**9.1) ELB Basics**

ELB = Elastic Load Balancer

ELB...

...**evenly distributes** incoming **application traffic across multiple EC2 instances** in **multiple Availability Zones** (increases fault tolerance.)

...**detects** unhealthy instances and **routes traffic only to healthy instances**.

**Pricing/Cost Overview**

**Free Tier use is NOT available for ELB.**

*Charged:*

(1) Each hour or partial hour the load balancer is running

(2) For each GB of data transferred through the load balancer

**9.2) Creating an ELB**

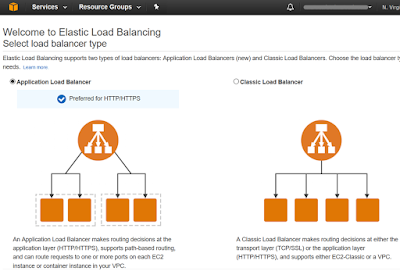
**Services** > **EC2** > Load Balancing: **Load Balancers** > **Create Load Balancer**

*Two types:*

(1) **Application Load Balancer**

(2) **Classic Load Balancer**

*Image: Services > EC2 > Load Balancing: Load Balancers > Create Load Balancer*

[](https://2.bp.blogspot.com/-x_iee1X_3Ec/WYeXZyzfCuI/AAAAAAAAHVM/zhaGmsT1YKoBNEC7cmRYJ5-WkZL2FMUhACLcBGAs/s1600/Create%2BLoad%2BBalancer.png)

*High-Level Steps Creating an ELB:*

1. Define Load Balancer

2. Assign Security Groups

3. Configure Security Settings

4. Configure Health Check

5. Add EC2 Instances

6. Add Tags

7. Review

**Elastic Load Balancers (ELB)**

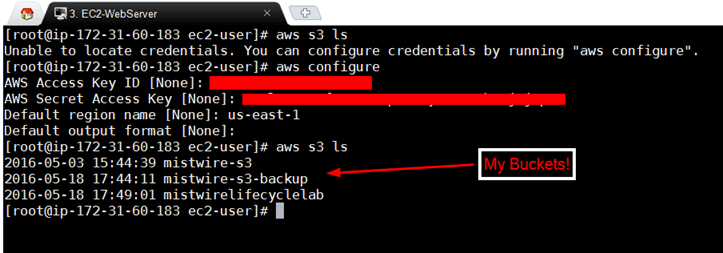
* ELB is never given a static IP address, just DNS name.
* ELBs can be “In Service” or “Out of Service”
* Thresholds
  + Unhealthy Threshold = how many intervals with no response before flagging as Out of Service
  + Healthy Threshold = how many intervals with response before flagging as In Service
* Support the following X-Forwarder headers:
  + X-Forwarded-For
  + X-Forwarded-Proto
  + X-Forwarded-Port

**CloudWatch – Performance Monitoring Service**

* Standard monitoring = 5 minutes
  + Turned on by default
* Detailed monitoring = 1 minute
* Monitors the hypervisor, NOT the guest OS
  + Does not monitor memory
* Dashboards – create/configure widgets to monitor your environment
* Alarms – notify when a given threshold is hit
* Events – automatically respond to state changes in your AWS resources
* Logs – aggregate, monitor & store logs. Agent installed onto EC2 instances

**AWS Command Line –**[**http://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-started.html**](http://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-started.html)

* You can now assign a role to an EC2 instance on the fly!
* AWS command line preinstalled on the AWS AMI
* Commands:
  + *Aws configure*
    - Input access key, Secret Access key, default region name (in doc above) & output format (I just hit enter)
  + *Aws s3 help*
    - Make Bucket = mb
    - Remove Bucket = rb



* If you use roles, you don’t have to store your credentials on your EC2 instance (which is a security risk)

## Monitoring

There are two types of EBS monitoring:

1. Basic Monitoring - available at no charge, and is enabled by default, with samples taken every 5 minutes
2. Detailed Monitoring - can be enabled for a fee, reducing the sample time to 1 minute.

System status checks make sure that packets can reach the instance (checking hypervisor is up)

Instance status checks make sure that the operating system can accept traffic

### Cloudwatch

Note that CloudWatch and CloudTrail are distinct products:

* CloudWatch - for performance monitoring and logging
* CloudTrail - for auditing i.e. when a new AWS role, user, etc is created. [Stores all of it’s data in S3.](https://aws.amazon.com/cloudtrail/faqs/) - when enabling CloudTrail, you need to provide a S3 bucket where all logs can be written to.

Default metrics:

* CPU
* Network
* Disk
* Status Checks

Everything else is a custom metric. i.e. **Memory is a custom CloudWatch metric.**

Types of monitoring:

* Standard - sample every 5 minutes
* Detailed - sample every minute

Cloudwatch can be use for:

* Dashboards
* Alarms
* Events
* Logs

Cloudwatch supports the following alarm states:

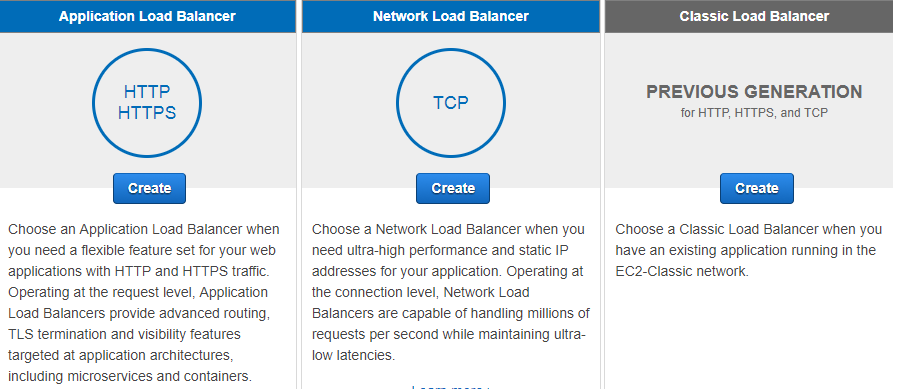
* **OK** - the metric is within the threshold
* **ALARM** - The metric is outside the threshold
* **INSUFFICIENT\_DATA** - The alarm has just started, but the metric is not available, or not enogh data is available for the metric to determine the alarm state

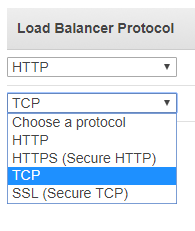
## ELBs (Elastic Load Balancer)

[ELB FAQ](https://aws.amazon.com/elasticloadbalancing/faqs/)

ELB supports Perfect Forward Secrecy.

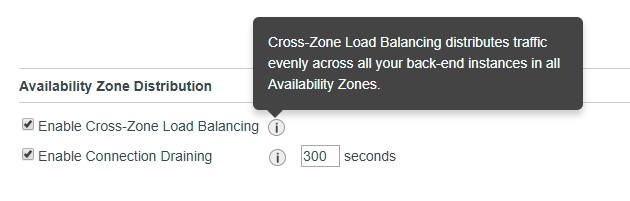
Types of ELB:



* **Classic Load Balancer** is the previous generation Load balancer for HTTP, HTTPS, and TCP traffic.
  + Can load balance HTTP/HTTPS applications
  + Can use layer 7-specific features such as X-Forwarded and sticky sessions
  + Can also use strict layer 4 load balancing for applications that rely purely on the TCP protocol.
  + When using the classic load balancer, you have the option of selecting which protocols it supports:
* **Application Load Balancer** is the current generation load balancer for HTTP, and HTTPS traffic.
  + Can be used instead of the Classic Load Balancer when using exclusively HTTP/HTTPS traffic. Do not use if the aplication depends on the TCP protocol.
  + Operates at the request level
  + Made available half way through 2016.
* **Network Load Balancer** is the current generation load balancer when using exclusively TCP traffic.
  + Can be used instead of the Classic Load Balancer when using TCP traffic.
  + Operates at the connection level
  + The Network Load Balancer is suitable when you need ultra-high performance, and have static IP addresses for your application.

### Cross-zone load balancing

By default, the Classic Load Balancer distributes traffic across all EC2 instances regardless of AZ. If you want to balance evenly across AZ, make sure you enable Cross-Zone load balancing:



[More info on Cross-Zone load balancing](http://docs.aws.amazon.com/elasticloadbalancing/latest/classic/enable-disable-crosszone-lb.html)

### Make sure you remember to shut down your ELBs if you’re not using them

The major reason that people exceed the free tier is because they forgot to terminate their ELBs. It’s a good idea to tag your ELBs so that they can be tracked via resource groups.

ELBs initially have a DNS name, but no public IP address.

When an ELB is monitoring instances, the instance can have one of two status:

* In service
* Out of service

Set Evalate Target Health to true, and enable Latency Based Routing for HA (High Availability)

## Auto scaling

**Auto scaling Groups**

* Have to have a launch configuration to have an auto scaling group
* Can create rules to spin-up and/or shut down instances based on monitor triggers
* Deleting an auto scaling group will automatically delete any instances it created

Scaling Policy is a set of rules i.e. Increase if average CPU > 80% for a consecutive peroid of 5 minutes.

**Desired Instances** is adjusted based on the scaling policy, and won’t go below the miniumum or above the maximum size of the group.

Note that **there is a default maximum of 20 running on-demand EC2 instances** regardless of the min/max you set in your ASG scaling policies. You can request a limit increase by getting in touch with AWS… if your auto scaling triggers are firing, but you are not getting any more instances, check that you haven’t reached the default maximum.

Deleting an ASG will automatically delete any instances that it created.

**Launch configurations cannot be modified after creation**. If you need to make a change, create a new launch configation and update your auto scaling group to use it.

Launch configurations can belong to multiple Auto Scaling groups, however you can only specify one launch configuration at a time for an Auto Scaling group.

AMIs can be used with Auto Scaling groups.

[More info on launch configurations](http://docs.aws.amazon.com/autoscaling/latest/userguide/LaunchConfiguration.html)

The following scale out options are available:

* Scheduled scaling - adjusting the size of a group at a specific time
* Dynamic scaling - via creating a scaling policy to automatically adjust the size of the group based on a specificed increase in demanc
* Manual scaling - via manually increasing the size of the group

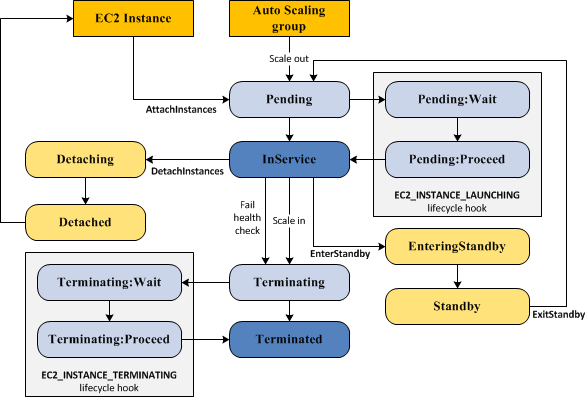
[More info on auto scaling lifecycles](http://docs.aws.amazon.com/autoscaling/latest/userguide/AutoScalingGroupLifecycle.html)

To attach EC2 instances to an Auto Scaling group, ensure that:

* The instance is in the running state
* The AMIs used to launch the instance still exist
* The instance is not a member of another Auto Scaling group
* The instance is in the same AZ as the Auto Scaling group

[More info on attaching instances to an Auto Scaling group](http://docs.aws.amazon.com/autoscaling/latest/userguide/attach-instance-asg.html)

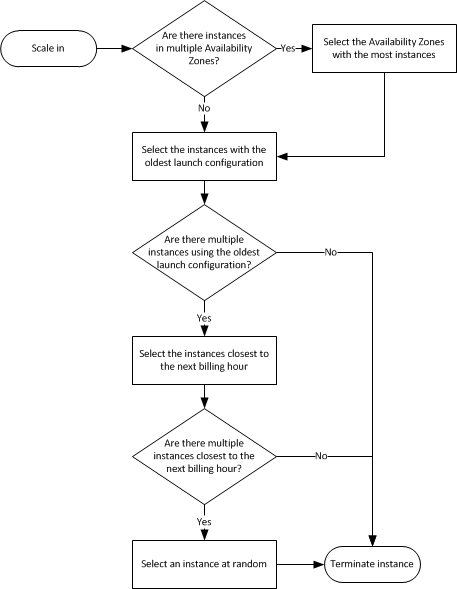
### Auto Scaling Lifecycle and Lifecycle Hooks



In the **Pending:Wait** state, no lifecycle policies take effect.

The cooldown period is the number of seconds after a scaling activity completes before another can start.

### Auto Scaling Group Termination Policy



**ELASTIC LOAD BALANCING (ELB)**

* ELB is a highly available service that distributes traffic across EC2 instances.

**AMAZON CLOUD WATCH (CW)**

* Cw monitors cloud resources and applications

**AUTO SCALING (AS)**

* AS maintains availability of applications by scaling EC2 capacity up or down

**ELB**

ELB distributes traffic across EC2 in on or more AZs

ELB supports routing and load balancing of

1. HTTP
2. HTTPS
3. TCP
4. SSL

ELB support integrated Certificate Management and SSL termination

Types

* Internet-Facing Load Balancer
  + request form Internet to EC2s
* Internal load balancer
  + load balancer between tiers of the applications after the internet facing load balancing

HTTPS Load Balancer

* SSL offload
* ELB does not support SNI
* Certs need Subject Alternative Name (SAN)

Listener

Process that checks for connection requests

supports

* HTTP
* HTTPS
* TCP
* SSL

**CONFIGURING ELB ( CONSOLE OR CLI )**

Idle connection  timeout

* triggered when no data is sent over the connection for a specified time periods
* HTTP and HTTPS enable keep-alive
* allows lb to reuse connections

Cross-zone load Balancing

* evenly route traffic regardless of AZ

Connection Draining

* Keep connections but stop sending new traffic

Proxy Protocol

* TCP or SSL forwards requests without modifying request

Sticky Sessions

* enables LB to bind a users’s session to a specific instance

health Checks

* test status of EC2 instances
* ping, connection attempt, or a page that is checked

**CLOUD WATCH**

Monitor AWS resource sin real time

support multiple actions – SNS or auto scaling

* basic 5 minutes
* detailed every minute
* CloudWatch uses GET request
* CloudWatch agent can send log data to CW logs running Linux or Ubuntu
* 5000 alarms per account
* retained for 2 weeks
  + need longer move to S3 and Glacier

**AUTO SCALING**

Scale EC2 capacity automatically by scaling out and scaling in.

Maintain a minimum or specified number of instances

* manual Scaling
  + max, min or desired capacity  of AS Group
  + for infrequent events
* scheduled scaling
  + scaling actions are performed automatically  as a function of a time and date
* Dynamic Scaling
  + define parameters controlled by AS Group
    - network bandwidth

**AUTO SCALING COMPONENTS**

Launch configuration – template to crate new instance

* Name
* AMI
* Instance Type
* Security Group
* key pairs

**AUTO SCALING GROUP**

As group is a collection of EC2 instances managed by the auto scaling service

AS Group must contain

* Name
* Max and min of instances

Can use on demand or spot instances

Ondemand by default

**SCALING POLICY**

increase or decrease by a specific number of instances

can associate more than one scaling policy with auto scalling group

* Scale out Quickly
* Scale in Slowly

cooldown period – configurable, when  to suspend scaling activities for a short time

Elastic Load Balancing offers two types of load balancers that both feature high availability, automatic scaling, and robust security. These include the Classic Load Balancer that routes traffic based on either application or network level information, and the Application Load Balancer that routes traffic based on advanced application level information that includes the content of the request.

* When configuring ELB health checks, bear in mind that you may want to create a file like healthcheck.html or point the ping path of the health check to the main index file in your application
* Remember the health check interval is how often a health check will occur
* Your Healthy/Unhealthy thresholds are how many times either will check before marking the origin either healthy or unhealthy
  + Health Check Interval: 10 seconds
  + Unhealthy Threshold: 2
  + Healthy Threshold: 3
  + This means that if the health check interval occurs twice without success, then the source will be marked as unhealthy. This is 2 checks @ 10 seconds per check, so basically after 20 seconds the origin will be marked unhealthy
  + Likewise, if the healthy threshold is marked at 3, then it would be 3 x health check interval or 10 seconds being 30 seconds. After 30 seconds with 3 consecutive success checks, the origin will be marked as healthy.
* Enable Cross-Zone Load Balancing will distribute load across all back-end instances, even if they exist in different AZ's
* ELBs are NEVER given public IP Addresses, only a public DNS name
* ELBs can be In Service or Out of Service depending on health check results
* Charged by the hour and on a per GB basis of usage
* Must be configured with at least one listener
* A listener must be configured with a protocol and a port for front end (client to ELB connection), as well as a protocol and port for backed end (ELB to instances connection)
* ELBs support HTTP, HTTPS, TCP, and SSL (Secure TCP)
* ELBs support all ports (1-65535)
* ELBs do not support multiple SSL certificates
* Classic ELBs support the following ports:
  + 25 (SMTP)
  + 80 (HTTP)
  + 443 (HTTPS)
  + 465 (SMTPS)
  + 587 (SMTPS)
  + 1024-65535
* HTTP Error Codes:
  + 200 - The request has succeeded
  + 3xx - Redirection
  + 4xx - Client Error (404 not found)
  + 5xx - Server Error

## Configure Elastic Load Balancing With SSL And AWS Certificate Manager For Bitnami Applications On AWS

### Introduction

Load balancing is a technique commonly used by high-traffic Web sites and Web applications to share traffic across multiple hosts, thereby ensuring quick response times and rapid adaptation to traffic peaks and troughs. The [Elastic Load Balancing](https://aws.amazon.com/elasticloadbalancing/) service from [Amazon Web Services (AWS)](https://aws.amazon.com/) with Secure Sockets Layer (SSL) support makes it easy to add secure load balancing for Bitnami applications running on AWS.

This guide walks you through the process of configuring and testing an Elastic Load Balancer with an SSL certificate for a Bitnami application running on AWS.

### Assumptions And Prerequisites

This guide assumes that:

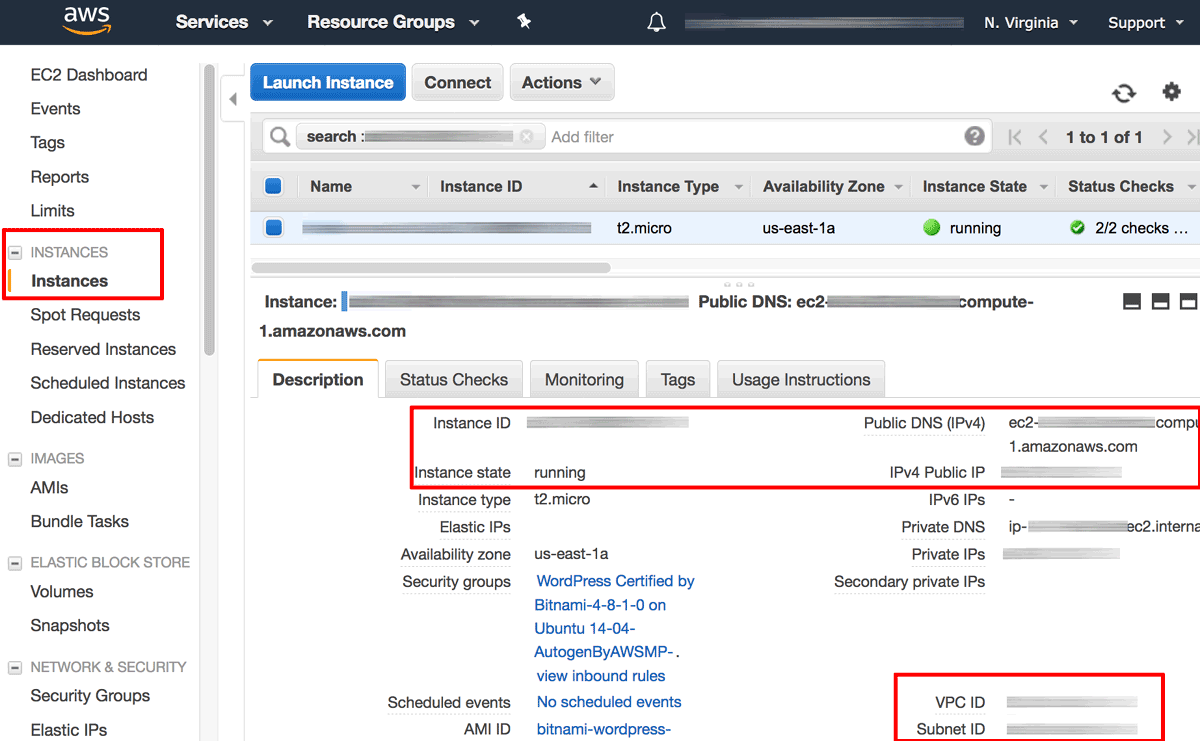
* You have a Bitnami application running on AWS, deployed using the [Bitnami Launchpad for AWS](https://docs.bitnami.com/aws/get-started-launchpad), the [AWS Marketplace](https://docs.bitnami.com/aws/get-started-marketplace) or the [AWS Console](https://docs.bitnami.com/aws/get-started-console).
* You have the necessary credentials to log in to the Bitnami application instance.
* You own a domain name.

|  |
| --- |
| NOTE: This guide only covers the process of configuring a load balancer with an SSL certificate for a single Bitnami application. If you wish to balance load across multiple applications, additional steps will be required. |

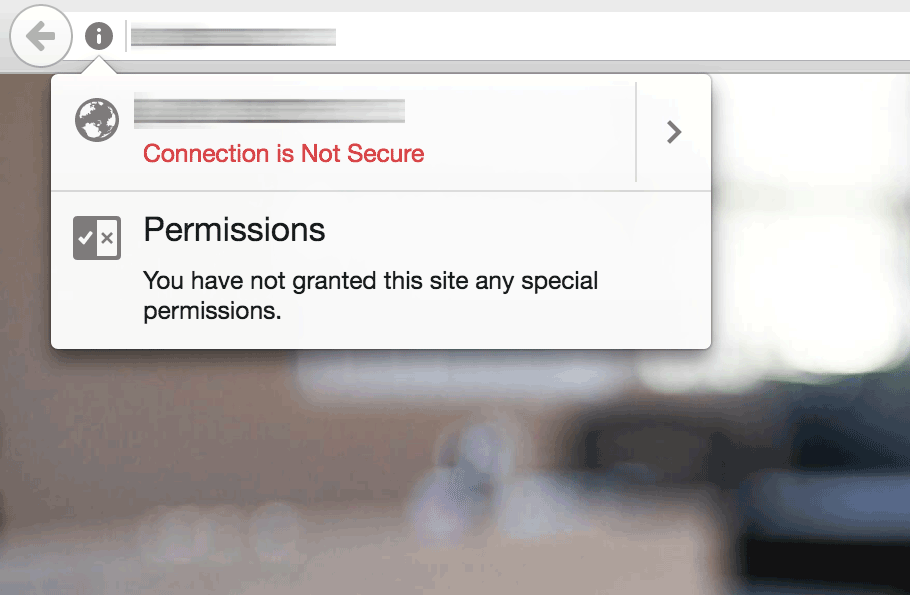
### Step 1: Identify Your Bitnami Application Instance In The AWS EC2 Console

The first step is to identify your Bitnami application instance and collect various important bits of information, such as the instance ID, DNS name, public IP address, availability zone and Virtual Private Cloud (VPC) ID. To do this:

* Log in to the [AWS EC2 console](https://console.aws.amazon.com/ec2/).
* From the "Services" menu, select the "EC2" service.
* From the EC2 Dashboard, select the "Instances -> Instances" menu item.
* Search for and select your Bitnami application instance from the list of available instances.
* From the instance details pane, note the instance ID, DNS name, public IP address, availability zone and VPC ID. You will need these details in subsequent steps.

[](https://docs.bitnami.com/images/img/how_to_guides/configure-elb-ssl-aws/elb-aws-1-0ad15f0d.png)

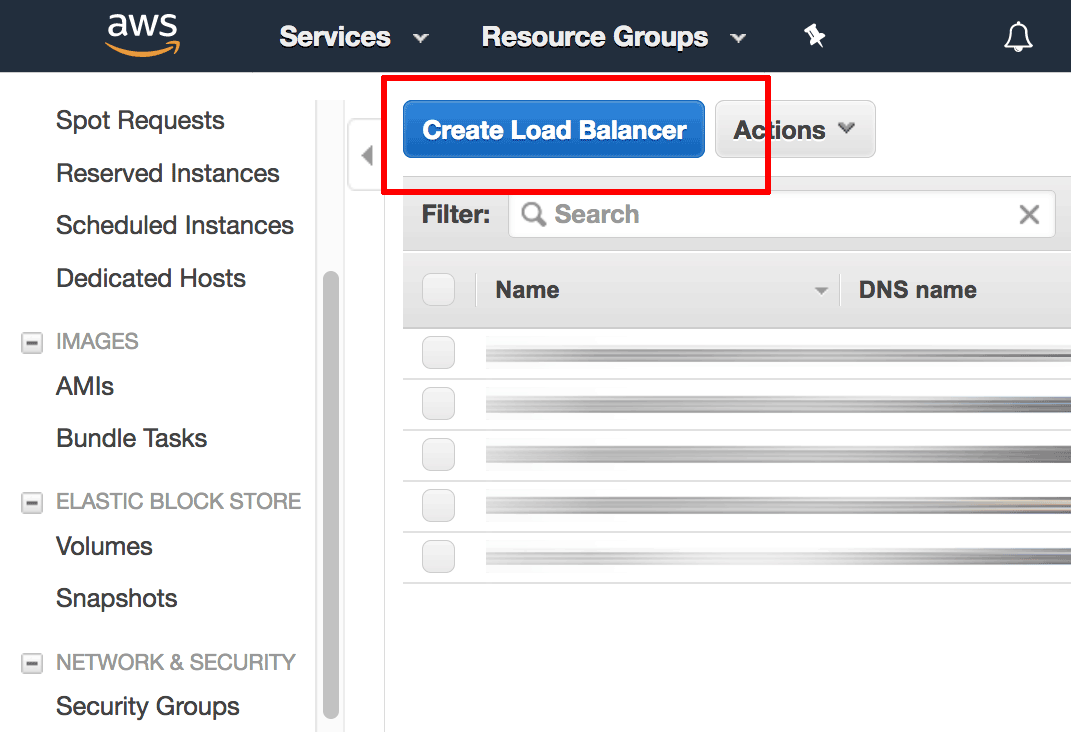
* Browse to the instance's public IP address and confirm that you get a positive response, such as the welcome page of the Bitnami application. This response will be necessary to pass health checks performed by the load balancer.

[](https://docs.bitnami.com/images/img/how_to_guides/configure-elb-ssl-aws/elb-aws-2-07e45d84.png)

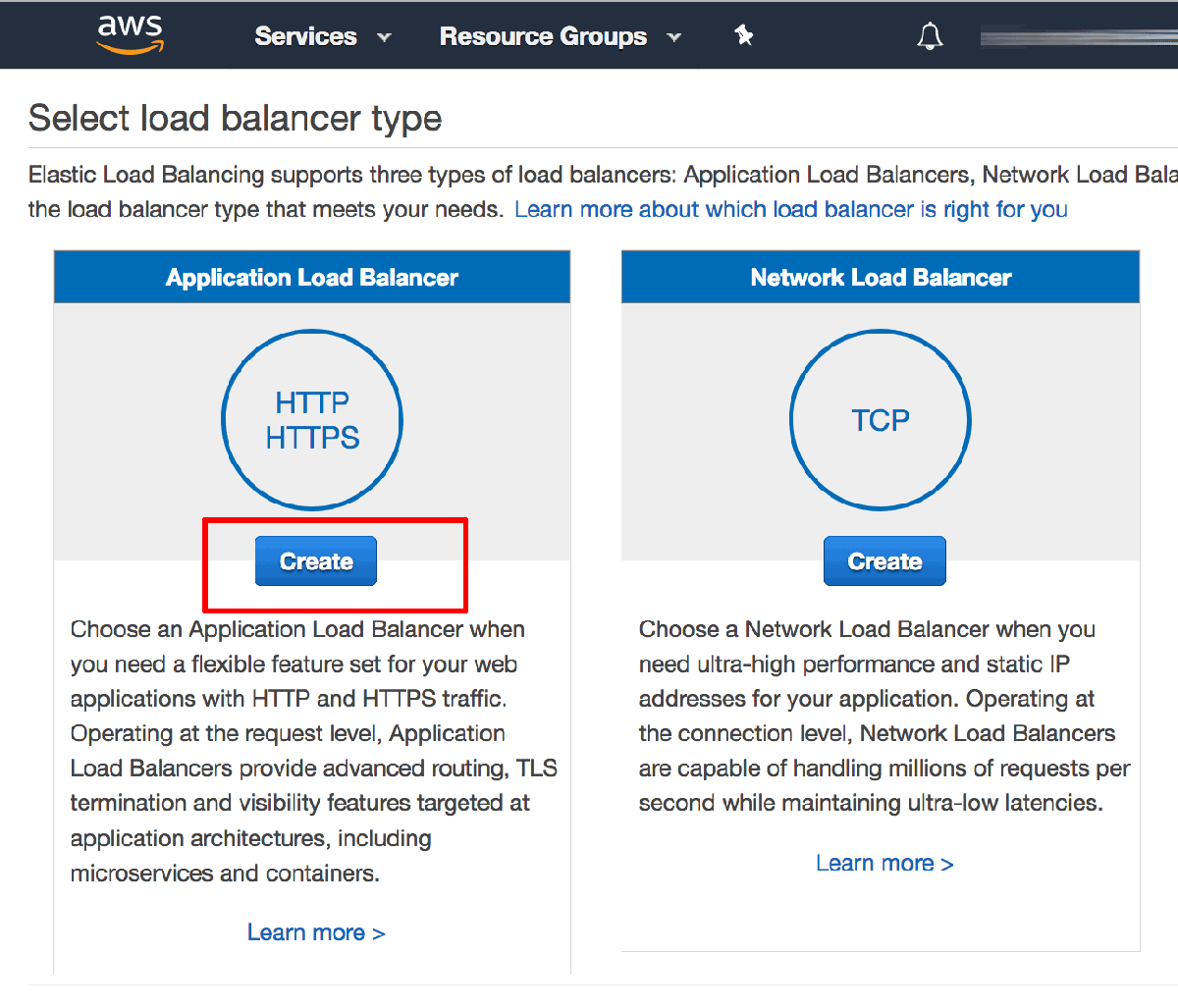
### Step 2: Create An Elastic Load Balancer With HTTP And HTTPS Support

Next, create an Elastic Load Balancer as follows:

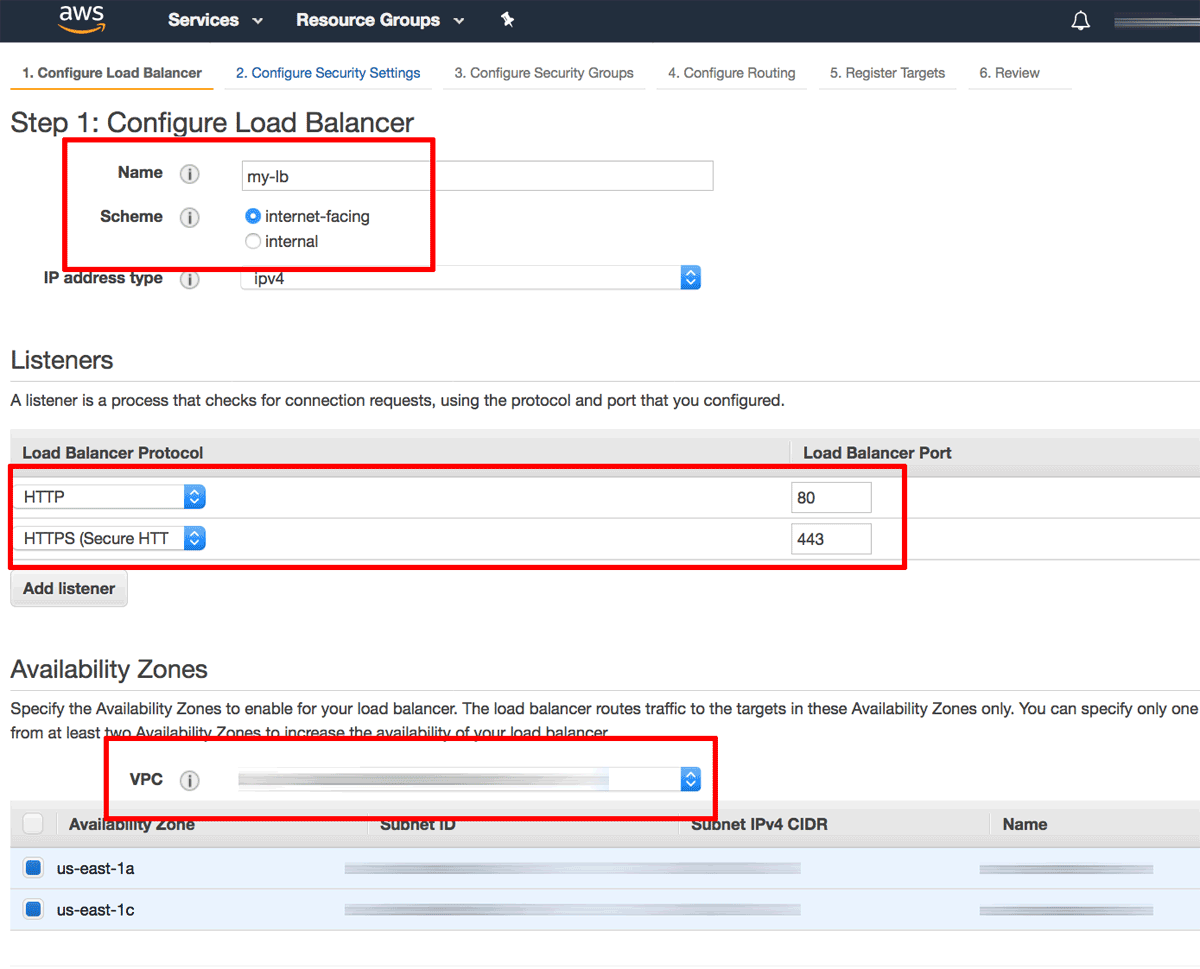
* From the EC2 Dashboard, select the "Load Balancing -> Load Balancers" menu item.
* Click the "Create Load Balancer" button.

[](https://docs.bitnami.com/images/img/how_to_guides/configure-elb-ssl-aws/elb-aws-3-2b237af3.png)

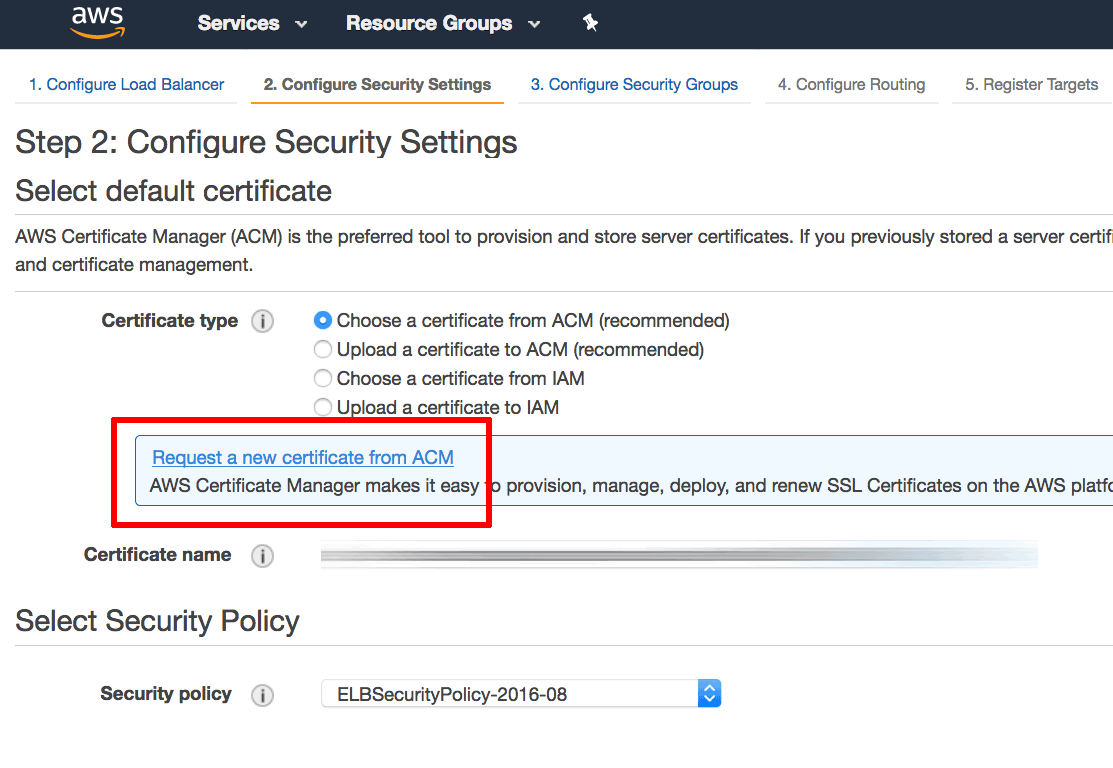
* On the "Select Load Balancer type" page, select the "Application Load Balancer" option and click "Create".

[](https://docs.bitnami.com/images/img/how_to_guides/configure-elb-ssl-aws/elb-aws-4-8d933d6c.png)

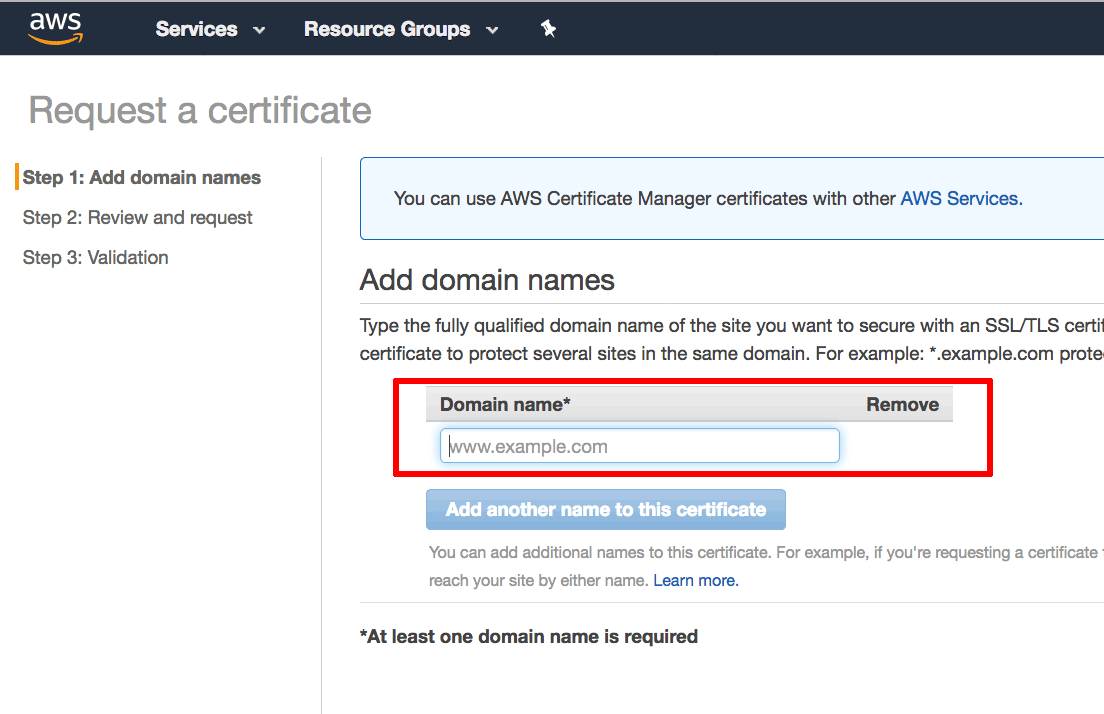
* On the subsequent "Configure Load Balancer" page:
  + Enter a name for the load balancer and specify the scheme as "Internet facing".
  + In the "Listeners" section, ensure that there is an HTTP listener on load balancer port 80. Click the "Add listener" button and add a second HTTPS listener on load balancer port 443. This configures the load balancer to handle both HTTP and HTTPS requests from clients.
  + In the "Availability Zones" section, select the same VPC as the one used by your Bitnami application instance and select a subnet from each availability zone.
  + Click the "Next: Configure Security Settings" button to proceed.

[](https://docs.bitnami.com/images/img/how_to_guides/configure-elb-ssl-aws/elb-aws-5-d477fd7a.png)

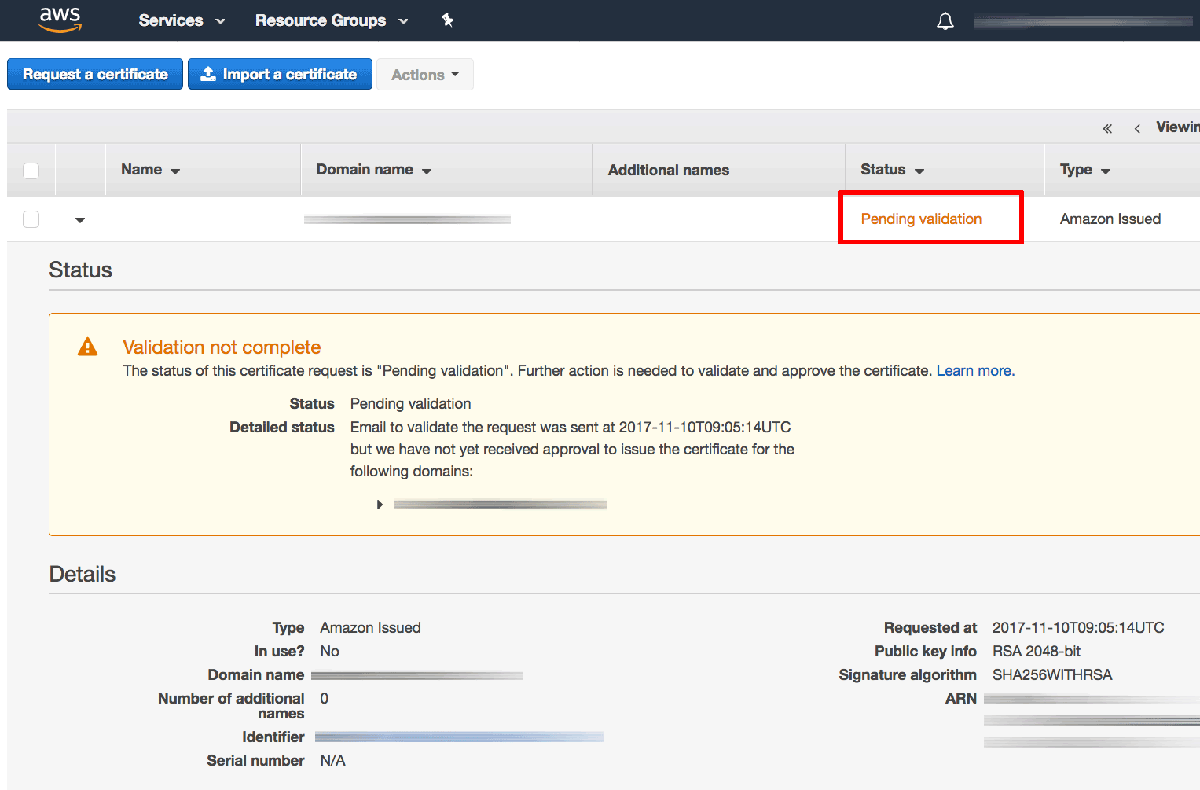
* On the "Configure Security Settings" page, select the "Request a new certificate from ACM" option to create a new SSL certificate for your load balancer. This will launch the [AWS Certificate Manager (ACM)](https://aws.amazon.com/certificate-manager/) in a new window.

[](https://docs.bitnami.com/images/img/how_to_guides/configure-elb-ssl-aws/elb-aws-6-40e8b643.png)

* Within the AWS Certificate Manager, on the "Request a certificate" page, enter your domain name. Click "Review and request" to review and confirm the request.

[](https://docs.bitnami.com/images/img/how_to_guides/configure-elb-ssl-aws/elb-aws-7-ca1bb875.png)

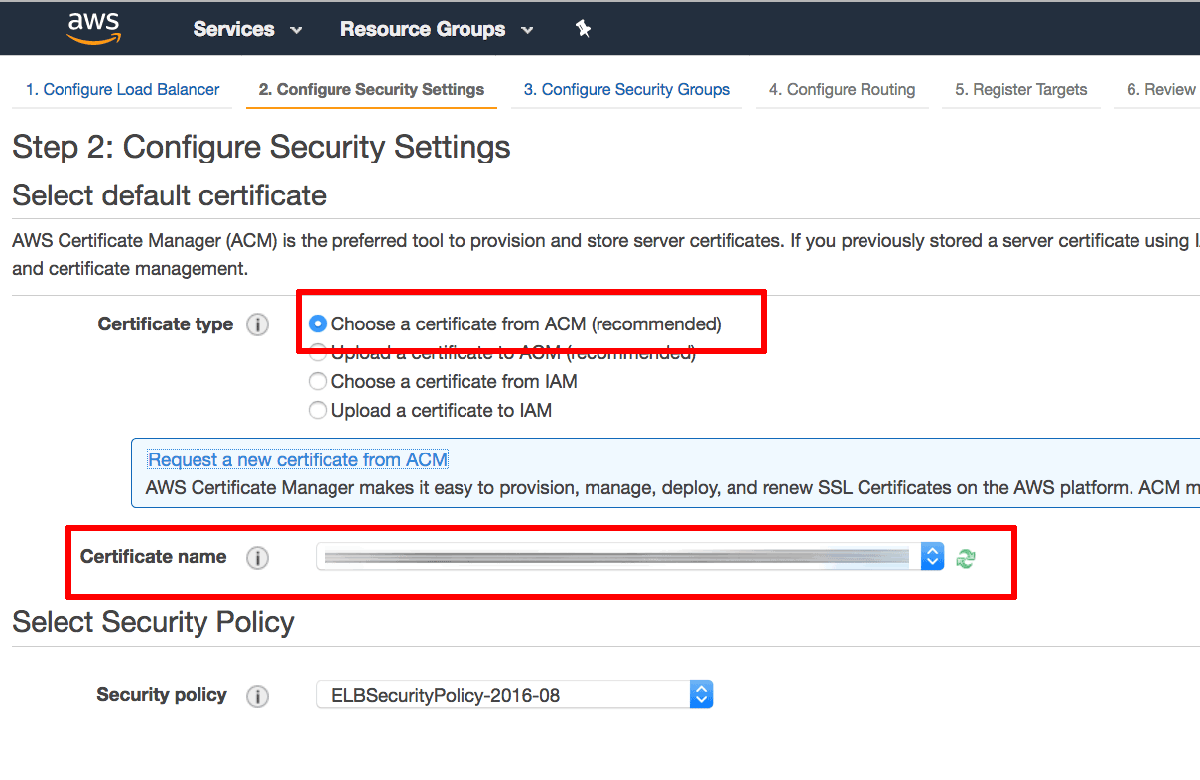
An email will now be sent to the registered owner of the domain with instructions to confirm the certificate request by validating the domain. The domain will appear in the AWS Certificate Manager with status set to "Pending validation".

[](https://docs.bitnami.com/images/img/how_to_guides/configure-elb-ssl-aws/elb-aws-8-2b730cda.png)

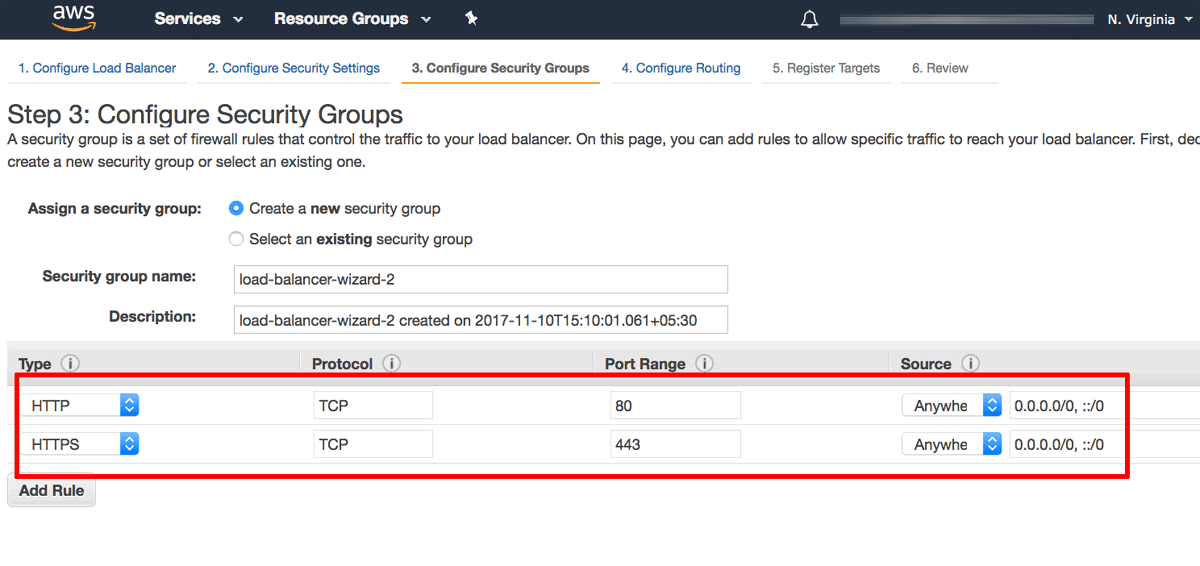
|  |
| --- |
| TIP: [Learn more about the AWS Certificate Manager domain validation process](https://aws.amazon.com/certificate-manager/faqs/). |

Once the domain has been validated, the certificate will be issued and will appear in the AWS Certificate Manager with status set to "Issued".

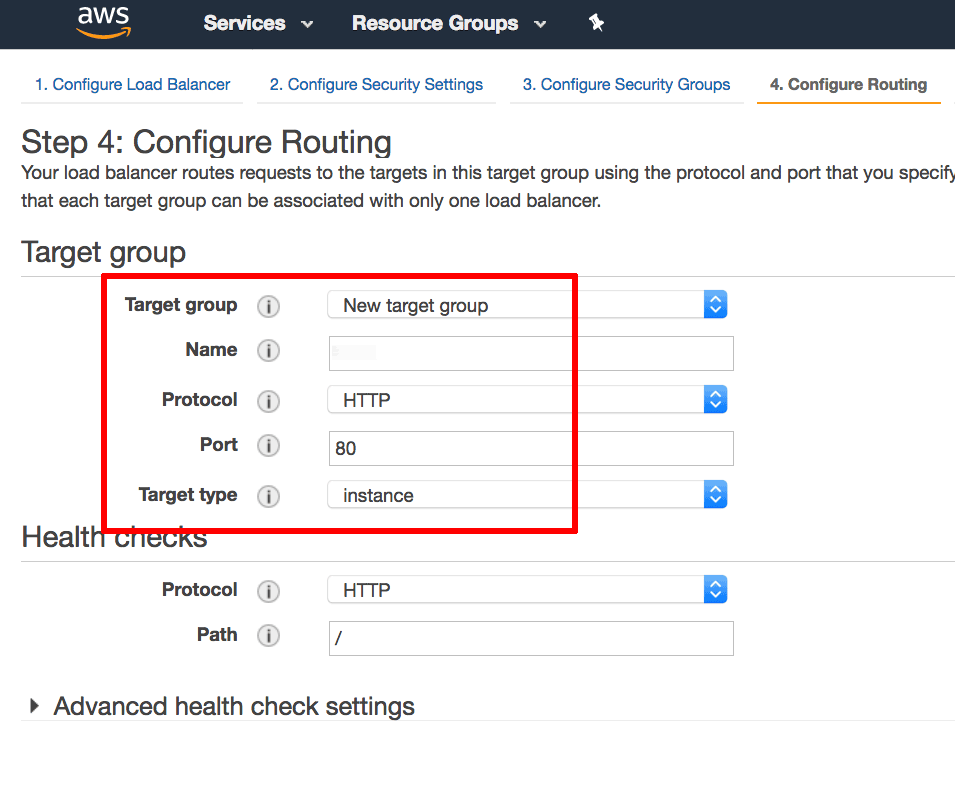
* Back on the "Configure Security Settings" page for the load balancer, the newly-issued certificate should now appear in the list of available certificates. Select it and click the "Next: Configure Security Group" button to proceed.

[](https://docs.bitnami.com/images/img/how_to_guides/configure-elb-ssl-aws/elb-aws-9-5aaa45d6.png)

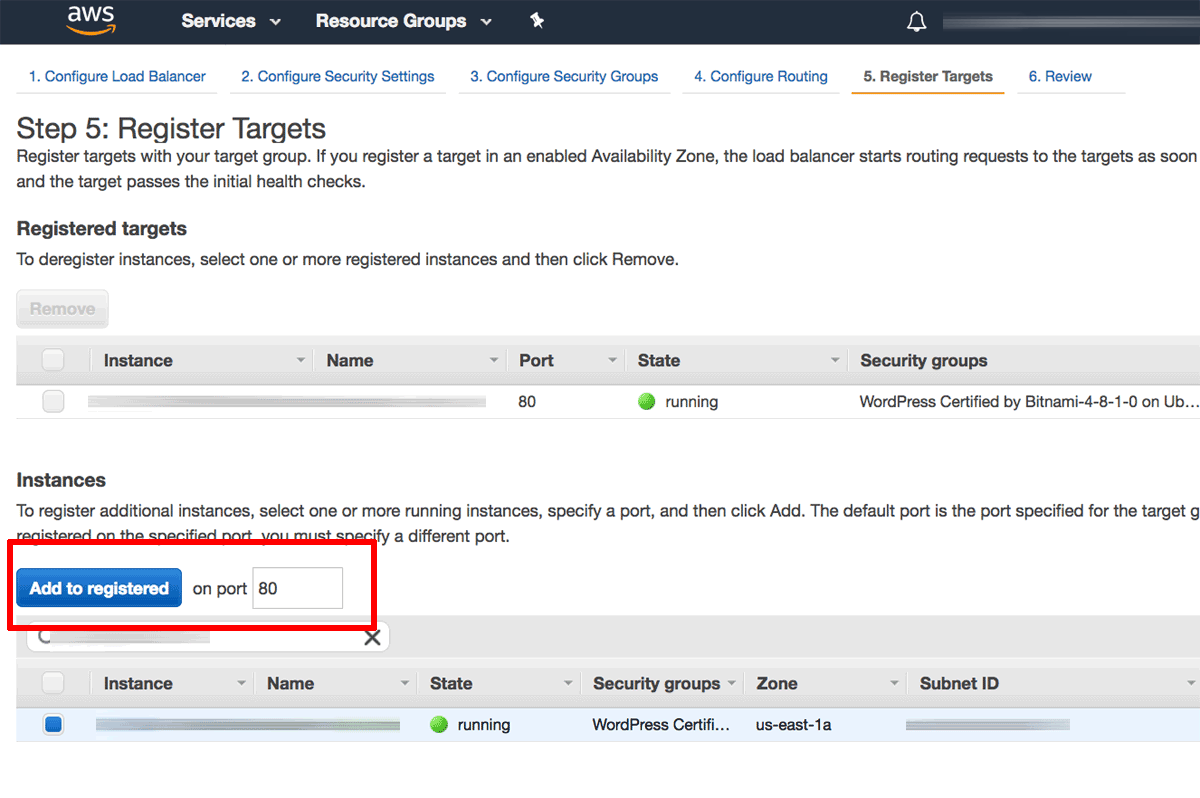
* On the "Configure Security Group" page:
  + Select the option to "Create a new security group".
  + Add a security rule to allow inbound traffic on port 80 (the HTTP port) with source "Anywhere".
  + Add a second security rule to allow inbound traffic on port 443 (the HTTPS port) with source "Anywhere".
  + Click the "Next: Configure Routing" button to proceed.

[](https://docs.bitnami.com/images/img/how_to_guides/configure-elb-ssl-aws/elb-aws-10-645e43e1.png)

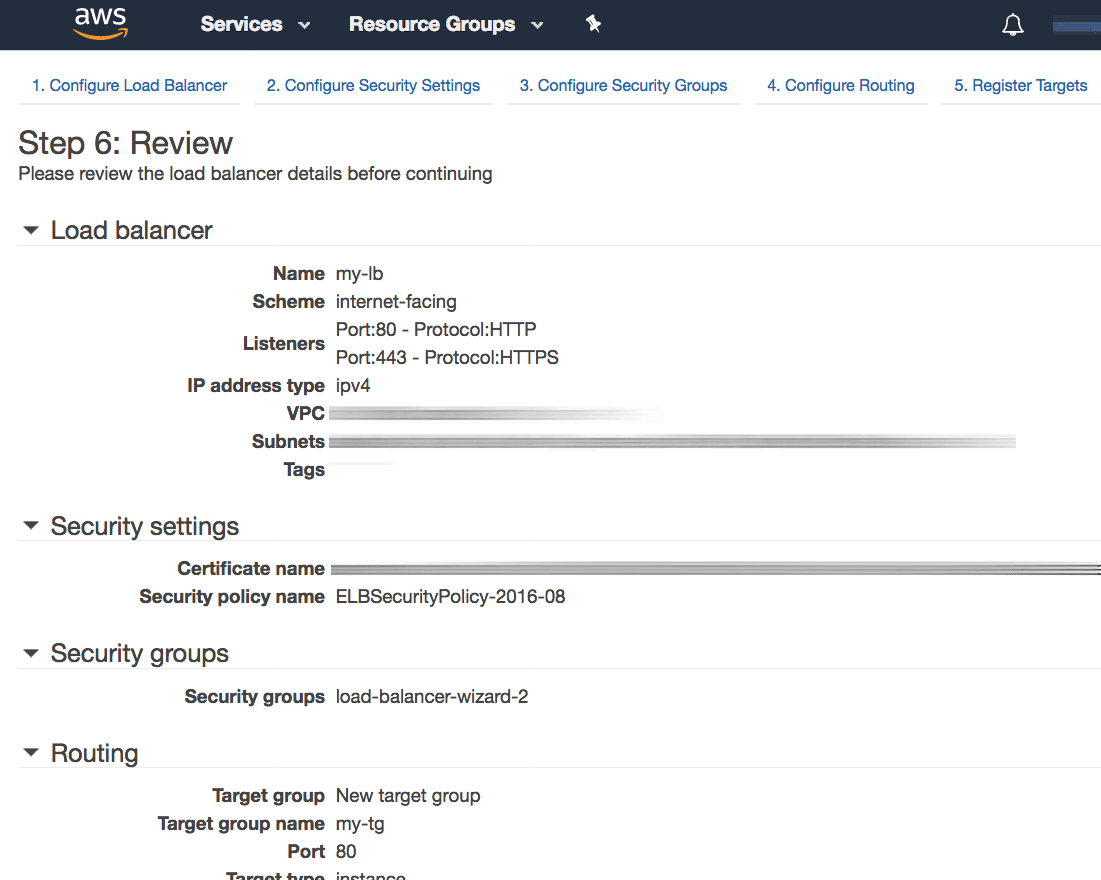
* On the "Configure Routing" page:
  + In the "Target group" section, create a new target group and assign it a name. Ensure that the protocol is set to "HTTP", the port to "80" and the target type to "instance". With this configuration, traffic between the load balancer and the instance will be transmitted using HTTP, even for HTTPS requests made by the client to the load balancer.
  + In the "Health checks" section, define the protocol as "HTTP" and the path to "/".
  + Click the "Next: Register Targets" button to proceed.

[](https://docs.bitnami.com/images/img/how_to_guides/configure-elb-ssl-aws/elb-aws-11-5c71551b.png)

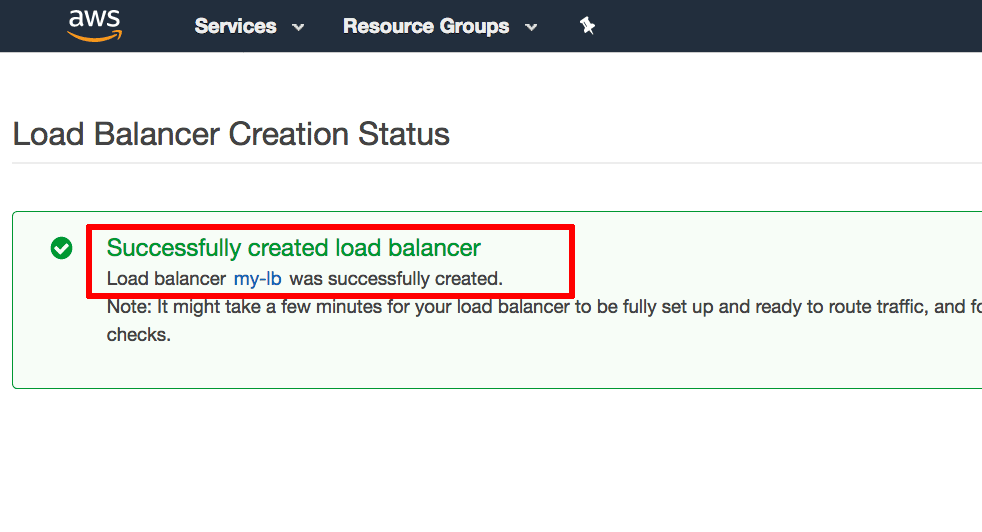
* On the "Register Targets" page, use the instance ID obtained in Step 1 to identify and select the Bitnami application instance. Click the "Add to registered" button to move the instance into the list of registered targets. Click the "Next: Review" button to proceed.

[](https://docs.bitnami.com/images/img/how_to_guides/configure-elb-ssl-aws/elb-aws-12-72463f8e.png)

* On the "Review" page, review the details of the configured load balancer. Click "Create" to confirm the configuration and deploy the load balancer.

[](https://docs.bitnami.com/images/img/how_to_guides/configure-elb-ssl-aws/elb-aws-13-42a871e3.png)

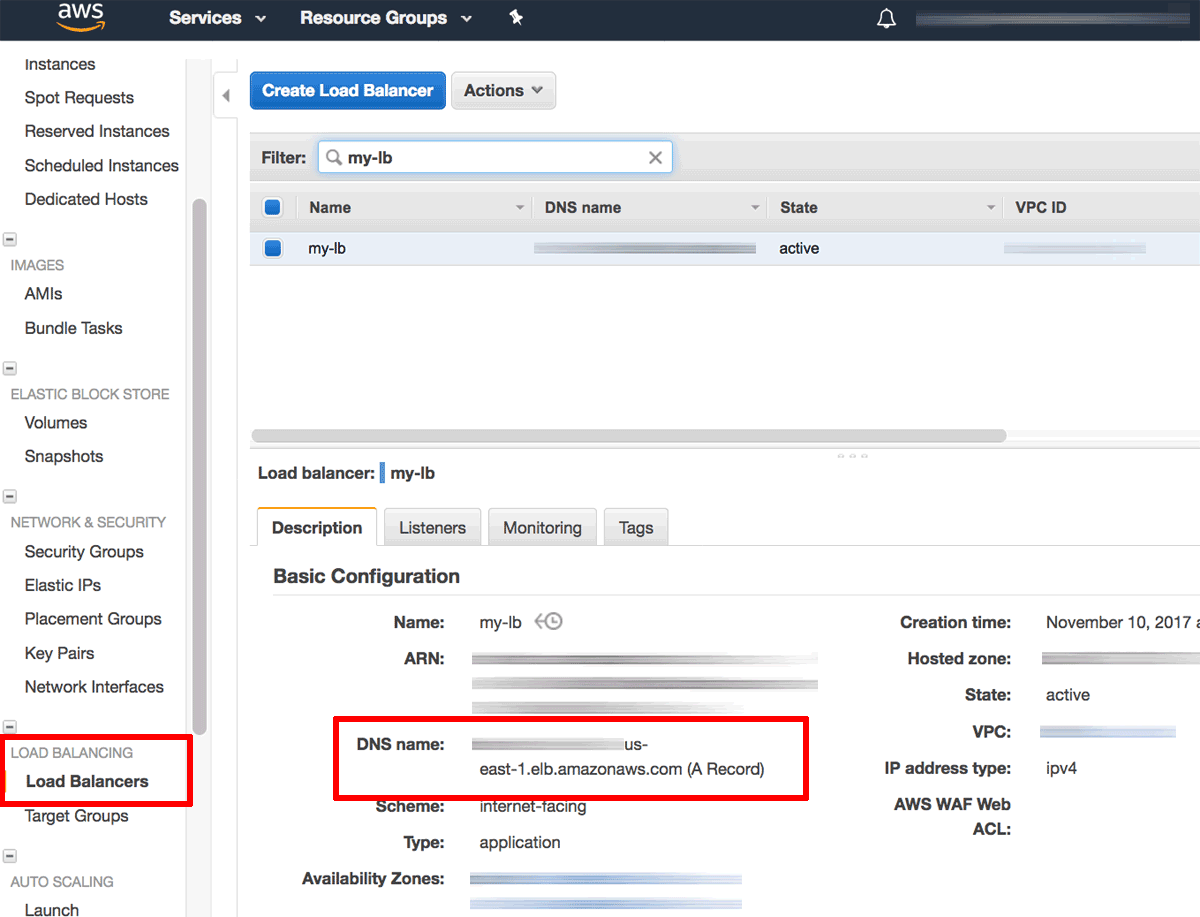
The load balancer will now be created and deployed.

[](https://docs.bitnami.com/images/img/how_to_guides/configure-elb-ssl-aws/elb-aws-14-cfc7f2df.png)

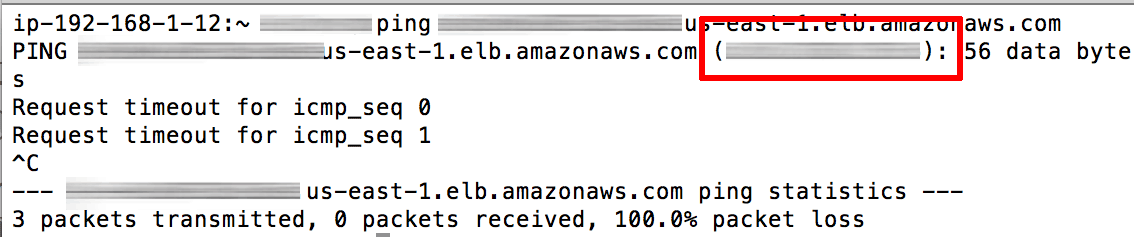
### Step 3: Point Your Domain Name To The Load Balancer's IP Address

Once the deployment is complete, perform the following steps:

* Visit the EC2 Dashboard and note the DNS name for the configured load balancer.

[](https://docs.bitnami.com/images/img/how_to_guides/configure-elb-ssl-aws/elb-aws-15-4e7d9ef8.png)

* Using the DNS name, obtain the corresponding public IP address, using a tool like ping or nslookup or a service like [DNS Lookup](https://www.whatismyip.com/dns-lookup/).

[](https://docs.bitnami.com/images/img/how_to_guides/configure-elb-ssl-aws/elb-aws-16-5bb59413.png)

* Update your domain's DNS settings by adding an A record that points to the public IP address of the load balancer. To do this, you will usually need to log in to your domain name provider's management console and make the necessary changes.

|  |
| --- |
| NOTE: Once you make the necessary changes, it can take up to 48 hours for the change to propagate across other DNS servers. |

### Step 4: Modify The Web Server Configuration On The Bitnami Application Instance

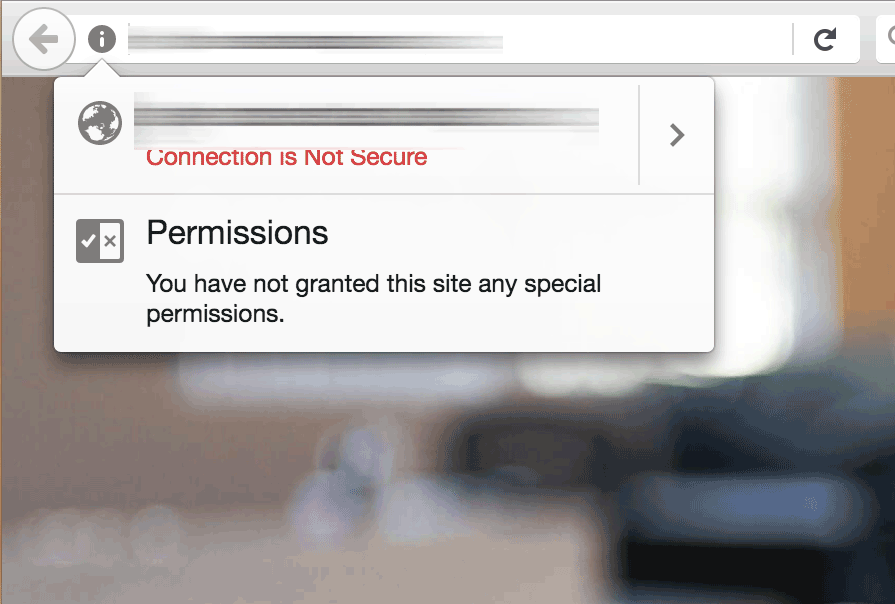
The final step is to update the Apache server configuration on the Bitnami application instance to correctly handle SSL requests from the load balancer. To do this:

* [Log in to the application instance using SSH](https://docs.bitnami.com/aws/faq#how-to-connect-to-the-server-through-ssh).
* Modify the /opt/bitnami/apps/APPNAME/conf/httpd-prefix.conf and add the following line at the top of the file. This line ensures correct operation of the HTTP connection between the load balancer and the instance, even for load-balanced HTTPS requests.
* SetEnvIf x-forwarded-proto https HTTPS=on
* Restart the server:
* sudo /opt/bitnami/ctlscript.sh restart apache

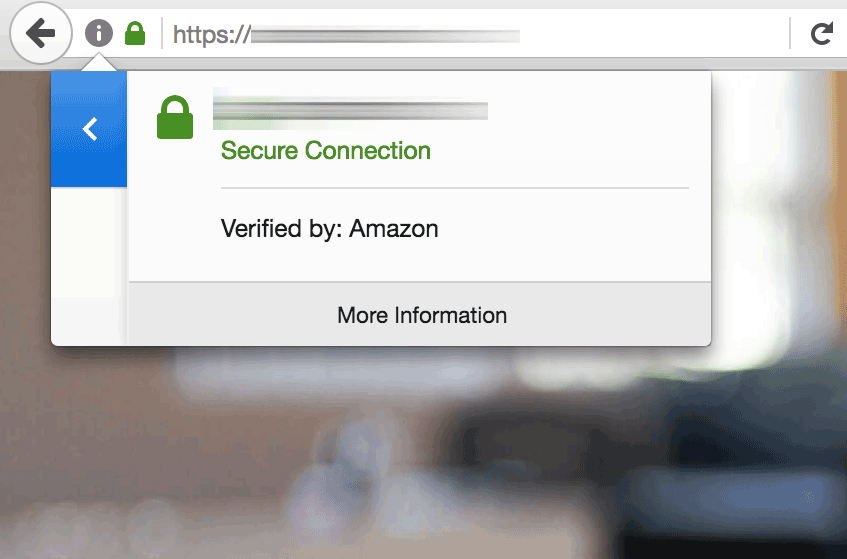
### Step 5: Test The Load Balancer

Verify that the changes to your domain name record have propagated by using the [Global DNS Propagation Checker](https://www.whatsmydns.net/) and entering your domain name into the search field. After confirming that the domain name now points to the public IP address of the load balancer, you can test it as follows (replace the DOMAIN placeholder with the correct domain name):

* Browsing to http://DOMAIN should result in the load balancer displaying the insecure welcome page of the Bitnami application.

[](https://docs.bitnami.com/images/img/how_to_guides/configure-elb-ssl-aws/elb-aws-17-a04ec713.png)

* Browsing to https://DOMAIN should result in the load balancer displaying the secure welcome page of the Bitnami application. Clicking the padlock icon in the browser address bar should display the details of the domain and SSL certificate.

[](https://docs.bitnami.com/images/img/how_to_guides/configure-elb-ssl-aws/elb-aws-18-cf20ad85.png)

In both cases, the request to the HTTP or HTTPS port of the load balancer is internally transferred to, and satisfied by, the Bitnami application instance.

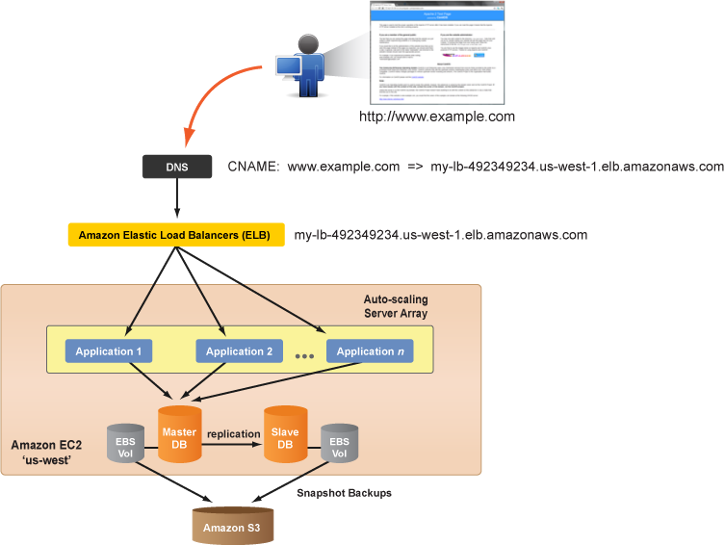
Your Elastic Load Balancer has now been configured with an SSL certificate for your Bitnami application running on AWS.

# **EC2 Load Balancing**

## Create an AWS Elastic Load Balancer

### Overview

You can use Amazon's Elastic Load Balancers (ELB) for load balancing purposes. Windows and Linux ServerTemplates can automatically register and unregister when servers are launched in a deployment or autoscaling array. For mission critical applications using AWS Elastic Load Balancers, you should configure the ELB to service application servers running in multiple availability zones.



Use the following procedure to create a new Amazon Elastic Load Balancer (ELB).

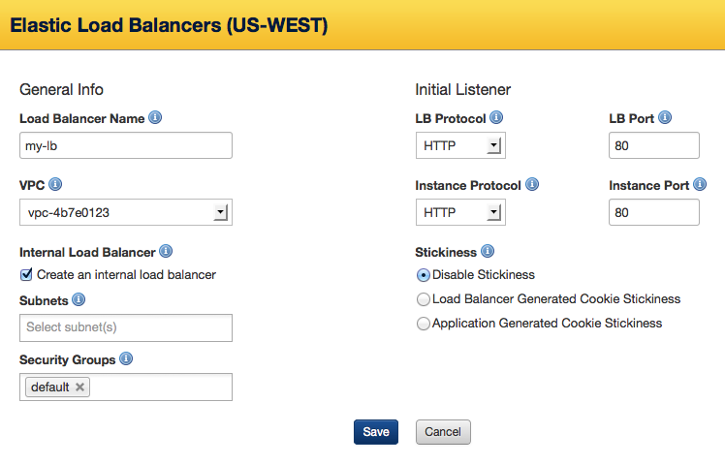
### Prerequisites

* If you have valid Amazon Web Services (AWS) EC2 credentials, you are automatically granted the Elastic Load Balancer service. No additional service sign ups are required.
* 'actor' user role privileges

### Steps

#### Create a Load Balancer

1. Go to **Clouds** > AWS Region > **Load Balancing**
2. Click **Create ELB** and fill out the required form fields:



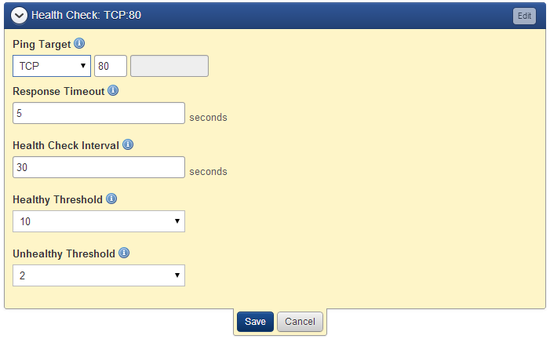
General Info:

* **Load Balancer name** - The name of our Elastic Load Balancer. It must be unique within your AWS account.
* **VPC** - The Virtual Private Cloud you'd like to associate to this load balancer.
* **Internal Load Balancer** - Select this option to create an internal load balancer. A DNS name will be created and it will contain the private IP address of the load balancer. This option is only available if you set a value for VPC.
* **Subnets** - Your load balancer can interface with instances in one or more subnets. Only subnets in the selected VPC are available. This option is only visible if you set a value for VPC.
* **Security Groups** - One or more security groups to manage traffic communicating with your load balancer. This option is only visible if you set a value for VPC.
* **Availability Zone(s)** - Select all of the availability zones for which the ELB will service. (Tip: Click in the box to view and select additional zones.) An ELB will only load balance across instances in the selected availability zones. An ELB is region-specific; it cannot load balance across multiple regions. For example, you cannot load balance your application traffic amongst instances in the 'us-east' and 'us-west' regions. Note: This field is not displayed if you have selected a value for VPC.

Default Listener:

* **LB Protocol** - The routing transport protocol that will be used by the ELB. (TCP, HTTP, HTTPS)
* **LB Port** - External port that the ELB will listen on to accept client requests. Valid ports are 80, 443, 1024-65535.
  + For an example (v13.5 LTS) 3-tier deployment, use **HTTP:80**.
* **Instance Protocol** - The routing transport protocol that the ELB will use to connect to the instances (i.e. application servers) in its load balancing pool. (HTTP, HTTPS, TCP, SSL) If you are using one of the application ServerTemplates (v12+) published by RightScale, select 'HTTP' for the protocol. Note: The Load Balancer and Instance Protocols must match. You can create additional listener rules after you create the ELB.]
* **Instance Port** - The port that the ELB will use to send traffic to the application instances. Valid ports are 1 to 65535. The Instance Port does not have to match the Load Balancer Port. If you are using one of the application ServerTemplates (v12+) published by RightScale, the default application listener port is '8000'.
  + For an example (v13.5 LTS) 3-tier deployment, use **HTTP:8000**.
* **Stickiness** - This is where you can create a session sticky policy for your load balancer (stickiness meaning that once a session starts, all requests will attempt to routed subsequent to the same server for that given session). You can select one of the following options:
  + **Disable Stickiness** - Does not allow session stickiness the load balancer and application server.
  + **Load Balancer Generated Cookie Stickiness** - A special cookie is used to track the backend server instance for each request. When the LB receives a request, it checks to see if this cookie is present in the request. If so, the load balancer sends the request to the application server specified in the cookie. If not, the request is sent to a server chosen based on the existing load balancing algorithm. This option creates a stickiness policy with sticky session lifetime controlled by the lifetime of the browser or a specified expiration period. This only works with HTTP/HTTPS listeners.
  + **Application Generated Cookie Stickiness** - This is similar to the Load Balancer Generated Cookie Stickiness option, with the following exception: once a cookie is explicitly removed or expires, the session stops being sticky until a new application cookie is issued.

1. Click **Save**.
2. Configure the Health Check test for the ELB and click **Save**.



#### Add Additional Listeners (optional)

Elastic Load Balancers support both HTTP and HTTPS (SSL) requests. When you create an ELB, you must define a default listener. If you wish to set up additional listeners (e.g. HTTPS), you can configure them after the ELB is created.

![File:12-Guidesdiag-3tierELB\_http\_https-v1.png)

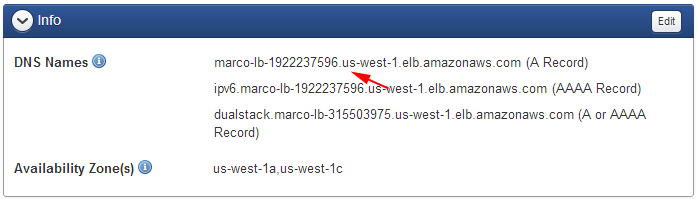
Listener ports should be defined when your ELB is created. If you are setting up an ELB to load balance across applications servers launched with one of RightScale's v12.11 or v13.5 LTS ServerTemplates, you may need to modify your ELB and add additional listeners, as necessary.

1. Click on the Elastic Load Balancer you would like to modify.
2. Check the configuration of the application ServerTemplate that will be used to launch the application servers that will connect to the ELB. Make sure that the application servers are configured to listen for requests from the ELB on the correct port. You will need to set up the listener appropriately, depending on how the application servers will be configured. (e.g. Application Listen Port (8000), LB\_PORT (80), etc.)
3. To configure **HTTP** access, add a listener to forward requests to the appropriate port (e.g. 8000) for the application server. Click **Add Listener** to create a new listener for the ELB.
   * **LB Protocol**: Select 'HTTP'
   * **LB Port**: Enter '80'
   * **Instance Protocol**: Select 'HTTP'
   * **Instance Port**: Use '8000' (for v12.11 or v13.5 LTS ServerTemplates); use '8080' (for v14 ServerTemplates)
   * **Stickiness**: You can choose either to have stickiness disabled or generate a sticky session based on either load balancer or application generated HTTP cookies.
4. To allow **HTTPS (SSL)** access, add a listener to forward requests to the appropriate port (e.g. 8000) for the application servers. Click **Add Listener** to create a new listener for the ELB.
   * **LB Protocol**: Select 'HTTPS'
   * **LB Port**: Enter '443'
   * **Instance Protocol**: Select 'HTTPS'
   * **Instance Port**: Use '8000' (for v12.11 or v13.5 LTS ServerTemplates); use '8080' (for v14 ServerTemplates)
   * **Stickiness**: You can choose either to have stickiness disabled or generate a sticky session based on either load balancer or application generated HTTP cookies.
   * **Certificate**: Select an SSL Certificate. See [Create an AWS Server Certificate](http://docs.rightscale.com/cm/dashboard/clouds/aws/actions/server_certificates_actions.html#create_an_aws_server_certificate.html).

#### Update the DNS Records

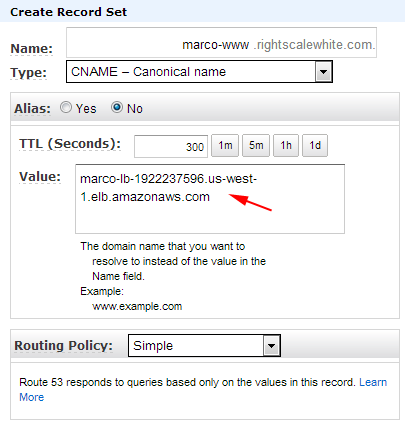
Once you've created your ELB, find its DNS name (e.g. elbname-12345678.us-east-1.elb.amazonaws.com) and use it to create a CNAME with your DNS provider. For an ELB, you cannot create a DNS record that points to an IP address because it could change over time.

1. Go to the Elastic Load Balancer's **Info** tab in the RightScale Dashboard.
2. Copy the DNS name for the ELB. (e.g. elb-name-2005463590.us-west-1.elb.amazonaws.com)



1. Create a CNAME with your DNS provider (e.g. AWS Route 53, DNS Made Easy, DynDNS, etc.)

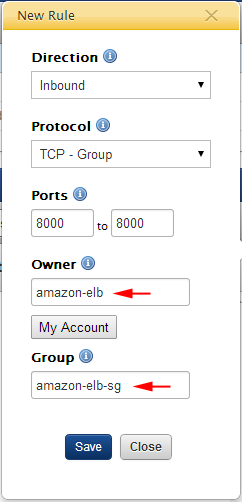
**Example: Amazon Route 53**



#### Create a Security Group for Application Servers

The next step is to create or update a security group that will be used by application servers to allows ingress communication from an ELB. By default, RightScale's Linux-based application ServerTemplates (e.g. PHP, Rails, Tomcat) listen on TCP port 8000 and the Windows-based application ServerTemplate (e.g. Microsoft IIS App) listens on TCP port 80.

1. Create or modify a security group in the AWS region where the ELB will service requests to the application servers. Later, you will configure the application servers to use this security group in order to connect to the ELB.
2. Create the desired port permissions. At a minimum, you will need to open up port 80 for HTTP access. You can also open up 443 for HTTPS (SSL) access. Specify the following information to create a permission for an ELB.
   * **Direction**: Inbound
   * **Protocol**: TCP - Group
   * **Ports**:
   * For v12.11 or v13.5 LTS ServerTemplates, use 8000..8000 (PHP, Rails, Tomcat, Django); 80..80 (Microsoft IIS)
   * For v14 ServerTemplates, use 8080..8080 (PHP, Rails, Tomcat, Django); 80..80 (Microsoft IIS)
   * **Owner**: amazon-elb
   * **Group**: amazon-elb-sg



1. Click **Save**.
2. Later, when you set up your application servers and/or server array, be sure to use the above security group for the application servers will connect to the Elastic Load Balancer.

**Note:**

You cannot add a new security group to a running instance. In such cases, it's recommended that you add the security group to the next server and relaunch the server.

## Add an Instance to an AWS Elastic Load Balancer

### Overview

Once you have created and configured your load balancer, you can add instances to it. Use the following procedure to add an instance to an AWS Elastic Load Balancer.

### Prerequisites

* A configured AWS Elastic Load Balancer (ELB)

### Steps

1. Navigate to **Clouds** > AWS Region > **Load Balancing** > LoadBalancerName.
2. Click **Add Instance**.
3. Select the instances that you want to add and click **Save**.
4. To remove an instance, click the check box for the instance and select **Remove** from the **Actions** drop down.

## Remove an Instance from an AWS Elastic Load Balancer

### Overview

Once you have created and configured your load balancer and added an instance to it, you may need to remove the instance. This is very simple to do. Use the following procedure to unregister a Server attached to an AWS Elastic Load Balancer (ELB).

### Prerequisites

* A configured AWS Elastic Load Balancer with a server attached to it.

### Steps

1. Navigate to **Clouds** > AWS Region > **Load Balancing** > LoadBalancerName
2. Expand the **Instances** tab.
3. Select the instances that you want to remove.
4. Select **Remove** from the Actions drop down.

## Add Listeners to an Elastic Load Balancer

### Overview

When you first create a new Elastic Load Balancer, it's automatically set up with one listener already predefined. By default, an ELB will be configured to listen over HTTP on port 80 and forward traffic to port 80 on the receiving EC2 instances, which are typically your application servers.

* HTTP; 80->80

To add additional listeners or make changes to existing ones, use the **Listeners** tab of the ELB. Use the following procedure to add listeners to an existing Amazon Elastic Load Balancer (ELB).

### Prerequisites

* An existing ELB
* 'actor' user role privileges
* To add a Listener for SSL communications, you'll need a [Server Certificate](http://docs.rightscale.com/cm/dashboard/clouds/aws/server_certificates.html).

### Steps

1. Go to **Clouds** > AWS Region > **Load Balancing**.
2. Select an existing ELB from the list.
3. Click **Add Listener**.

To add a Listener, specify the following information:

* **Load Balancer Protocol** - The routing transport protocol that will be used by the ELB.
  + HTTP
  + HTTPS
  + TCP
  + SSL
* **Load Balancer Port** - External port that the ELB will listen on to accept client requests. Valid ports are 80, 443, 1024-65535.
* **Instance Protocol** - The front end protocol used by the instance.
* **Instance Port** - The port that the ELB will use to send traffic to the application instances. Valid ports are 1 to 65535. The Instance Port does not have to match the Load Balancer Port.
* **Certificate** (for HTTPS and SSL only) - Select or upload the Server Certificate that will be used for encryption purposes. See [Server Certificates](http://docs.rightscale.com/cm/dashboard/clouds/aws/server_certificates.html).
* **Stickiness** - (HTTP and HTTPS only) - Use stickiness to bind a user's session to a specific application instance. Also known as session affinity.

## Create a New Sticky Session

### Overview

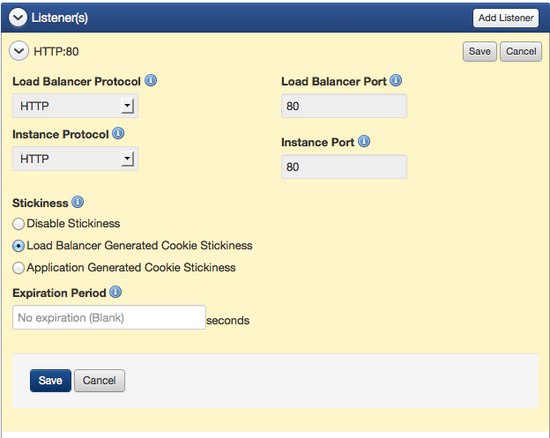
By default, AWS ELB distributes requests to Instances running your application based on the lowest load. However, you can create a Sticky Session so that a request is bound to the same Instance for the length of the entire session. Sticky sessions are based on either load balancer or application generated HTTP cookies. This tutorial will step you through how to create a Sticky Session for your ELB. Use the following procedure to create a new Sticky Session policy for an AWS Elastic Load Balancer (ELB).

### Prerequisites

* A created ELB and two or more front end Servers that have registered with the ELB.
* There is an AWS-imposed limit of 20 policies per ELB.
* Load Balancer protocol set to either HTTP or HTTPS

### Steps

1. Navigate to **Clouds** > AWS Region > **Load Balancing**.
2. Click the ELB that you want to create a Sticky Session for.
3. Expand the **Listener(s)** tab.
4. Select either **Load Balancer Generated Cookie Stickiness** or **Application Generated Cookie Stickiness** to determine the basis for stickiness and set the following parameters:
   * For Load Balancer Generated Cookie Stickiness, set the Expiration Period (in seconds), after which stickiness will be disabled. Leaving this field blank means that stickiness will not expire.
   * For Application Generated Cookie Stickiness, enter the Cookie Name.
5. Click **Save**.



### Additional ELB Information

The following is an excerpt from the AWS ELB Developer Guide (Sept 2010), where it speaks to the Cookie Type:

**Load Balancer-Generated HTTP Cookies**

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

For more information about the policy configuration for load balancer-generated HTTP cookies, see [Create LB Cookie Stickiness Policy](http://docs.aws.amazon.com/ElasticLoadBalancing/latest/APIReference/API_CreateLBCookieStickinessPolicy.html).

**Application-Generated HTTP Cookies**

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

For more information about the policy configuration for application-generated HTTP cookies, see [Create App Cookie Stickiness Policy](http://docs.aws.amazon.com/ElasticLoadBalancing/latest/APIReference/API_CreateAppCookieStickinessPolicy.html).

### Post Tutorial Steps

#### Disable a Sticky Session

* Navigate to the Load Balancer
* Expand the **Listener(s)** tab.
* Select **Disable Stickiness**

## Edit an AWS Elastic Load Balancer Configuration

### Overview

There are several ways you can modify your load balancer's configuration that will affect the way it operates. One of the more common edits you can perform from the Dashboard is changing (adding to or deleting from) the Availability Zones your load balancer distributes traffic in. Use the following procedure to edit the configuration of an AWS Elastic Load Balancer (ELB) in the RightScale Dashboard.

### Prerequisites

* EC2 credentials, and a previously created AWS Elastic Load Balancer

### Steps

1. Navigate to **Clouds** > AWS Region > **Load Balancing**.
2. Select the correct LoadBalancerName link.
3. You will be given the option to modify these areas:
   * **Info** - This section displays the DNS Name of the LB and the Availability Zone(s) associated to the LB. You can edit this section to modify the zones the LB should service traffic in.
   * **Health Check** - Based on the information configured in this section, ELB routinely checks the health of the load balancer within an AWS instance. You can specify either TCP, HTTP, HTTPS, or SSL and a valid port between one and 65535. The default is TCP:80. Additionally, you can specify the following:
   * **Response Timeout** - Time to wait when receiving a response from the health check (between two and 60 seconds)
   * **Health Check Interval** - Amount of time between health checks (0.1 min. to 5 min.)
   * **Unhealthy Threshold** - Number of consecutive health check failures before an instance is declared as unhealthy (two to 10)
   * **Healthy Threshold** - Number of consecutive health check successes before an instance is declared as healthy (two to 10)
   * **Instance(s)** - You can view current instances associated to your LB and add instances to your LB.
   * **Listener(s)** - You can view current listeners associated to your LB and add new listeners to your LB.

**Note:**

Consult the AWS documentation for additional information. (Our default values align with AWS and the above values could possibly become dated over time.)

* After editing one of all of these sections, don't forget to **Save** your changes.

### Post Tutorial Steps

To view the current configuration of your load balancer, navigate to:

**Clouds** > AWS Region > **Load Balancing** > LoadBalancerName

## Delete an AWS Elastic Load Balancer Configuration

### Overview

Use the following procedure to delete an AWS Elastic Load Balancer in the RightScale Dashboard.

### Prerequisites

* EC2 credentials, and a previously created AWS Elastic Load Balancer.

### Steps

1. Navigate to **Clouds** > AWS Region > **Load Balancing**.
2. Select the correct LoadBalancerName checkbox.
3. Select the **Actions** dropdown and **Delete**.
4. Confirm deletion in the dialog window.