on how to set up advanced DNS failover configurations, see Managing Resource Availability with Route 53 DNS Failover in the *Amazon Route 53 Developer Guide*.

Create Sticky Sessions

Topics

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.

Enable 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 cookie is automatically updated with each request. 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:

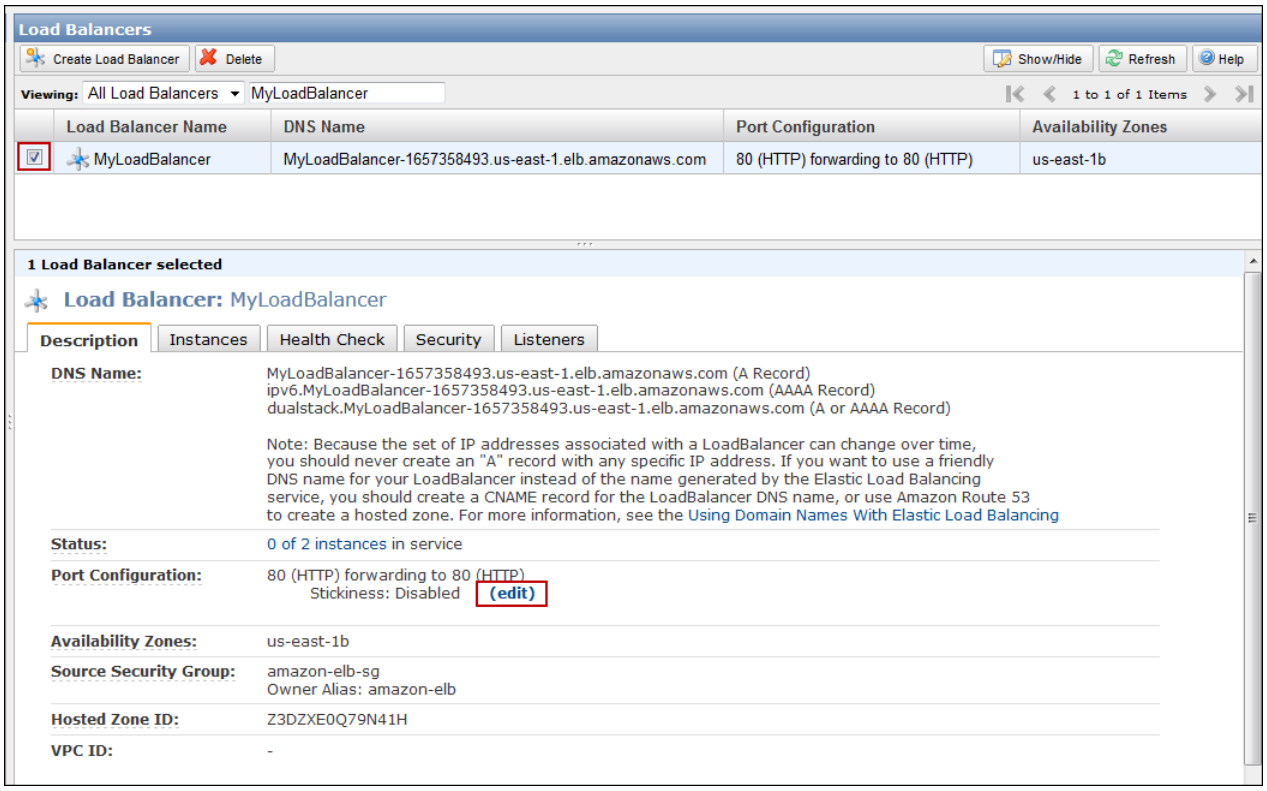
- Created a load balancer with Elastic Load Balancing. For information on how to set up a HTTPS/SSL load balancer with the AWS Management Console, command line interface (CLI), or Query API, see Create a Load Balancer with SSL Cipher Settings and Back-End Server Authentication (p. 73). To learn how to set up a HTTP load balancer with the AWS Management Console, see Get Started with Elastic Load Balancing (p. 14).
- Installed the Elastic Load Balancing tool that you plan to use to perform load balancing tasks. You can create a stickiness policy using the AWS Management Console, the command line interface (CLI), or the Query API. For information on installing the CLI or the Query API, see Get Set Up with Elastic Load Balancing Interfaces (p. 44).
  - 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 AWS Management Console

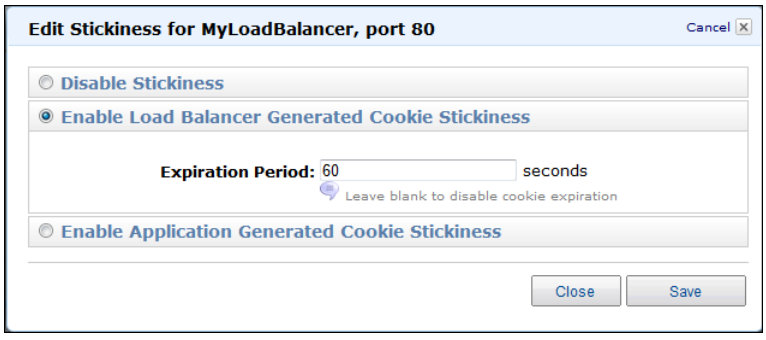
To enable duration-based sticky sessions for a load balancer

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the Amazon EC2 Getting Started page, in the **EC2 Dashboard** pane, under **NETWORK & SECURITY**, click **Load Balancers**.

3. On the **Load Balancers** page, select your load balancer.
4. The bottom pane displays the details of your load balancer.
5. Click **(edit)** in the **Port Configuration:** row.



6. On the **Edit Stickiness for *name of your load balancer*, *port number of your load balancer*** page, click **Enable Load Balancer Generated Cookie Stickiness**.
7. In the **Expiration Period** box, enter the cookie expiration period. This example creates a cookie stickiness policy with a cookie expiration period of 60 seconds.
8. Click **Save**.



9. The **Port Configuration**row in the **Description** pane shows the newly created cookie stickiness policy.

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
```

Enable 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. The load balancer stickiness cookie does not update with each request. If the application cookie is explicitly removed or expires, the session stops being sticky until a new application cookie is issued.

If an application server fails or is removed, the load balancer will try to route the sticky session to another healthy application server. The load balancer will try to stick to new healthy application server and continue routing to currently stick application server even after the failed application server comes back. However, it is up to the new application server on how it'll respond to a request which it has not seen previously.

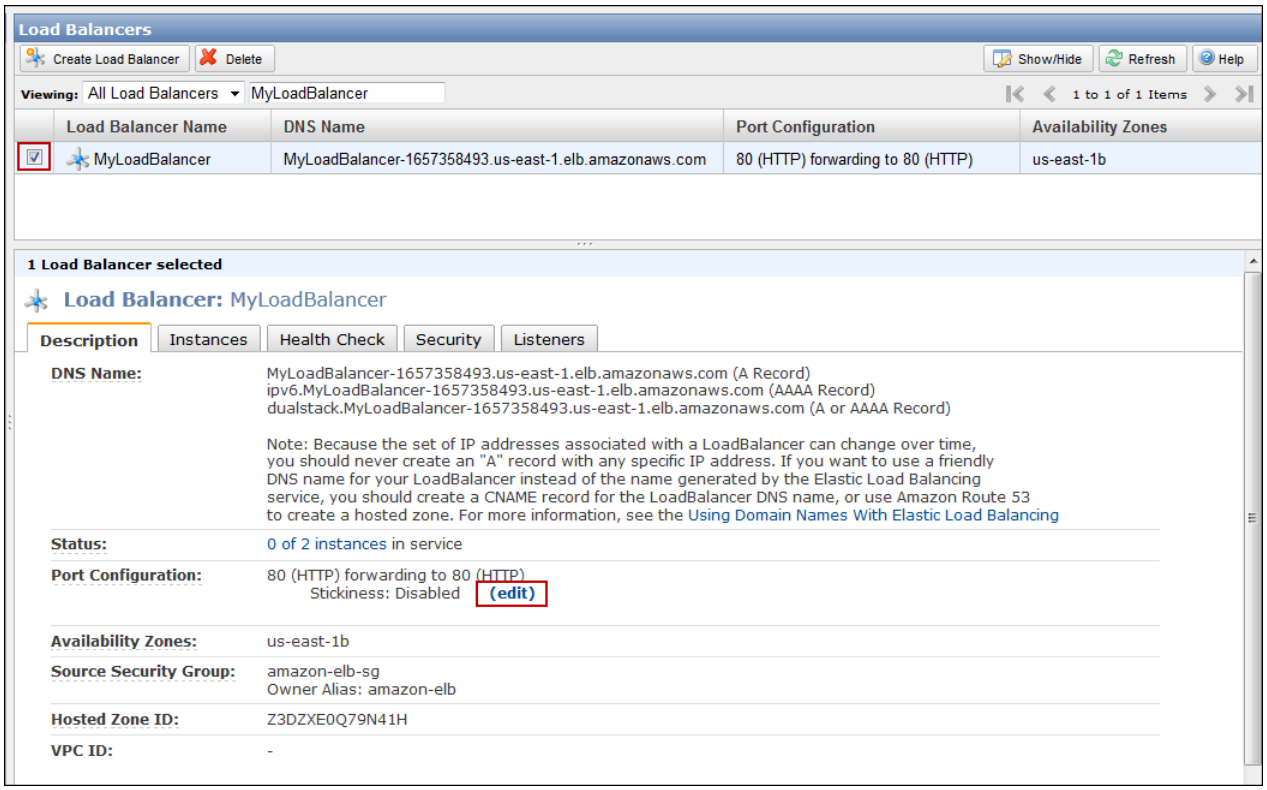
In this example, you configure a load balancer for session stickiness when the life of the session follows that of an application-generated cookie. Before you get started, be sure you've done the following:

- 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 Create a Load Balancer with SSL Cipher Settings and Back-End Server Authentication (p. 73). To learn how to set up a HTTP load balancer with the AWS Management Console, see Get Started with Elastic Load Balancing (p. 14).
- Installed the Elastic Load Balancing tool that you plan to use to perform load balancing tasks. You can create a session stickiness policy using the AWS Management Console, the command line interface (CLI) or the Query API. For information on installing the CLI or the Query API, see Get Set Up with Elastic Load Balancing Interfaces (p. 44).
  - 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*

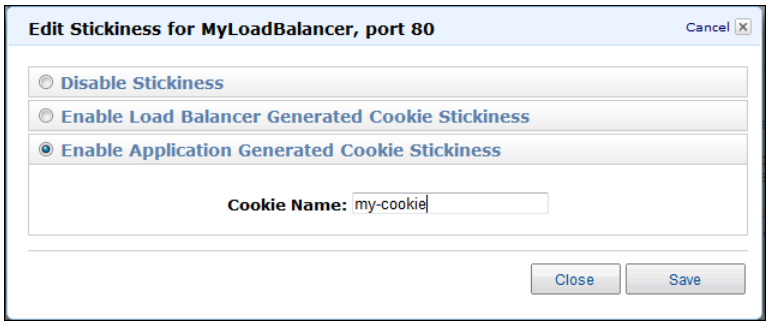
Using the AWS Management Console

To Enable Application-Controlled Session Stickiness

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the Amazon EC2 Getting Started page, in the **EC2 Dashboard** pane, under **NETWORK & SECURITY**, click **Load Balancers**.
3. On the **Load Balancers** page, select your load balancer.
4. The bottom pane displays the details of your load balancer.
5. Click **(edit)** in the **Port Configuration:** row.



6. On the **Edit Stickiness for *`name of your load balancer`*, *`port number of your load balancer`*** page, click **Enable Application Generated Cookie Stickiness**.
7. In the **Cookie Name** box, enter the the name for your cookie, for example `my-cookie`.
8. Click **Save**.



9. The **Port Configuration**row in the **Description** pane shows the newly created cookie stickiness policy.

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 policy.

```
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
```

Monitor Your Load Balancer Using Amazon CloudWatch

Topics

Elastic Load Balancing provides data to Amazon CloudWatch about your load balancers and your back-end application instances. 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 Availability Zone over a specified time period.

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. A CloudWatch alarm watches over a specified metric and initiates an action if the metric goes outside of the specified range. An action can be a notification sent to you.

This section defines the metrics that Elastic Load Balancing sends to Amazon CloudWatch. It also explains how to view the metrics, 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 load balancers to Amazon CloudWatch. By default, Amazon CloudWatch uses these metrics to provide detailed monitoring of the load balancers. 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.
> If there are requests flowing through the load balancer, Elastic Load Balancing will measure and send metrics for that load balancer in 60-second intervals.

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 request leaves the load balancer 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
`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

Metric	Description
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. This response count does not include any responses that were generated by back-end instances. 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
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 dimensions for Elastic Load Balancing that are listed in the table in this section. 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 the count of backend instances change over time.

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

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*.

View Metrics

You can view the CloudWatch metrics for your Elastic Load Balancing load balancers directly in the EC2 console's **Load Balancer** page. The load balancer metrics are displayed as monitoring graphs. The following metrics are available in the EC2 console:

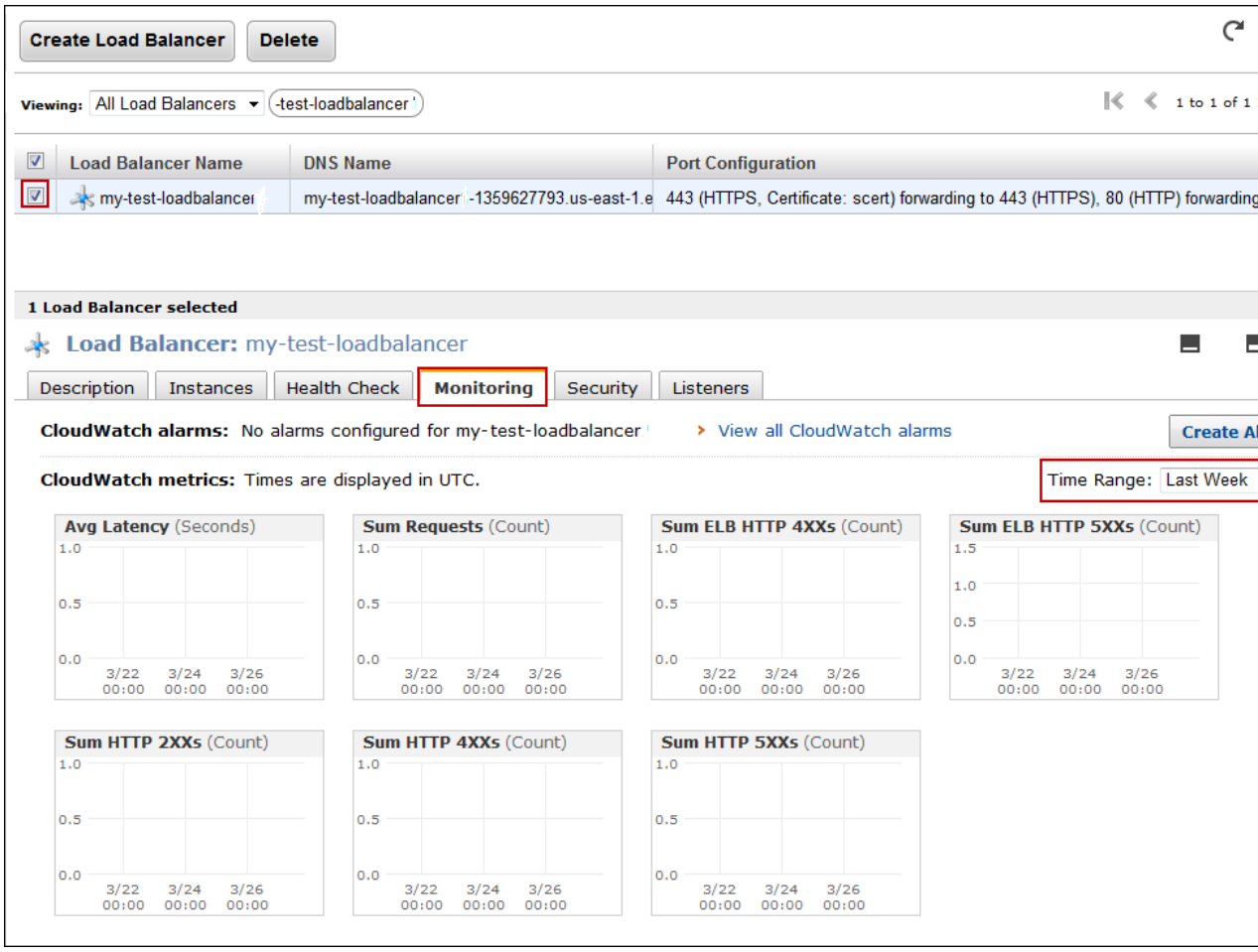
- Average Latency
- Sum Requests
- Sum ELB HTTP 4XXs
- Sum ELB HTTP 5XXs
- Sum HTTP 2XXs
- Sum HTTP 4XXs
- Sum HTTP 5XXs

To view the metrics 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. On the Amazon EC2 console Getting Started page, in the **EC2 Dashboard** pane, under **NETWORK & SECURITY**, click **Load Balancers**.
3. On the **Load Balancers** page, select the check box next to your load balancer.

   The bottom pane displays the details of your load balancer.
4. Click the **Monitoring** tab.

   The bottom pane displays the graphs of metrics sent by Elastic Load Balancing for the selected load balancer.
5. If you want to filter the results by time, select the time range in the **Time Range** drop-down box.



6. Click on an individual graph to get a larger view of the individual metric.

The `my-test-loadbalancer` load balancer used in this example is inactive and is used for display purposes only. The monitoring graphs will show data points if the load balancer is active and receiving requests.

You can also view metrics for your load balancer using the Amazon CloudWatch console, command line interface (CLI), or the Query API. For information on using the CloudWatch console to view metrics, see Viewing Your AWS Metrics with Amazon CloudWatch. To view metrics using the command line interface, use the mon-list-metrics command. To view metrics using the Query API, use the ListMetrics action.

Create Alarms

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. Amazon SNS is a web service that enables applications, end users, and devices to instantly send and receive notifications. For more information, see Get Started with Amazon SNS.

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

- *OK*—This is the state the alarm is in when the value of the metric remains within the range you've specified.
- *ALARM*—This is the state the alarm goes to when the value of the metric goes out of the range you've specified and remains outside of the range for a specified time duration.
- *INSUFFICIENT_DATA*—When the alarm is in this state, it either means that the metric is not yet available or not enough data is available for the metric to determine the alarm state.

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

You can create alarms for your load balancer using either the Elastic Load Balancing wizard in the EC2 console or the Amazon CloudWatch console. This section walks you through the steps for creating an alarm using the Elastic Load Balancing wizard. For information on creating a load balancer alarm using the Amazon CloudWatch console, see Send Email Based on Load Balancer Alarm.

To create an alarm for 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. On the Amazon EC2 console Getting Started page, in the **EC2 Dashboard** pane, under **NETWORK & SECURITY**, click **Load Balancers**.
3. On the **Load Balancers** page, select the check box next to the load balancer for which you want to create an alarm.

   The bottom pane displays the details of your load balancer.
4. On the **Monitoring** tab in the bottom pane, click **Create Alarm**.
5. In the **Create Alarm** dialog box, set the criteria for your alarm. In this example, we'll set an alarm if the load balancer's latency is above 120 seconds for 1 consecutive period of 5 minutes.
6. The check box next to **Send a notification to** is selected by default.

   If you've already created a SNS topic and want to use it, select your topic from the drop-down box. Skip the next step.

   If you do not have a SNS topic, click **create topic**.
7. In the **With these recipients** box, enter the email addresses of the recipients you want to notify. You can enter up to 10 email addresses, each separated by a comma.

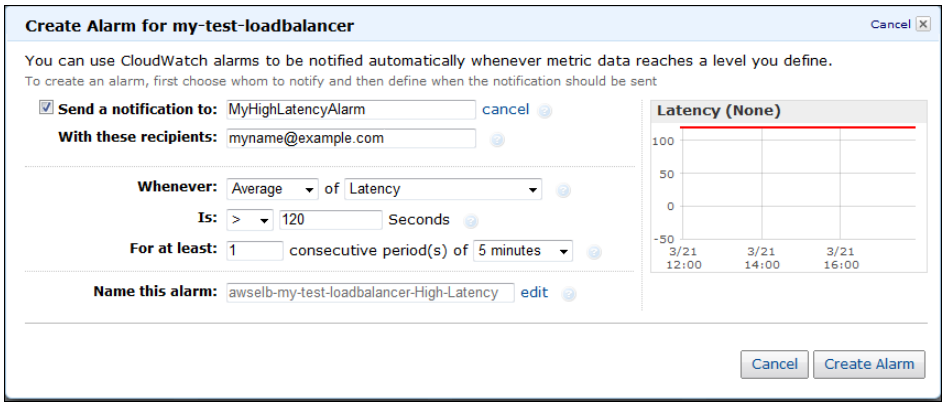
   Note
   If your email recepients have not yet been subscribed to the SNS topic, each email recipient will receive an email from Amazon SNS with a link to confirm their subscription to the SNS topic. If they do not confirm subscription, they will not receive future email Alarm notifications.
8. Configure the threshold for your alarm.

   a. In the **Whenever** boxes, select **Average** and **Latency**.
   b. In the **Is** boxes, define the threshold for the alarm by selecting **>** and entering `120`.
   c. In the **For at least** boxes, enter `1` and then select **5 minutes**, or you can define your own evaluation period.

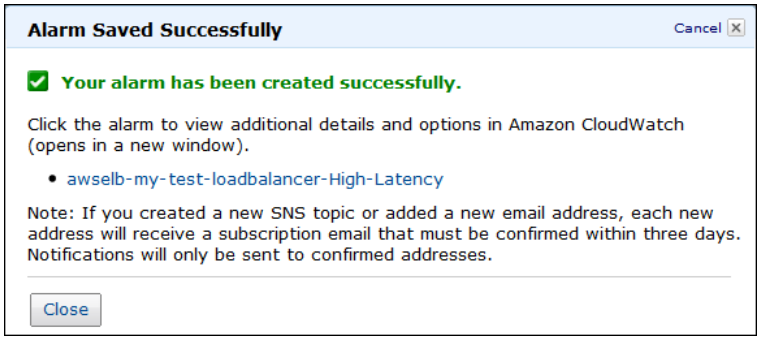
      Note
      A shorter period creates a more sensitive alarm. A longer period can mitigate brief spikes in a metric.

   d. In **Name this alarm**, a name is automatically generated for you. Click **Edit** if you want to change the name.

      Important
      You cannot modify the name after you create the alarm.



9. Click **Create Alarm**.



10. Click **Close**.

After you create the alarm, you can use the **Monitoring** tab to view a summary of alarms that have been set for that load balancer. From there, you can also edit the alarm.

Delete 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.

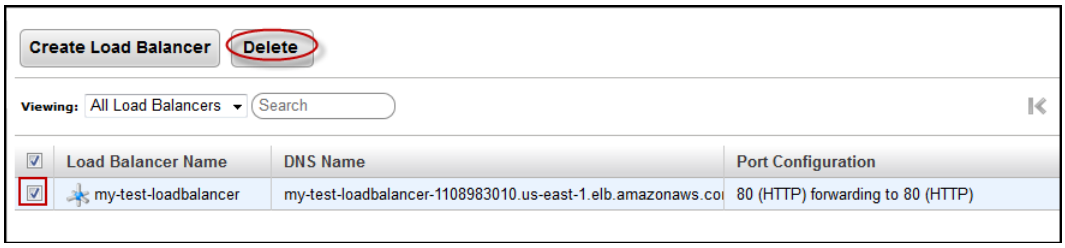
> **Note**
> Even after you delete the load balancer and the associated EC2 instances are deregistered, the EC2 instances continue to run. You will continue to incur charges on the Amazon EC2 instances while they are running. For information on stopping your EC2 instances, go to Stopping and Starting 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. 44).

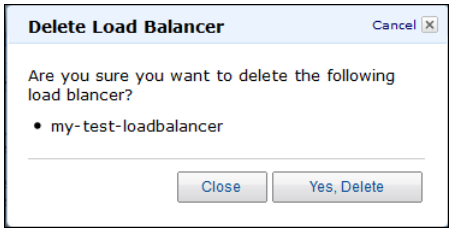
Using the AWS Management Console

To delete your load balancer

1. On the Amazon EC2 Getting Started page, in the **EC2 Dashboard** pane, under **NETWORK & SECURITY**, click **Load Balancers**.
2. On the **Load Balancers** page, select the check box next to the load balancer you want to delete, and then click **Delete**.



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



Using the Query API

To delete a load balancer

- Call `DeleteLoadBalancer` with *`LoadBalancerName`* = `my-test-loadbalancer`.

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  my-test-loadbalancer
```

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
```

Control 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. 169).

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, go to Actions in the *Elastic Load Balancing API Reference*.

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`–To check for date/time conditions.
- `aws:EpochTime`–To check for date/time conditions using a date in epoch or UNIX time.
- `aws:MultiFactorAuthAge`–To check how long ago (in seconds) the MFA-validated security credentials making the request were issued using Multi-Factor Authentication (MFA). Unlike other keys, if MFA is not used, this key is not present.
- `aws:principaltype`–To check the type of principal (user, account, federated user, etc.) for the current request.
- `aws:SecureTransport`–To check whether the request was sent using SSL. For services that use only SSL, such as Amazon RDS and Amazon Route 53, the `aws:SecureTransport` key has no meaning.
- `aws:SourceArn`–To check the source of the request, using the Amazon Resource Name (ARN) of the source. (This value is available for only some services. For more information, see Amazon Resource Name (ARN) under "Element Descriptions" in the *Amazon Simple Queue Service Developer Guide*.)
- `aws:SourceIp`–To check the IP address of the requester. Note that if you use `aws:SourceIp`, and the request comes from an Amazon EC2 instance, the public IP address of the instance is evaluated.
- `aws:UserAgent`–To check the client application that made the request.
- `aws:userid`–To check the user ID of the requester.
- `aws:username`–To check the user name of the requester, if available.

Note
Key names are case sensitive.

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":"*"
      }
   ]
}
```

Troubleshoot 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 Messages

Topic	Resource
HTTP error codes	What HTTP error codes can my load balancer return? (p. ?)
HTTP 400	HTTP 400: BAD_REQUEST (p. 172)
HTTP 405	HTTP 405: METHOD_NOT_ALLOWED (p. 172)
HTTP 408	HTTP 408: Request Timeout (p. 172)
HTTP 502	HTTP 502: Bad Gateway (p. 173)
HTTP 503	HTTP 503: Service Unavailable (p. 173)
HTTP response codes	Identifying the source of HTTP errors (p. 174)

Troubleshooting Elastic Load Balancing: Health Check

Topic	Resource
Registered instances failing health check	Troubleshooting Elastic Load Balancing: Health Check Configuration (p. 174)
Health check URL does not match the configuration	Health check URL and frequency different from configuration in API and console (p. 175)
Unhealthy instance sending 302 response code to load balancer	Instances treated as unhealthy when sending a 302 redirect response (p. 175)

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. 175)
Unable to register an instance launched from a paid AMI	Unable to register instance launched from a paid AMI. (p. 176)
Registered instance is out of service	OutofService: A Transient Error Occurred (p. 172)

Troubleshooting Elastic Load Balancing: Error Messages

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

CertificateNotFound: undefined

Cause: There is a delay in propogating the certificate to all the regions when it is created using the AWS Management Console. When the delay occurs, the error message is shown in the last step of the Create Load Balancer process.

Solution: Wait approximately 15 minutes and then try again. If the problem persists, contact the AWS Support Center.

OutofService: A Transient Error Occurred

Cause: This error message usually indicates some sort of internal problem within the Elastic Load Balancing service or the underlying network which is quite transient in nature. This temporary issue may also occur when Elastic Load Balancing queries the health of the load balancer and its associated back-end instances.

Solution: Retry the API call. If the problem persists, contact the AWS Support Center.

HTTP 400: BAD_REQUEST

Description: Indicates that the client sent a bad request.

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

Solution: Inspect the request headers being sent to the back-end EC2 instance for the malformed request.

HTTP 405: METHOD_NOT_ALLOWED

Description: Indicates that the method length is not valid.

Cause: The length of the method in the request header exceeds 127 characters.

Solution: Check the length of the method.

HTTP 408: Request Timeout

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

Cause: 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.

HTTP 502: Bad Gateway

Description: Indicates that the load balancer was unable to parse the response sent from a registered instance.

Cause: Malformed response from the instance or potentially an issue with the load balancer.

Solution: Verify that the response being sent from the instance conforms to HTTP specifications. Contact the AWS Support Center for further assistance.

HTTP 503: Service Unavailable

Description: Indicates that either the load balancer or the registered instances are causing 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 the AWS Support Center for further assistance.
- **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.
- **Cause 5**: Connection to the client is closed

  Solution: If your instances are in a Auto Scaling group, ensure that the desired capacity, if set, is equal to the number of Availability Zones in your Auto Scaling group. Or, reduce the number of Availability zones in your Auto Scaling group to match the minumum number of instances specified. By doing so, you make sure that none of your Availability zones is empty.

  If your instances are not part of an Auto Scaling group, your client must have closed the connection.

Troubleshooting Elastic Load Balancing: HTTP Error Codes

This section provides information about the HTTP error codes 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 400, 405, 408, 502 and 503 error responses in addition to any error responses returned by the registered instances.

Identifying the source of HTTP errors

Your load balancer emits metrics to Amazon CloudWatch for the HTTP response codes sent to clients, identifying the source of the errors as either the load balancer or the back-end instances. For information on using Amazon CloudWatch and defination of the available metrics, see Monitor Your Load Balancer Using Amazon CloudWatch (p. 159). The following metrics are available:

- **Response code**: HTTPCode_ELB_4XX
  Cause: Indicates a malformed or a cancelled request from the client.
  Solution:See HTTP 400: BAD_REQUEST (p. 172).
  Solution:See HTTP 405: METHOD_NOT_ALLOWED (p. 172).
  Solution: See HTTP 408: Request Timeout (p. 172).
- **Response code**: HTTPCode_ELB_5XX
  Cause: Either the load balancer or the registered instance is causing the error or the load balancer is unable to parse the response.
  Solution: See HTTP 502: Bad Gateway (p. 173).
  Solution: See HTTP 503: Service Unavailable (p. 173).
- **Response code**: HTTPCode_Backend_2XX
  Cause: Indicates a normal, successful response from the registered instance(s).
  Solution: None
- **Response code**: 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.
- **Response code**: 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.
- **Response code**: 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.

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. However, there are several reasons your health check can fail from the load balancer perspective. This section provides information about potential causes and steps you can take to narrow down and resolve the failed health check issues.

Registered instances failing 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:

- **Problem**: 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.

- **Problem**: 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.

- **Problem**: 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.

Health check URL and frequency different from configuration in API and console

Problem: 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.

Instances treated as unhealthy when sending a 302 redirect response

Problem: The load balancer treats a redirect response (302) as a failed health check.

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. 174) 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.

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

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

Solution: Try the steps recommended in the preceding section Registered instances failing load balancer health check (p. 174).

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*.

- API version: 2012-06-01
- Latest documentation update: June 3, 2013

Change	Description	Release Date
Support for DNS Failover	Added information about configuring Route 53 DNS failover for load balancers. For more information, see Configure DNS Failover for Your Load Balancer (p. 149).	3 June 2013
HTTP Methods	Added information about the HTTP methods supported by Elastic Load Balancing. For more information, see HTTP Methods (p. 8).	20 May 2013
Console support for viewing Amazon CloudWatch metrics and for creating alarms	Added information about viewing Amazon CloudWatch metrics and creating alarms for a specified load balancer using the Monitoring tab in the console. For more information, see Monitor Your Load Balancer Using Amazon CloudWatch (p. 159).	28 March 2013
Load Balancing in Default VPC	Added information about default VPC and load balancing Amazon EC2 instances launched within the default VPC. For information on default VPC, see Default VPC (p. 111). For information on creating a load balancer within a default VPC, see Create a Basic Load Balancer in Default VPC (p. 34).	11 March 2013

Change	Description	Release Date
Internal Load Balancing in Amazon VPC	With this release, a load balancer in Amazon Virtual Private Cloud (Amazon VPC) can be made either internal or Internet-facing. An internal load balancer has a publicly resolvable DNS name that resolves to private IP addresses. An Internet-facing load balancer has a publicly resolvable DNS name that resolves to public IP addresses. For more information, see Create a Basic Internal Load Balancer in Amazon VPC (p. 113).	10 June 2012
New features	Added information about console support for managing listeners, certificates, and cipher settings for existing load balancers. For information on managing listeners and cipher settings, see Add a Listener to Your Load Balancer (p. 128) and Delete a Listener from Your Load Balancer (p. 133). For information on managing certificates, see Update an SSL Certificate for a Load Balancer (p. 138).	18 May 2012
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. 14).	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 Configure Listeners for Your Load Balancer (p. 65).	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 Troubleshoot Elastic Load Balancing (p. 171).	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 Add a Listener to Your Load Balancer (p. 128) and Delete a Listener from Your Load Balancer (p. 133).	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 Deploy Elastic Load Balancing in Amazon VPC (p. 110).	21 November 2011
New content	Added new content for de-registering Amazon EC2 instances from your load balancer. For more information see, De-Register and Register Amazon EC2 Instances (p. 136).	21 November 2011
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

Change	Description	Release Date
New content	Added documentation for monitoring the load balancer using Amazon CloudWatch. For more information see Monitor Your Load Balancer Using Amazon CloudWatch (p. 159).	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 Creating a HTTPS/SSL Load Balancer (p. 73).	30 August 2011
Restructured content	Consolidated the instructions for setting up a load balancer with HTTP support and with HTTPS support into Create 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. 45).	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 Configure Custom Domain Names for Your Load Balancer (p. 143).	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 Manage Security Groups in Amazon EC2 (p. 99).	24 May 2011
New feature	Added documentation for the new Internet Protocol version 6 (IPv6) feature. For more information, see Use IPv6 with Elastic Load Balancing (p. 108).	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. 168).	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

W

Z