LAB2: CREATING YOUR FIRST CLASSIC LOAD BALANCER

What is ELB?

Elastic Load Balancing allows the distribution of incoming traffic to your Amazon AWS infrastructure across multiple instances. This represents a great tool in avoiding failures in your applications and web traffic. ELB automatically detects fails in your EC2 instances and redirects traffic to other available instances.

During this lab you'll learn to create and use your first ELB instance to balance the HTTP traffic between two EC2 instances. You'll also gain valuable understanding of the Classic Load Balancer behavior during an instance outage.

Pre-requisites:

This is a beginner level Lab; however, in order to follow the next steps you should be able to:

- Describe and launch EC2 instances
- Describe and create a VPC
- Describe, create and configure Security Groups

Learning Objectives:

By the end of this lab you should be able to:

- Create and configure a Classic Load Balancer
- Add existing EC2 instances and a Classic Load Balancer
- Restrict direct access to EC2 instances using Security Groups

AGENDA:

You'll build and learn following these steps:

Log In to the Amazon Web Service Console
Your first step to start the laboratory experience

Classic Load Balancer planning
IN this step we will plan the deployment for the Classic Load Balancer

Create a Classic Load Balancer and register EC2 instances
This step will guide you through the process of creating a Classic Load Balancer

Configuring security groups for ELB
This step will guide you through configuring security groups for you EC2 instances running behind a Classic Load Balancer

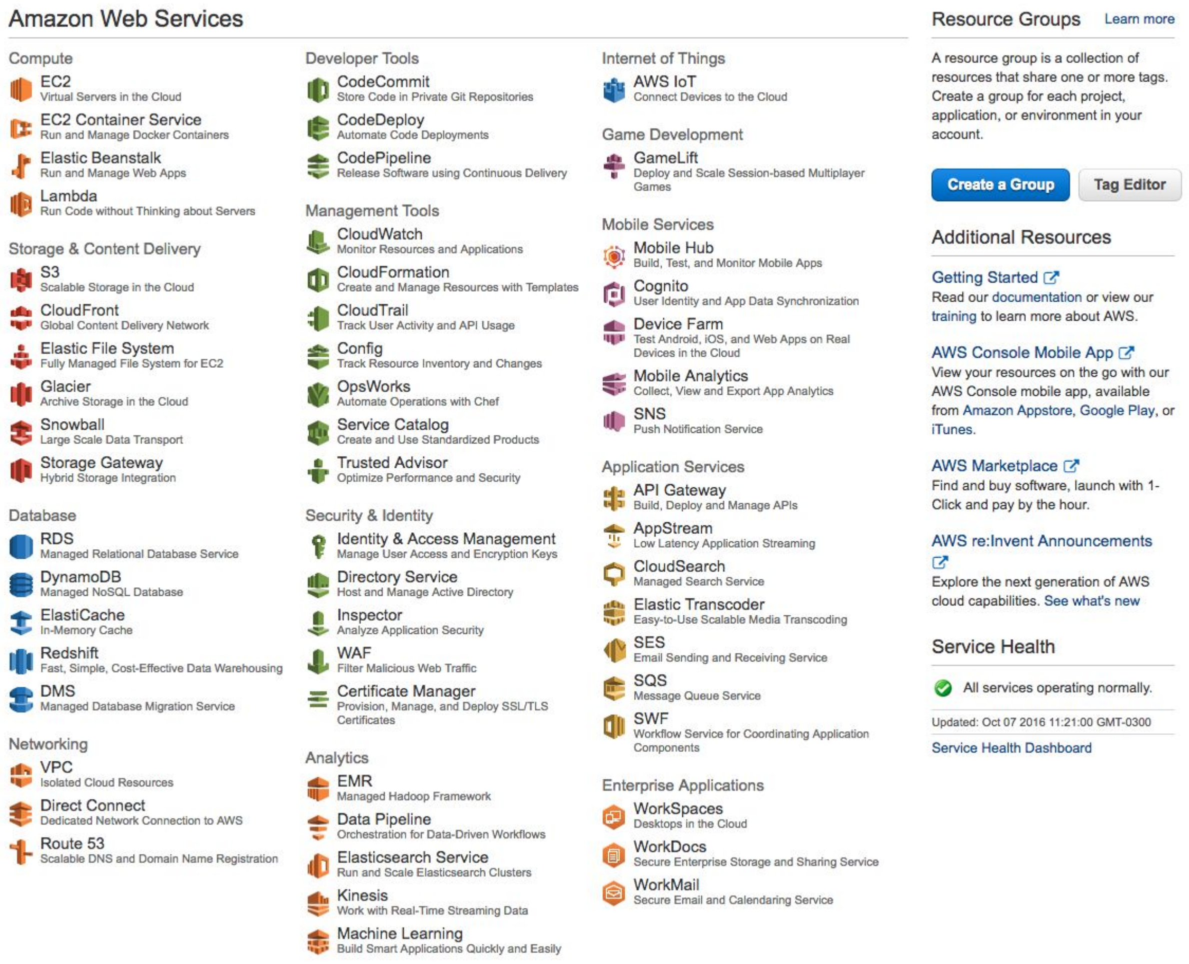
Checking a load balancer's behavior during instance failures
In this step, we will test LB's behavior during instance failures

Monitoring your Classic Load Balancer
In this step we will take a look at the metrics on a Classic Load Balancer

LAB WORK:

Step 1 Log In to the Amazon Web Service Console

This laboratory experience is about Amazon Web Services and you will use the AWS Management Console in order to complete all the lab steps.



Amazon Web Services

Compute
- EC2 — Virtual Servers in the Cloud
- EC2 Container Service — Run and Manage Docker Containers
- Elastic Beanstalk — Run and Manage Web Apps
- Lambda — Run Code without Thinking about Servers

Storage & Content Delivery
- S3 — Scalable Storage in the Cloud
- CloudFront — Global Content Delivery Network
- Elastic File System — Fully Managed File System for EC2
- Glacier — Archive Storage in the Cloud
- Snowball — Large Scale Data Transport
- Storage Gateway — Hybrid Storage Integration

Database
- RDS — Managed Relational Database Service
- DynamoDB — Managed NoSQL Database
- ElastiCache — In-Memory Cache
- Redshift — Fast, Simple, Cost-Effective Data Warehousing
- DMS — Managed Database Migration Service

Networking
- VPC — Isolated Cloud Resources
- Direct Connect — Dedicated Network Connection to AWS
- Route 53 — Scalable DNS and Domain Name Registration

Developer Tools
- CodeCommit — Store Code in Private Git Repositories
- CodeDeploy — Automate Code Deployments
- CodePipeline — Release Software using Continuous Delivery

Management Tools
- CloudWatch — Monitor Resources and Applications
- CloudFormation — Create and Manage Resources with Templates
- CloudTrail — Track User Activity and API Usage
- Config — Track Resource Inventory and Changes
- OpsWorks — Automate Operations with Chef
- Service Catalog — Create and Use Standardized Products
- Trusted Advisor — Optimize Performance and Security

Security & Identity
- Identity & Access Management — Manage User Access and Encryption Keys
- Directory Service — Host and Manage Active Directory
- Inspector — Analyze Application Security
- WAF — Filter Malicious Web Traffic
- Certificate Manager — Provision, Manage, and Deploy SSL/TLS Certificates

Analytics
- EMR — Managed Hadoop Framework
- Data Pipeline — Orchestration for Data-Driven Workflows
- Elasticsearch Service — Run and Scale Elasticsearch Clusters
- Kinesis — Work with Real-Time Streaming Data
- Machine Learning — Build Smart Applications Quickly and Easily

Internet of Things
- AWS IoT — Connect Devices to the Cloud

Game Development
- GameLift — Deploy and Scale Session-based Multiplayer Games

Mobile Services
- Mobile Hub — Build, Test, and Monitor Mobile Apps
- Cognito — User Identity and App Data Synchronization
- Device Farm — Test Android, iOS, and Web Apps on Real Devices in the Cloud
- Mobile Analytics — Collect, View and Export App Analytics
- SNS — Push Notification Service

Application Services
- API Gateway — Build, Deploy and Manage APIs
- AppStream — Low Latency Application Streaming
- CloudSearch — Managed Search Service
- Elastic Transcoder — Easy-to-Use Scalable Media Transcoding
- SES — Email Sending and Receiving Service
- SQS — Message Queue Service
- SWF — Workflow Service for Coordinating Application Components

Enterprise Applications
- WorkSpaces — Desktops in the Cloud
- WorkDocs — Secure Enterprise Storage and Sharing Service
- WorkMail — Secure Email and Calendaring Service

Resource Groups — Learn more

A resource group is a collection of resources that share one or more tags. Create a group for each project, application, or environment in your account.

Create a Group | Tag Editor

Additional Resources

Getting Started
Read our documentation or view our training to learn more about AWS.

AWS Console Mobile App
View your resources on the go with our AWS Console mobile app, available from Amazon Appstore, Google Play, or iTunes.

AWS Marketplace
Find and buy software, launch with 1-Click and pay by the hour.

AWS re:Invent Announcements
Explore the next generation of AWS cloud capabilities. See what's new

Service Health

All services operating normally.

Updated: Oct 07 2016 11:21:00 GMT-0300

Service Health Dashboard

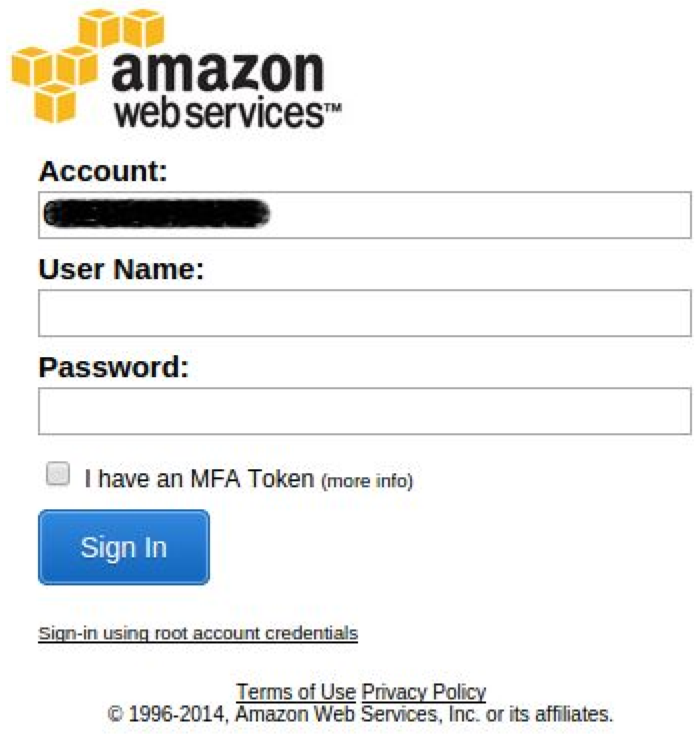
The AWS Management Console is a web control panel for managing all your AWS resources, from EC2 instances to SNS topics. The console enables cloud management for all aspects of the AWS account, including managing security credentials, or even setting up new IAM Users.

Log in to the AWS Management Console

In order to start the laboratory experience, open the Amazon Console by clicking this button:

https://aws.amazon.com

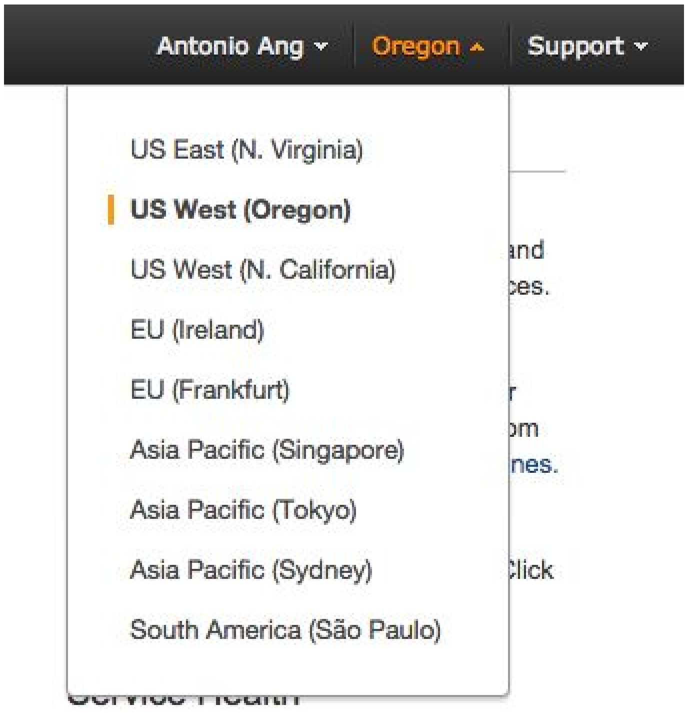
We created a Console User just for you. Log in with the username **student** and the password **Ca1_9iUsNImj** .


Account:
[redacted]
User Name:
Password:
☐ I have an MFA Token (more info)
Sign In
Sign-in using root account credentials
Terms of Use Privacy Policy
© 1996-2014, Amazon Web Services, Inc. or its affiliates.

Select the right AWS Region

Amazon Web Services is available in different regions all over the world, and the console lets you provision resources across multiple regions. You usually choose a region that best suits your business needs to optimize your customer's experience, but you must use the region **US West (Oregon)** for this laboratory.

You can select the **US West (Oregon)** region using the upper right dropdown menu on the AWS Console page.



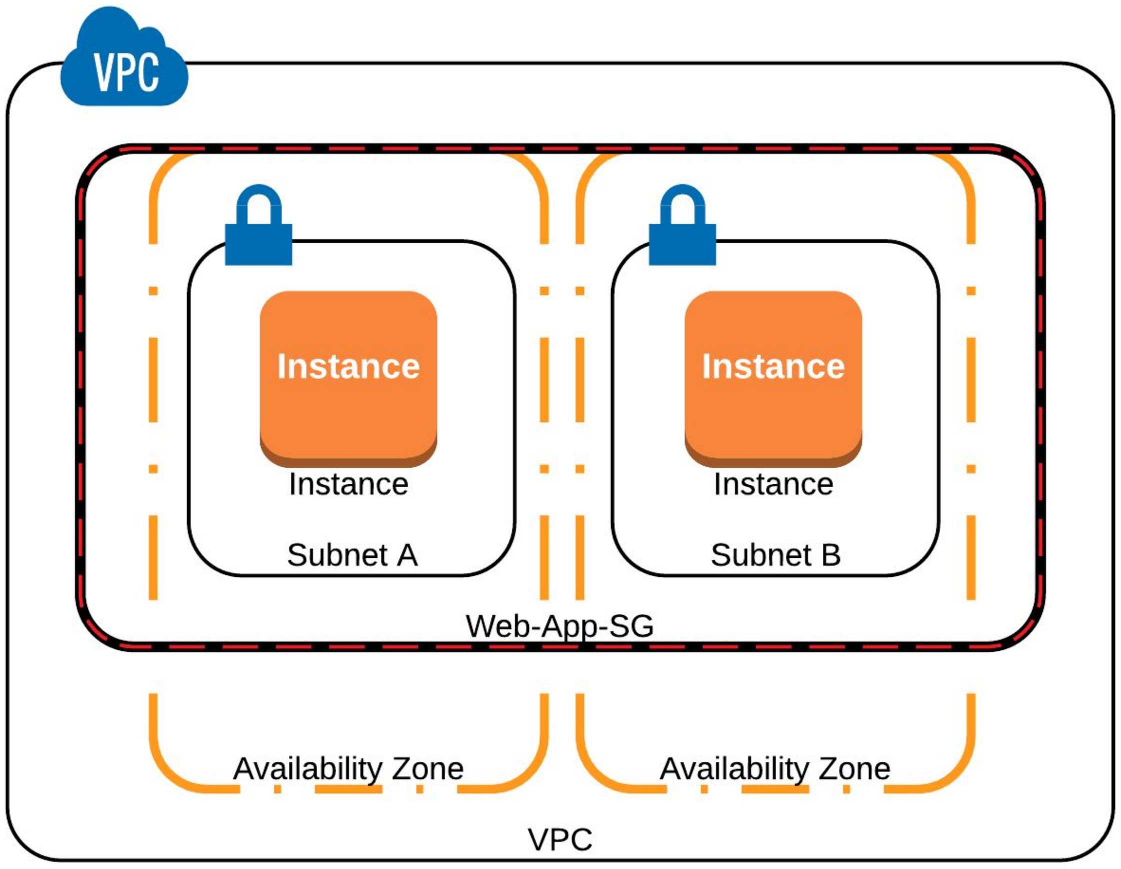
Step 2 Classic Load Balancer planning

The Elastic Load Balancing Service automatically distributes incoming application traffic across multiple Amazon EC2 instances. It enables you to achieve fault tolerance in your applications, seamlessly providing the required amount of load balancing capacity needed to route application traffic.

Elastic Load Balancing offers two types of load balancers both of which 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. The Classic Load Balancer is ideal for simple load balancing of traffic across multiple EC2 instances, while the Application Load Balancer is ideal for applications needing advanced routing capabilities, micro services, and container-based architectures. Application Load Balancer offers the ability to route traffic to multiple services or load balance across multiple ports on the same EC2 instance. In this lab, we are going to focus on the **Classic Load Balancer**.

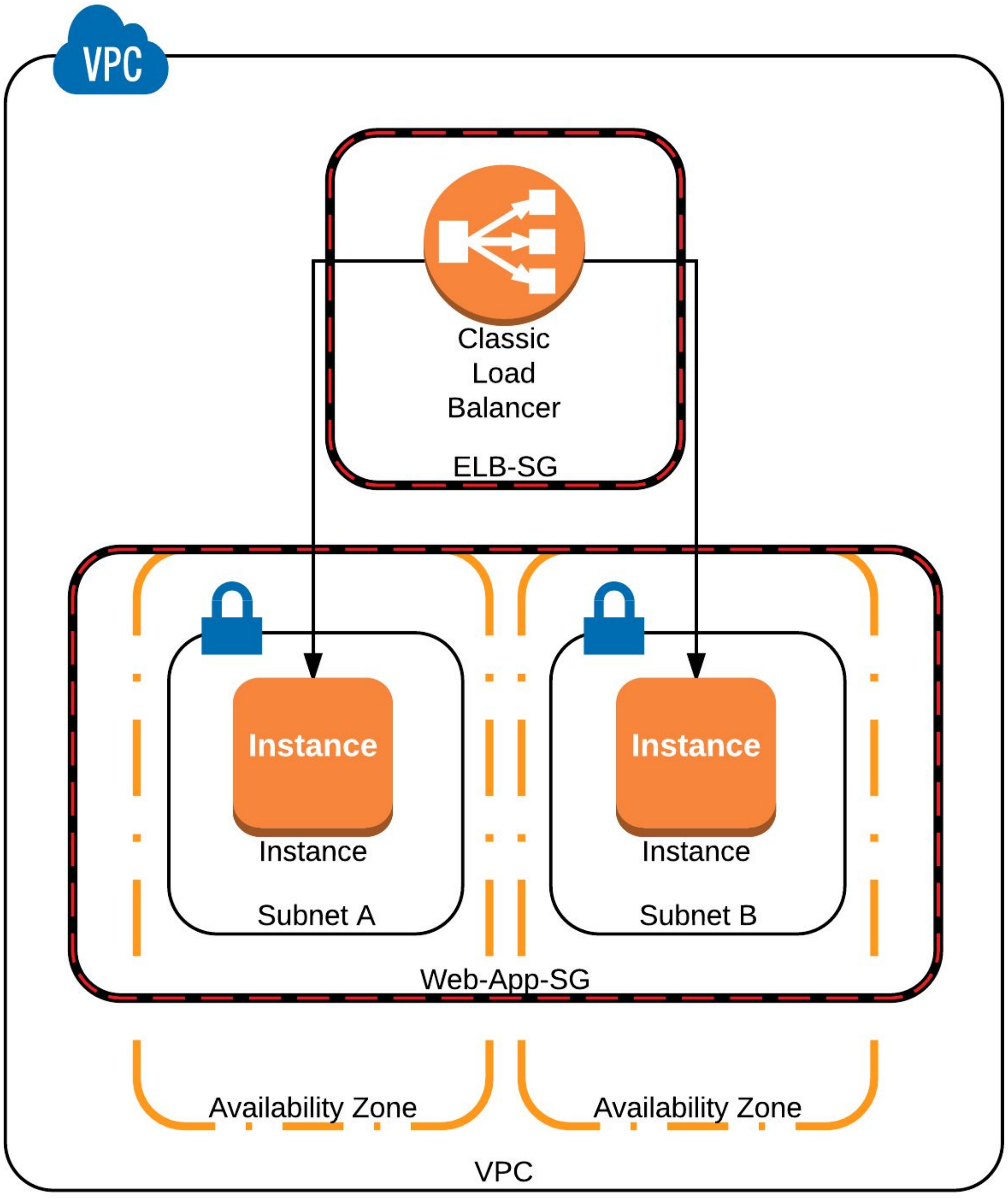
Planning

Before going ahead and creating a Load Balancer (LB), let's take a look at an overview of our current infrastructure. When you connected to the AWS account provided in the former step, you had a few things that were already deployed. This is the current infrastructure that was already deployed for you:



You already have a VPC with some subnets and 2 EC2 instances running inside the VPC in different Availability Zones. Both instances are inside the same Security Group called `Web-App-SG`, which is allowing HTTP access from port 80 to anywhere (0.0.0.0/0). Each EC2 instance is running the same web application. We want to **configure an LB to create a central point of access to our application**, and we also want to configure our architecture in a way that **users can only access the application through the ELB**.

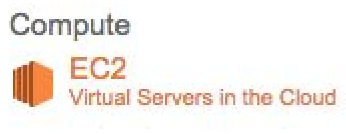
In the end, we should have a solution similar to this one:



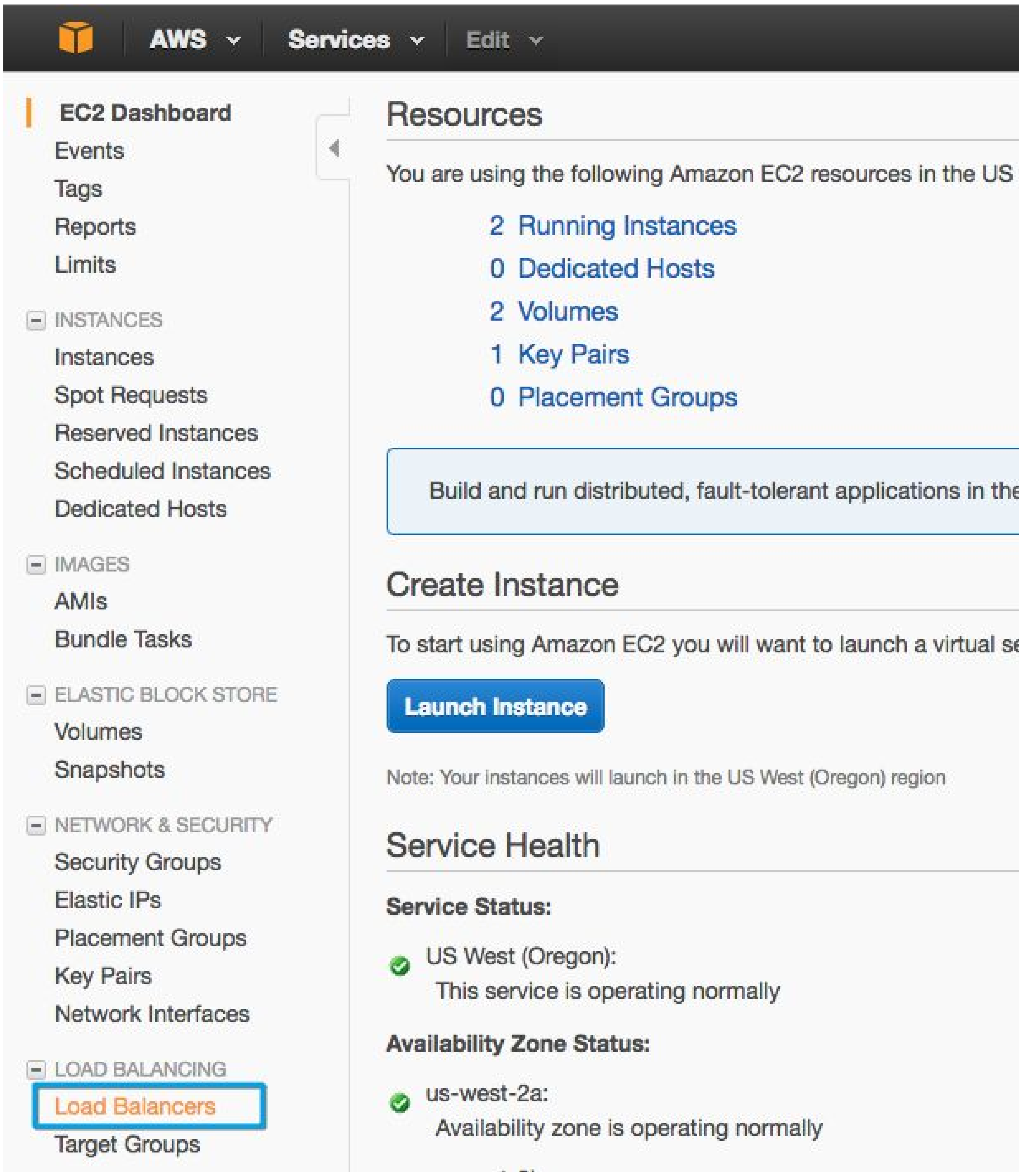
To do that we will have to create and configure a Classic Load Balancer, and properly configure the needed Security Groups to make sure that our application will work as expected.

Step 3 Create a Classic Load Balancer and register EC2 instances

Go to the EC2 console:



Click on Load Balancers:



Resources

You are using the following Amazon EC2 resources in the US

2 Running Instances
0 Dedicated Hosts
2 Volumes
1 Key Pairs
0 Placement Groups

Build and run distributed, fault-tolerant applications in the

Create Instance

To start using Amazon EC2 you will want to launch a virtual se

Note: Your instances will launch in the US West (Oregon) region

Service Health

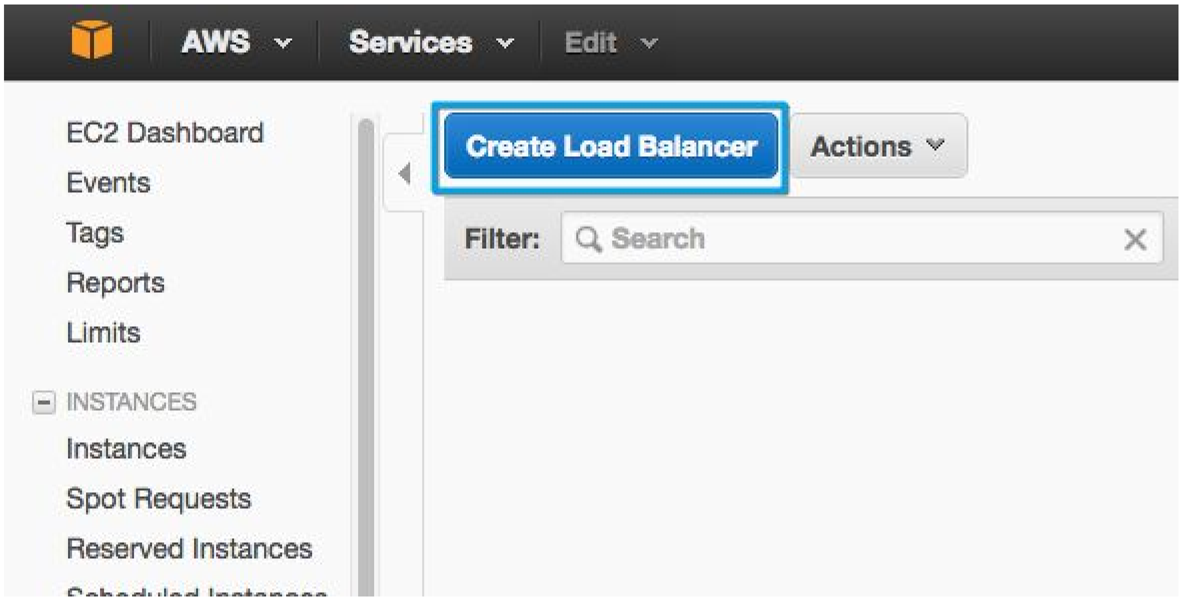
Service Status:

US West (Oregon):
This service is operating normally

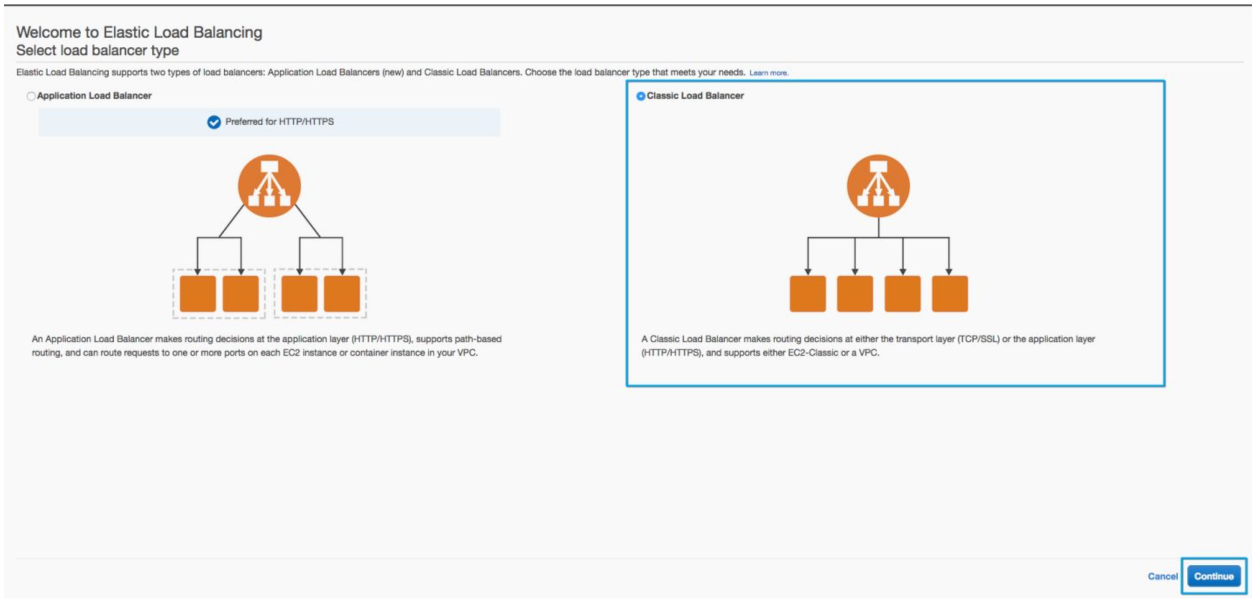
Availability Zone Status:

us-west-2a:
Availability zone is operating normally

Click on **Create Load Balancer**:



As stated before, you can choose from two flavors of ELBs: Application Load Balancer or Classic Load Balancer. In this lab, we will use the Classic Load Balancer, so simply choose the proper one in this step and click on **Continue**.


Welcome to Elastic Load Balancing
Select load balancer type

Elastic Load Balancing supports two types of load balancers: Application Load Balancers (new) and Classic Load Balancers. Choose the load balancer type that meets your needs. Learn more.

Application Load Balancer

Preferred for HTTP/HTTPS

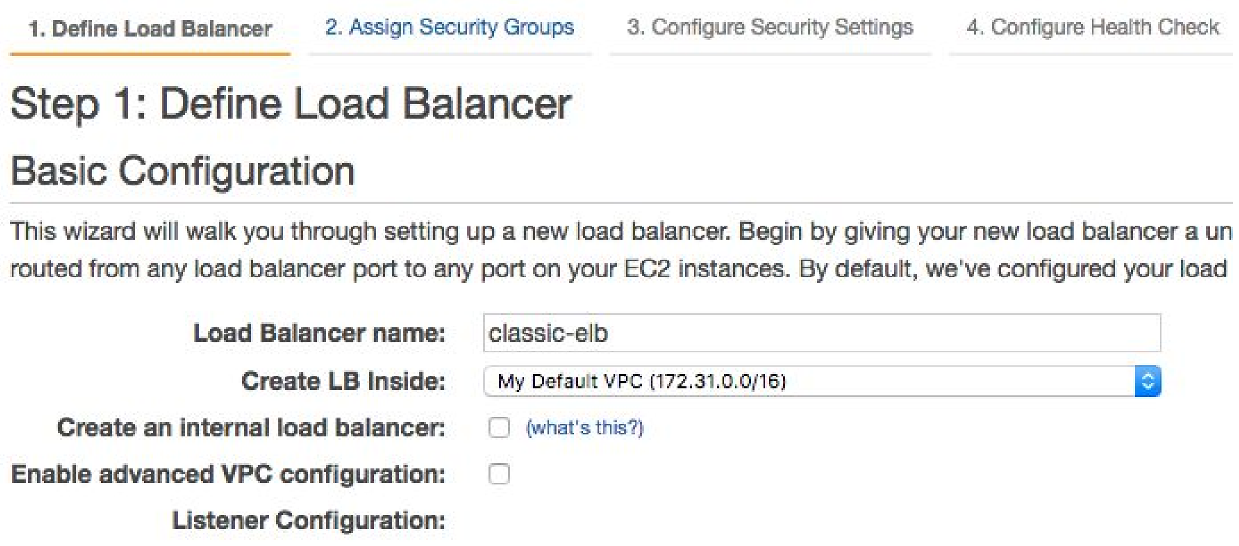
An Application Load Balancer makes routing decisions at the application layer (HTTP/HTTPS), supports path-based routing, and can route requests to one or more ports on each EC2 instance or container instance in your VPC.

Classic Load Balancer

A Classic Load Balancer makes routing decisions at either the transport layer (TCP/SSL) or the application layer (HTTP/HTTPS), and supports either EC2-Classic or a VPC.

Now to start configuring the specifics of the Load Balancer (LB), you will need to follow a 7-step wizard.

In Step 1, you need to specify a name for the LB; this name can be anything that will make sense for you in the future. But be aware of the limitations (Only a-z, A-Z, 0-9 and hyphens are allowed). You need to select the VPC where the LB will live, this VPC should be the same VPC where the EC2 instances are running, you will probably have only the Default VPC in your account, choose this one. You now should have something like this:



1. Define Load Balancer
2. Assign Security Groups
3. Configure Security Settings
4. Configure Health Check

Step 1: Define Load Balancer

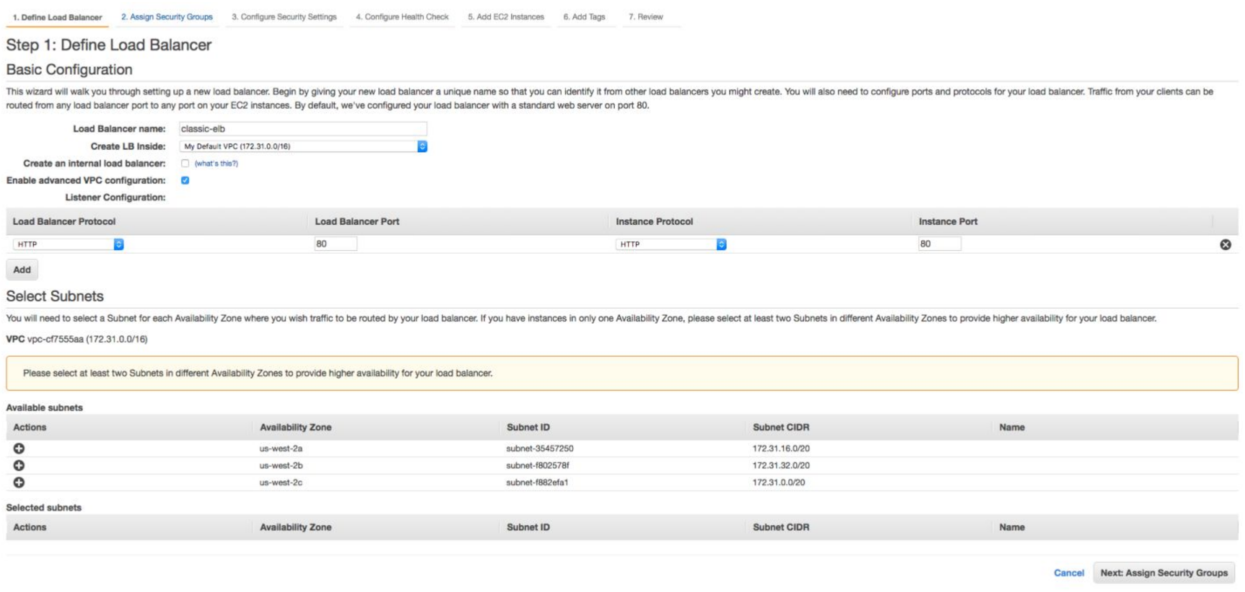
Basic Configuration

This wizard will walk you through setting up a new load balancer. Begin by giving your new load balancer a un
routed from any load balancer port to any port on your EC2 instances. By default, we've configured your load

Load Balancer name:	classic-elb
Create LB Inside:	My Default VPC (172.31.0.0/16)
Create an internal load balancer:	☐ (what's this?)
Enable advanced VPC configuration:	☐
Listener Configuration:	

Some info about the next config points. We will create an LB to receive traffic from the internet and forward to our instances, therefore we need a publicly accessible LB. If you select **Create an internal load balancer** you will be creating a load balancer that won't be publicly accessible - in this case, the LB will only be accessible inside the VPC, which is not the goal, so you should leave this box unchecked.

There are instances running in different availability zones, and we need to configure the LB to work in all subnets where we will be launching web instances. To configure this behavior we need to **Enable advanced VPC configuration** in order to select the subnets we want. After that you will be able to see more options, it will look like this:


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

Step 1: Define Load Balancer

Basic Configuration

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

Load Balancer name:	classic-elb
Create LB Inside:	My Default VPC (172.31.0.0/16)
Create an internal load balancer:	☐ (what's this?)
Enable advanced VPC configuration:	✓
Listener Configuration:	

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

Select Subnets

You will need to select a Subnet for each Availability Zone where you wish traffic to be routed by your load balancer. If you have instances in only one Availability Zone, please select at least two Subnets in different Availability Zones to provide higher availability for your load balancer.

VPC vpc-cf7555aa (172.31.0.0/16)

Please select at least two Subnets in different Availability Zones to provide higher availability for your load balancer.

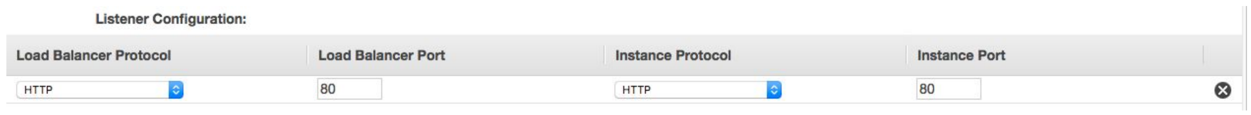
Available subnets

Actions	Availability Zone	Subnet ID	Subnet CIDR	Name
⊕	us-west-2a	subnet-35457250	172.31.16.0/20	
⊕	us-west-2b	subnet-f802578f	172.31.32.0/20	
⊕	us-west-2c	subnet-f882efa1	172.31.0.0/20	

Selected subnets

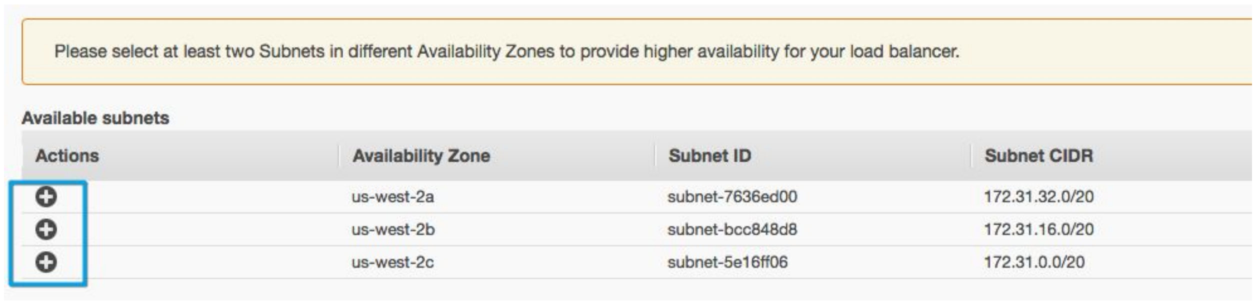
Actions	Availability Zone	Subnet ID	Subnet CIDR	Name

A LB listens to a specific port for requests coming from outside, in this case, the internet, and forwards the request to the instances running behind it on a specific port. In this lab, we will be using the port 80 for HTTP requests for both ELB and the EC2 instances, therefore, the default choice will work for us:


Listener Configuration:

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

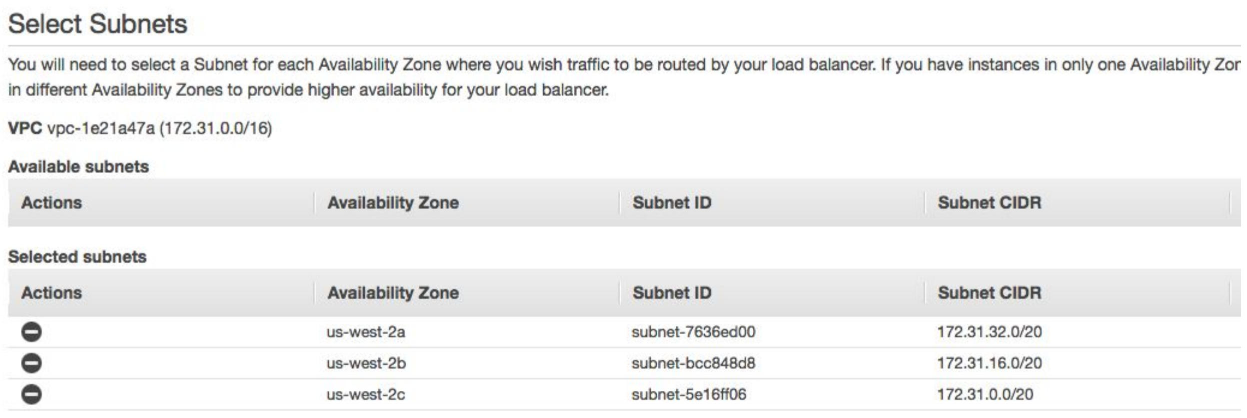
All though there are only 2 instances running in the account, in different Availability Zones, we will want to select ALL the available subnets in this VPC, just in case we want to launch another instance later on in a different Availability Zone. Simply click on the plus button for all the subnets available in this VPC.


Please select at least two Subnets in different Availability Zones to provide higher availability for your load balancer.

Available subnets

Actions	Availability Zone	Subnet ID	Subnet CIDR
⊕	us-west-2a	subnet-7636ed00	172.31.32.0/20
⊕	us-west-2b	subnet-bcc848d8	172.31.16.0/20
⊕	us-west-2c	subnet-5e16ff06	172.31.0.0/20

And in the end, you should see something like this:



Select Subnets

You will need to select a Subnet for each Availability Zone where you wish traffic to be routed by your load balancer. If you have instances in only one Availability Zor
in different Availability Zones to provide higher availability for your load balancer.

VPC vpc-1e21a47a (172.31.0.0/16)

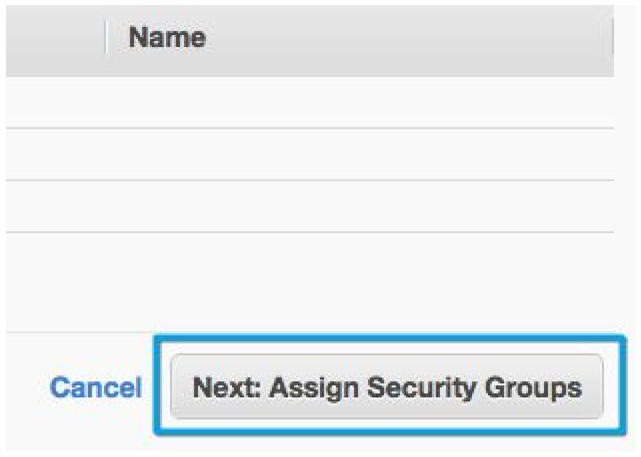
Available subnets

Actions	Availability Zone	Subnet ID	Subnet CIDR

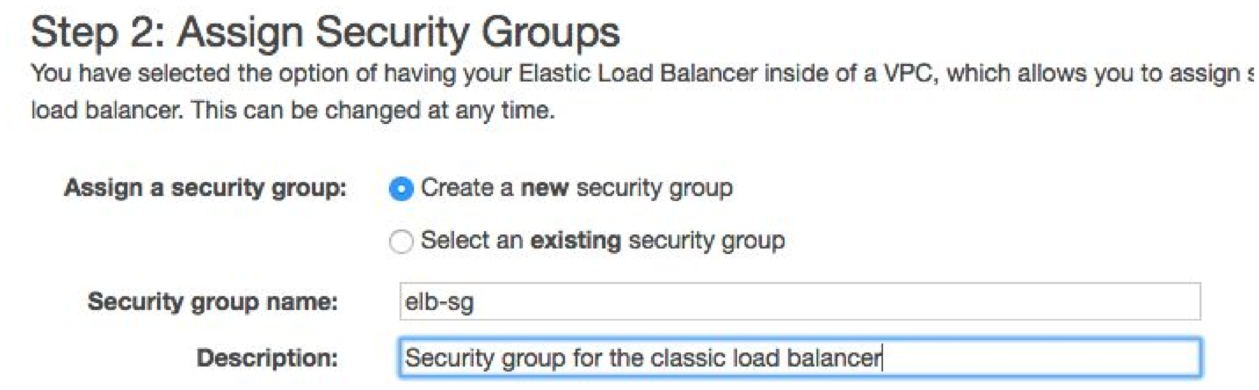
Selected subnets

Actions	Availability Zone	Subnet ID	Subnet CIDR
⊖	us-west-2a	subnet-7636ed00	172.31.32.0/20
⊖	us-west-2b	subnet-bcc848d8	172.31.16.0/20
⊖	us-west-2c	subnet-5e16ff06	172.31.0.0/20

The first step of creating the LB is complete. You can click on **Next: Assign Security Groups**



On step 2, we need to configure a Security Group (SG) for the LB. This SG will be used to manage the security for the LB itself, therefore, since we only defined the port 80 (HTTP) in the listener section of the last step, we will want to create a new SG for the LB that will accept connections coming from anywhere to port 80 of the LB. To do that select **Create a new security group** and provide a name and a quick description for this SG.


Step 2: Assign Security Groups

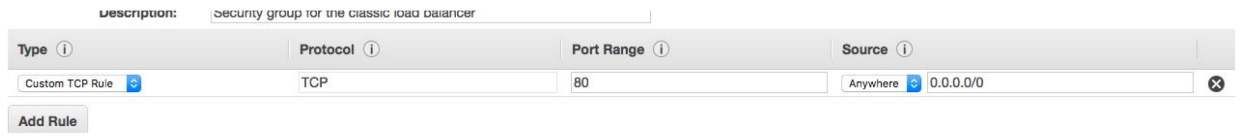
You have selected the option of having your Elastic Load Balancer inside of a VPC, which allows you to assign s
load balancer. This can be changed at any time.

Assign a security group: [✓ Create a **new** security group] Select an **existing** security group

Security group name: elb-sg

Description: Security group for the classic load balancer

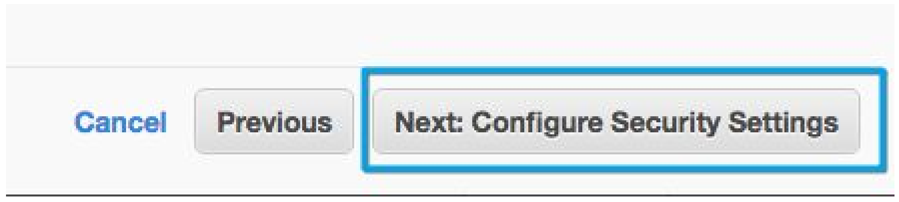
And allow connections coming from Anywhere (0.0.0.0/0) to the port 80 and nothing more



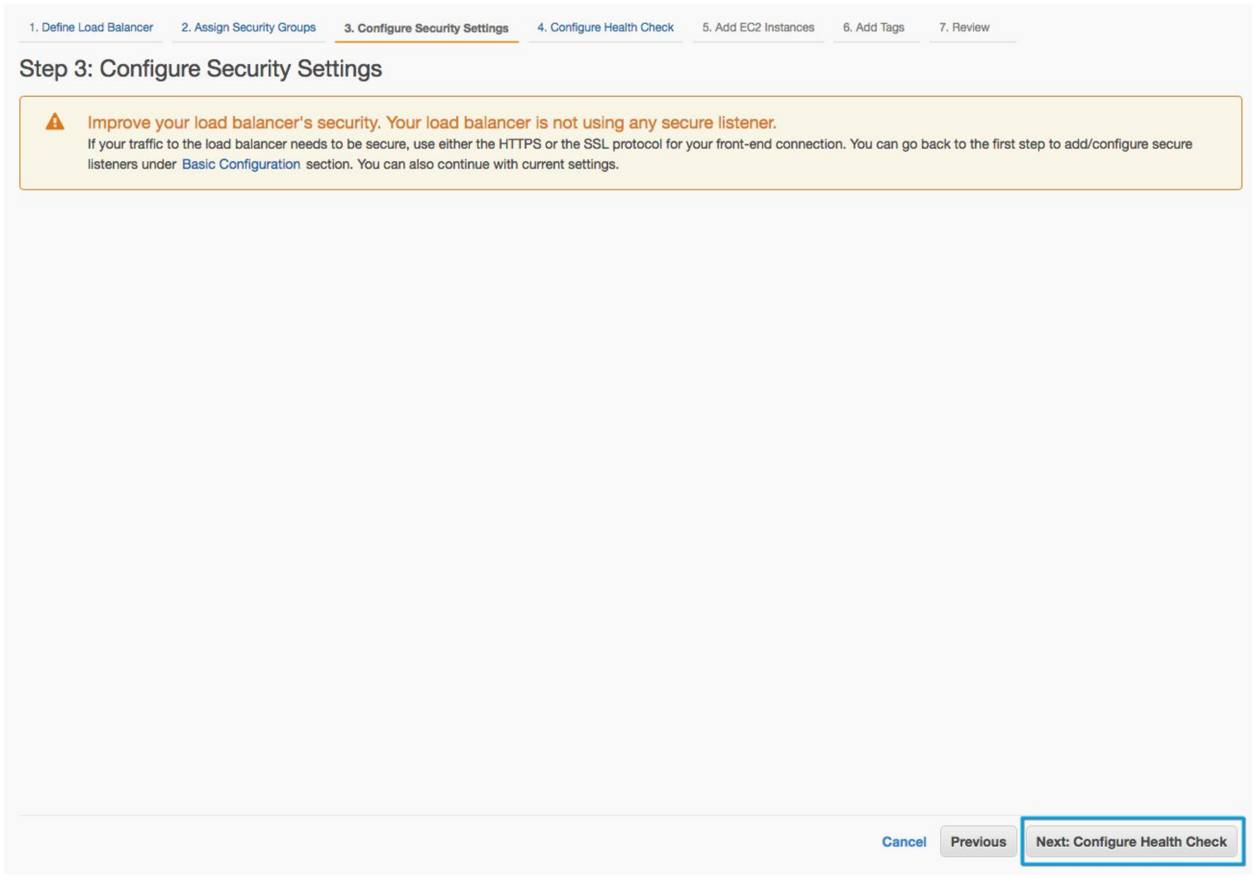
Description: Security group for the classic load balancer

Type	Protocol	Port Range	Source
Custom TCP Rule	TCP	80	Anywhere 0.0.0.0/0

After that, you can click on **Next: Configure Security Settings**



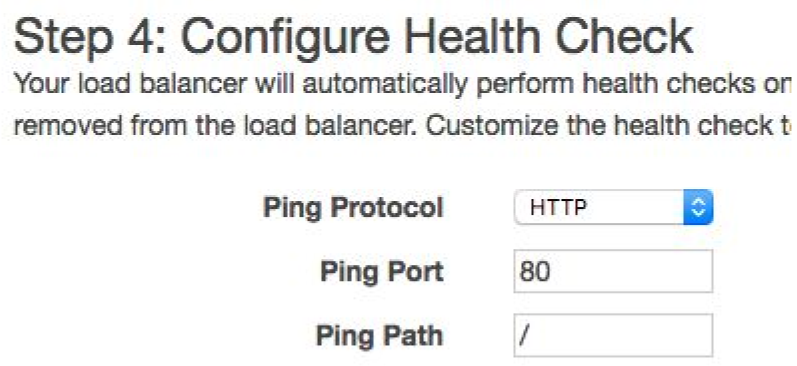
Step 3 consists of configuring the LB to use HTTPS or SSL for security purposes. Although it is highly recommended that you reinforce security in your applications, configuring it is beyond the scope of this lab, therefore, you can simply click on **Next: Configure Health Check**


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

Step 3: Configure Security Settings

Improve your load balancer's security. Your load balancer is not using any secure listener.
If your traffic to the load balancer needs to be secure, use either the HTTPS or the SSL protocol for your front-end connection. You can go back to the first step to add/configure secure listeners under Basic Configuration section. You can also continue with current settings.

In step 4, you need to configure a health check. This is how the LB will evaluate the health of an EC2 instance and decide whether to send requests or avoid a particular instance. The first thing to configure in here is the protocol, port, and path that will be used for the health check. The instances running in the account are serving an application in **port 80**, using the **HTTP** protocol, and using the **root path (/). Y**ou should configure this in the first part of this step



Step 4: Configure Health Check

Your load balancer will automatically perform health checks or
removed from the load balancer. Customize the health check t

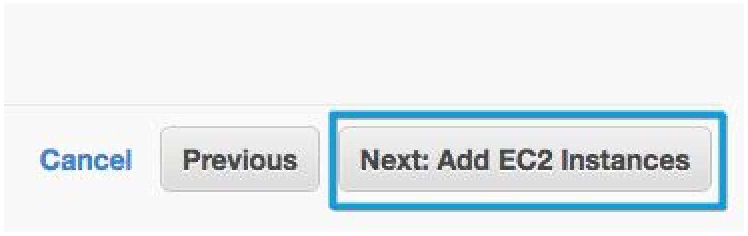
Ping Protocol	HTTP
Ping Port	80
Ping Path	/

There are some **Advanced details** in this step you can configure, but for the purposes of this lab we will stick with the default settings. However, for your reference, this is what they mean:

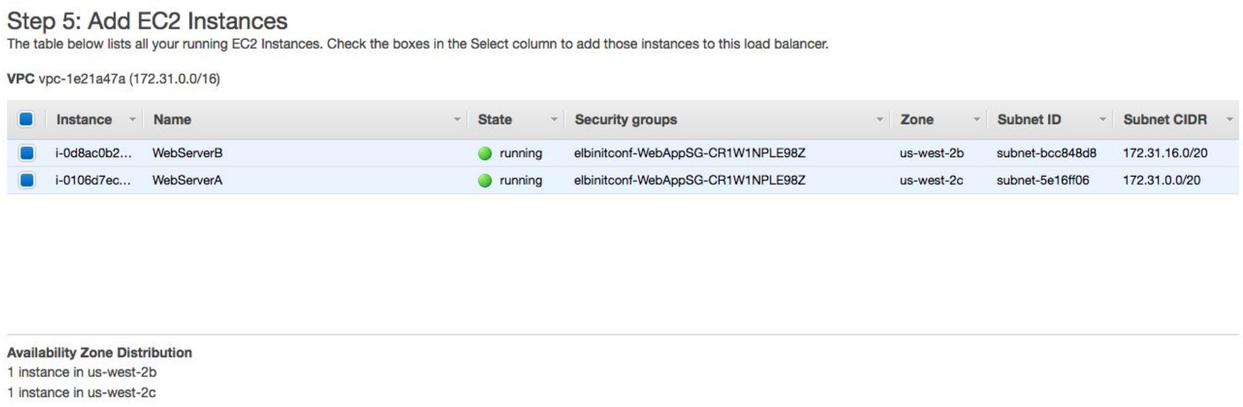
Response Timeout:	The amount of time to wait when receiving a response from the health check, in seconds. Valid values: 2 to 60. Default: 5
HealthCheck Interval:	The amount of time between health checks of an individual instance, in seconds. Valid values: 5 to 300. Default: 30
Unhealthy Threshold:	The number of consecutive failed health checks that must occur before declaring an EC2 instance unhealthy. Valid values: 2 to 10. Default: 2
Healthy Threshold:	The number of consecutive successful health checks that must occur before declaring an EC2 instance healthy. Valid values: 2 to 10. Default: 10

http://docs.aws.amazon.com/elasticloadbalancing/latest/classic/elb-healthchecks.html

Click on **Next: Add EC2 instances** to move on



In step 5, it is time to add EC2 instances to the LB. The first thing to do is to select the instances called **WebServerA** and **WebServerB**:



Step 5: Add EC2 Instances

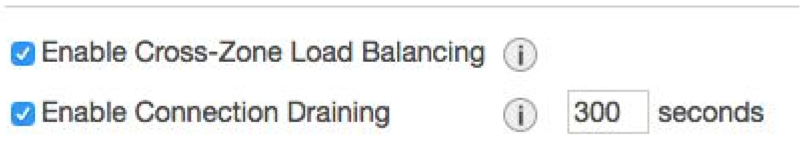
The table below lists all your running EC2 Instances. Check the boxes in the Select column to add those instances to this load balancer.

VPC vpc-1e21a47a (172.31.0.0/16)

☑	Instance	Name	State	Security groups	Zone	Subnet ID	Subnet CIDR
☑	i-0d8ac0b2...	WebServerB	● running	elbinitconf-WebAppSG-CR1W1NPLE98Z	us-west-2b	subnet-bcc848d8	172.31.16.0/20
☑	i-0106d7ec...	WebServerA	● running	elbinitconf-WebAppSG-CR1W1NPLE98Z	us-west-2c	subnet-5e16ff06	172.31.0.0/20

Availability Zone Distribution
1 instance in us-west-2b
1 instance in us-west-2c

There are 2 config points in here as well:

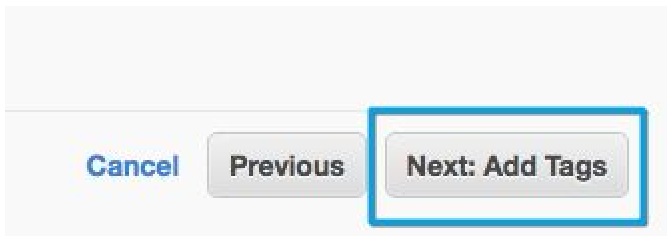

☑ Enable Cross-Zone Load Balancing

☑ Enable Connection Draining 300 seconds

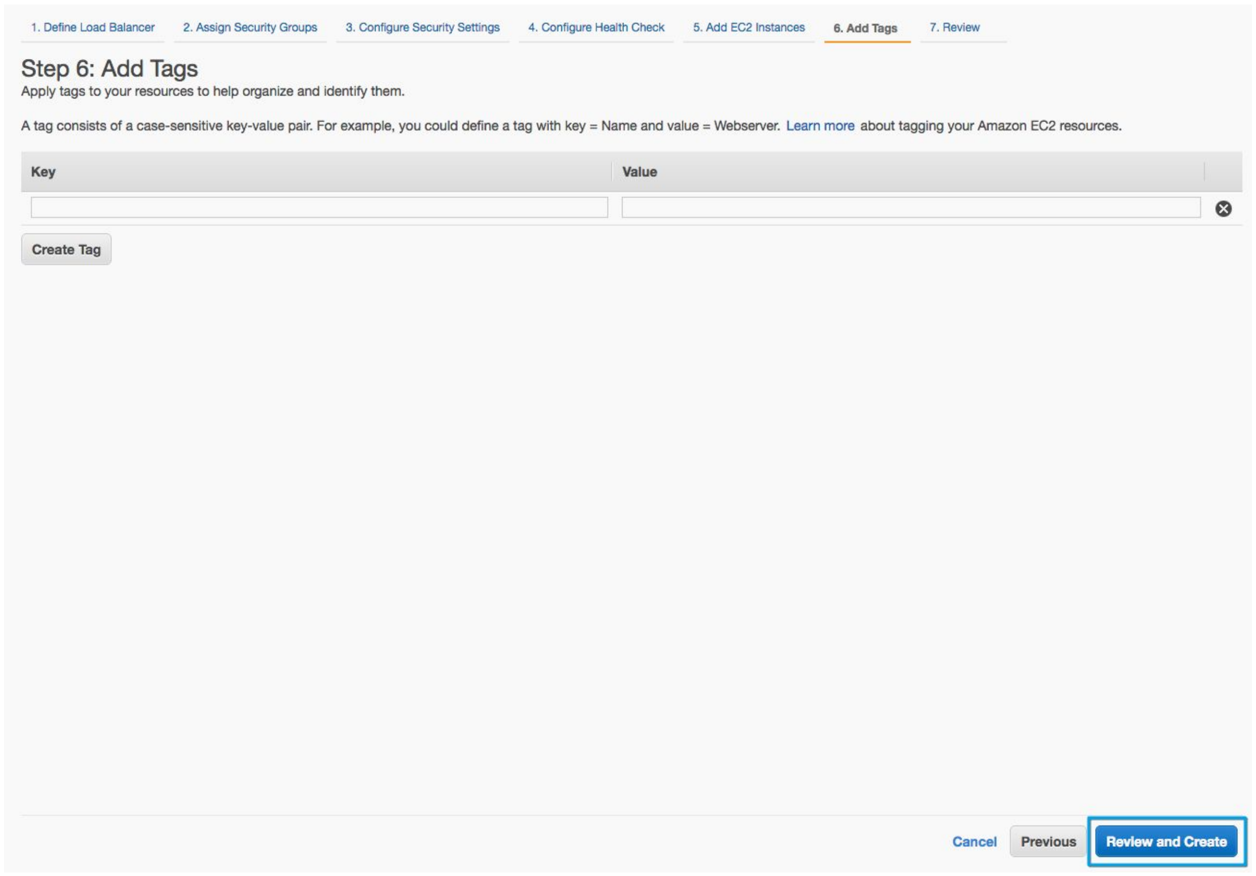
Cross-Zone Load Balancing is used to ensure that your LB distributes incoming requests evenly across all instances in its enabled Availability Zones. That means that the LB will ignore the default of round-robin and will also take into consideration the Availability Zone in which the instance is running.
This reduces the need to maintain equivalent numbers of instances in each enabled Availability Zone, and improves your application's ability to handle the loss of one or more instances.

Connection Draining is used to ensure that a Classic Load Balancer stops sending requests to instances that are de-registering or unhealthy while keeping the existing connections open.

For the purposes of this lab, you can the default settings and click on **Next: Add Tags** to move on.



In Step 6, you have the ability to add tags to the LB. You can either leave it in blank or add as many tags as you want and click on **Review and Create** to move on.


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

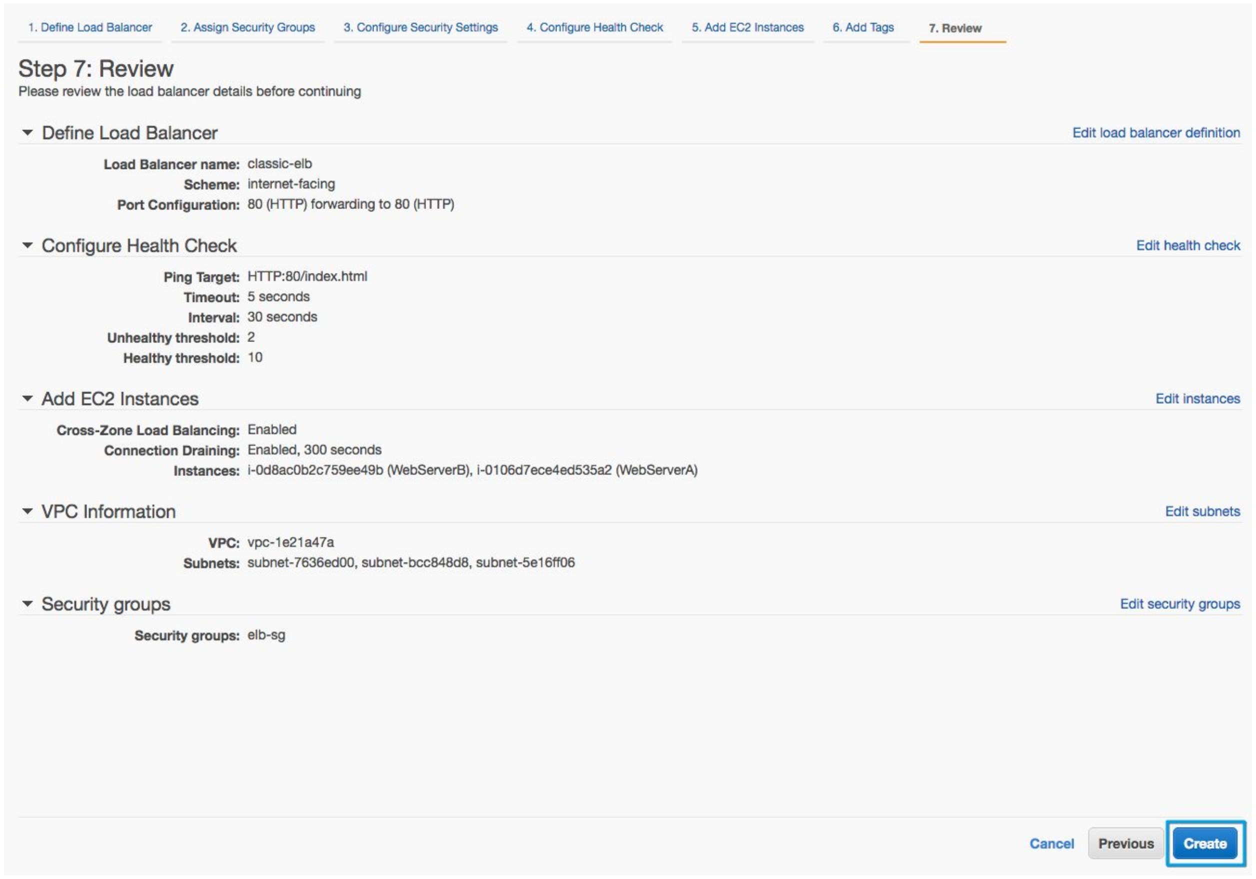
Step 6: Add Tags

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

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. Learn more about tagging your Amazon EC2 resources.

Key	Value

In Step 7, it is time to review the LB's settings, double check the config points and click on **Create:**


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

Step 7: Review

Please review the load balancer details before continuing

Define Load Balancer — Edit load balancer definition

Load Balancer name:	classic-elb
Scheme:	internet-facing
Port Configuration:	80 (HTTP) forwarding to 80 (HTTP)

Configure Health Check — Edit health check

Ping Target:	HTTP:80/index.html
Timeout:	5 seconds
Interval:	30 seconds
Unhealthy threshold:	2
Healthy threshold:	10

Add EC2 Instances — Edit instances

Cross-Zone Load Balancing:	Enabled
Connection Draining:	Enabled, 300 seconds
Instances:	i-0d8ac0b2c759ee49b (WebServerB), i-0106d7ece4ed535a2 (WebServerA)

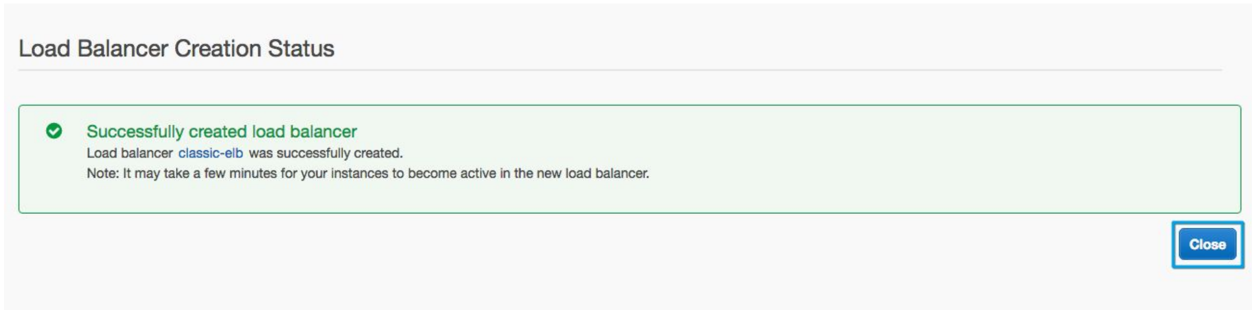
VPC Information — Edit subnets

VPC:	vpc-1e21a47a
Subnets:	subnet-7636ed00, subnet-bcc848d8, subnet-5e16ff06

Security groups — Edit security groups

Security groups:	elb-sg

If you see a Success message it means that you LB is created, you can click on **Close** in the AWS console, and move on the the next lab step



Load Balancer Creation Status

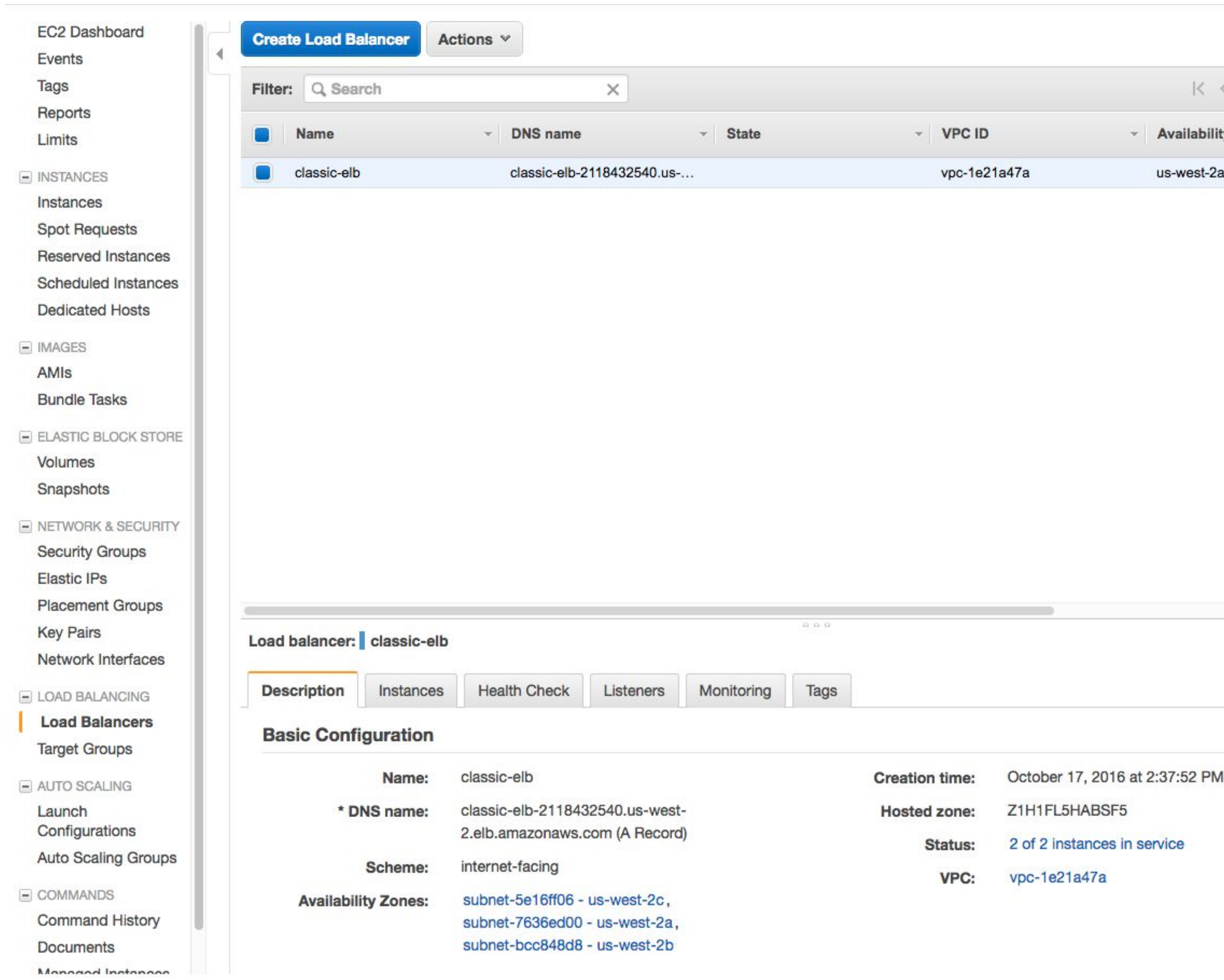
Successfully created load balancer

Load balancer classic-elb was successfully created.

Note: It may take a few minutes for your instances to become active in the new load balancer.

Step 4 Configuring security groups for ELB

Now that you completed the creation of you first Classic Load Balancer, you should be seeing this screen:



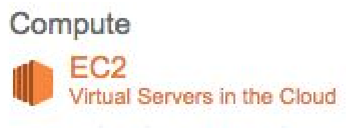
Name	DNS name	State	VPC ID	Availabilit
classic-elb	classic-elb-2118432540.us-...		vpc-1e21a47a	us-west-2a

Load balancer: classic-elb

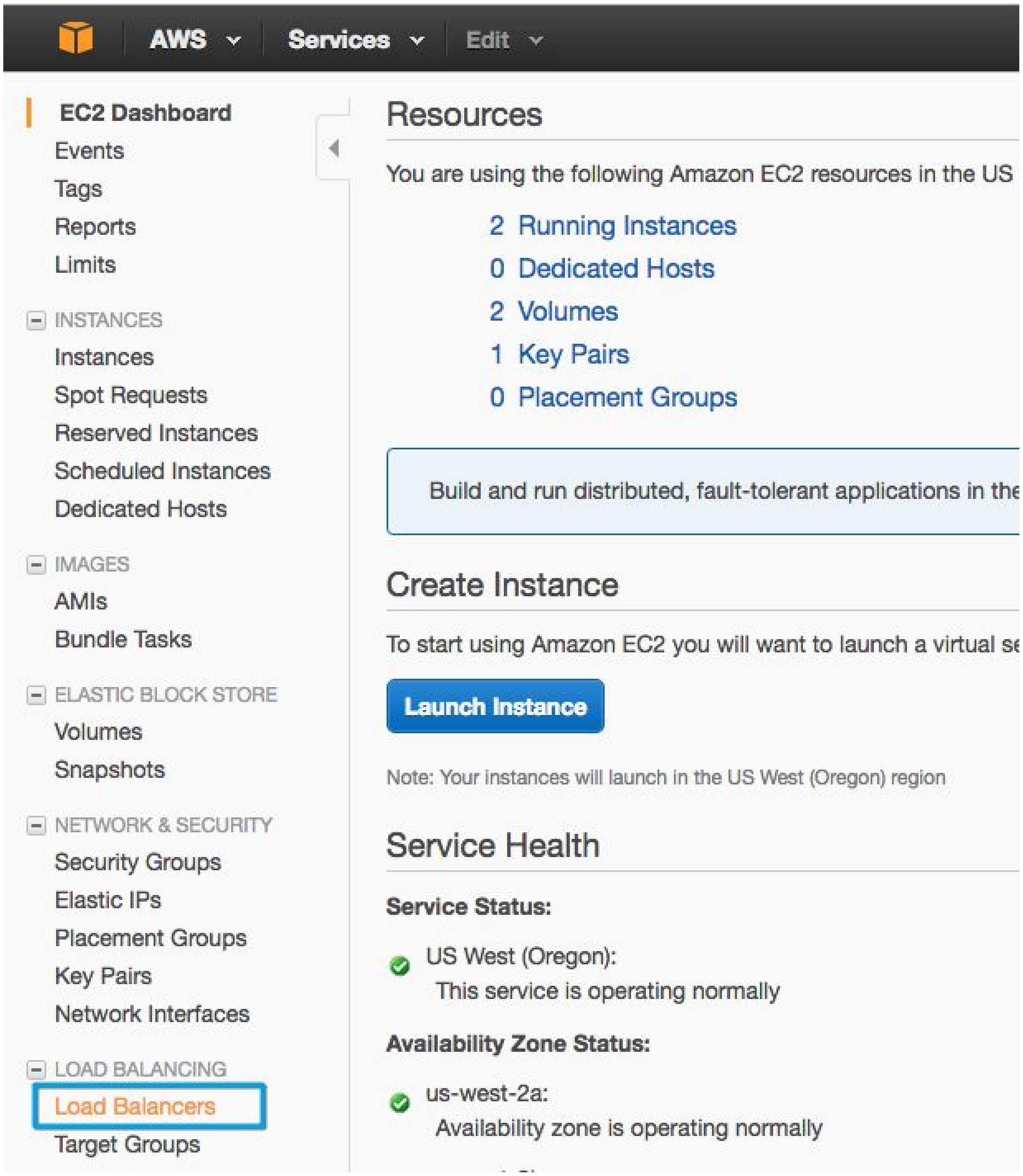
Basic Configuration

Name:	classic-elb	Creation time:	October 17, 2016 at 2:37:52 PM
* DNS name:	classic-elb-2118432540.us-west-2.elb.amazonaws.com (A Record)	Hosted zone:	Z1H1FL5HABSF5
Scheme:	internet-facing	Status:	2 of 2 instances in service
Availability Zones:	subnet-5e16ff06 - us-west-2c, subnet-7636ed00 - us-west-2a, subnet-bcc848d8 - us-west-2b	VPC:	vpc-1e21a47a

If not, go back to the EC2 console:



And click on Load Balancers:



Resources

You are using the following Amazon EC2 resources in the US

2 Running Instances
0 Dedicated Hosts
2 Volumes
1 Key Pairs
0 Placement Groups

Build and run distributed, fault-tolerant applications in the

Create Instance

To start using Amazon EC2 you will want to launch a virtual se

Note: Your instances will launch in the US West (Oregon) region

Service Health

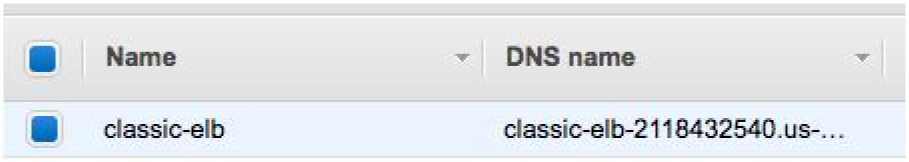
Service Status:

US West (Oregon):
This service is operating normally

Availability Zone Status:

us-west-2a:
Availability zone is operating normally

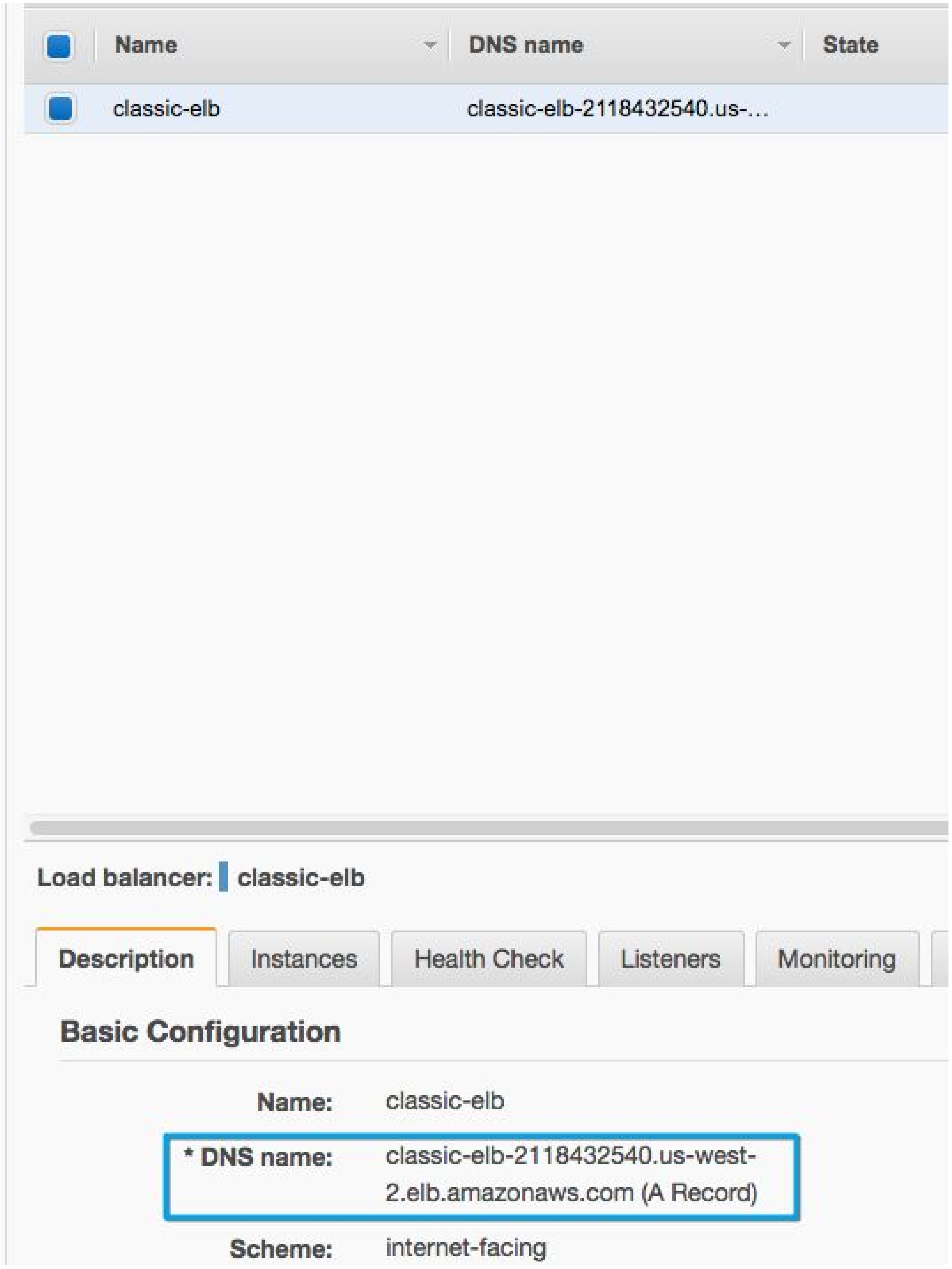
Then select the LB you just created:



Name	DNS name
classic-elb	classic-elb-2118432540.us-...

In this step, we will configure the Security Group (SG) associated with the EC2 instances running in the account to **accept only connections coming from the LB**.

But before that, let's take a look at what we have just created. With the LB selected copy the **DNS Name** of the LB



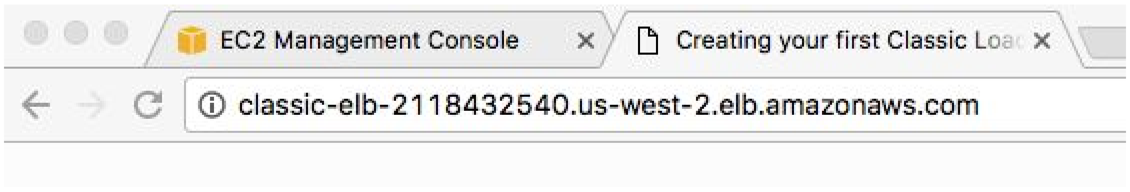
Basic Configuration

Name: classic-elb

* DNS name: classic-elb-2118432540.us-west-2.elb.amazonaws.com (A Record)

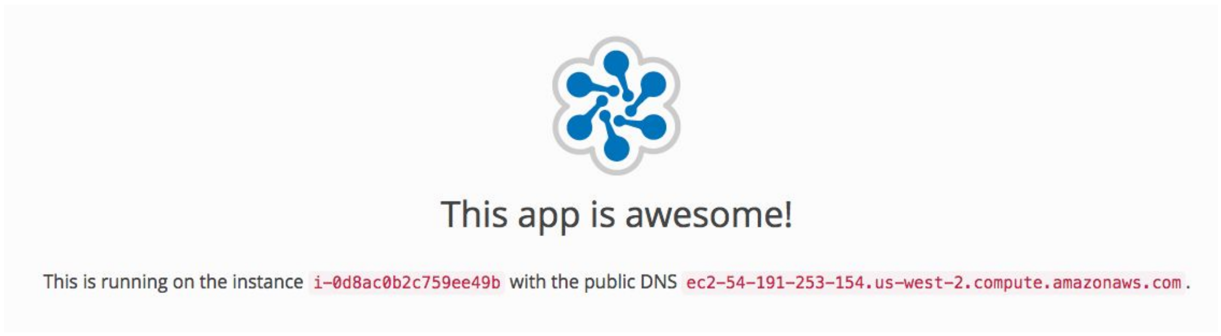
Scheme: internet-facing

Paste this address into a new tab in your browser (don't include the "A Record" information in parentheses):



classic-elb-2118432540.us-west-2.elb.amazonaws.com

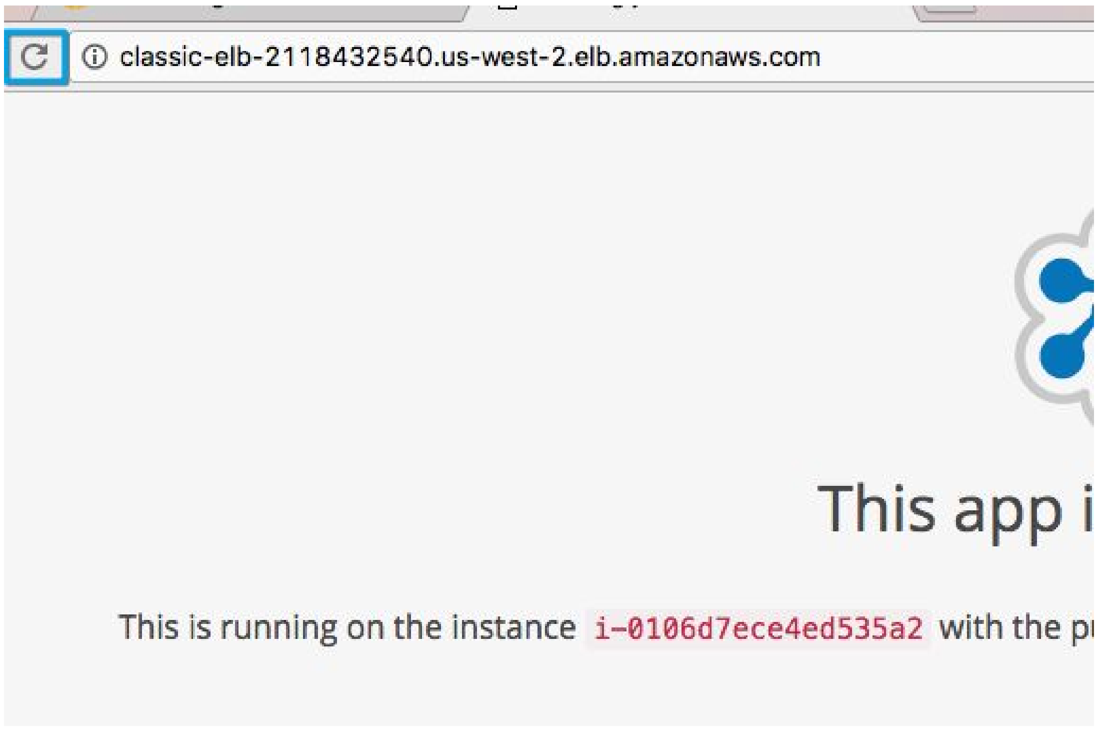
You should be able to see something like this



This app is awesome!

This is running on the instance `i-0d8ac0b2c759ee49b` with the public DNS `ec2-54-191-253-154.us-west-2.compute.amazonaws.com`.

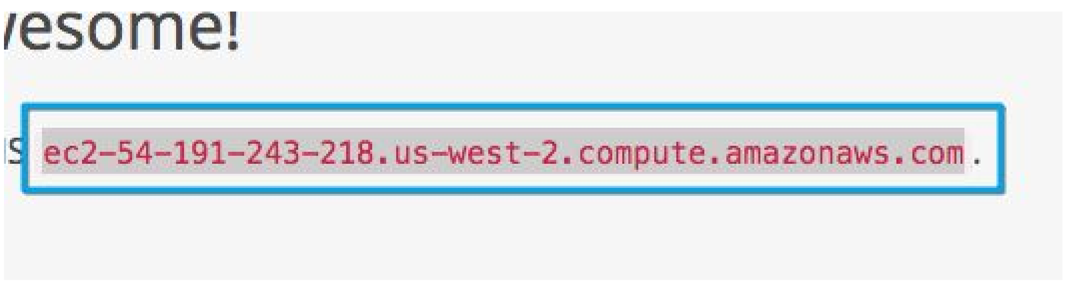
This page is dynamically generated and will show the instance ID of the EC2 instance running the app and its public DNS address. You can click on refresh page a couple of times to see this value changing. It means that both instances are in service and the ELB service is forwarding the traffic to a different instance each time you hit refresh.


classic-elb-2118432540.us-west-2.elb.amazonaws.com

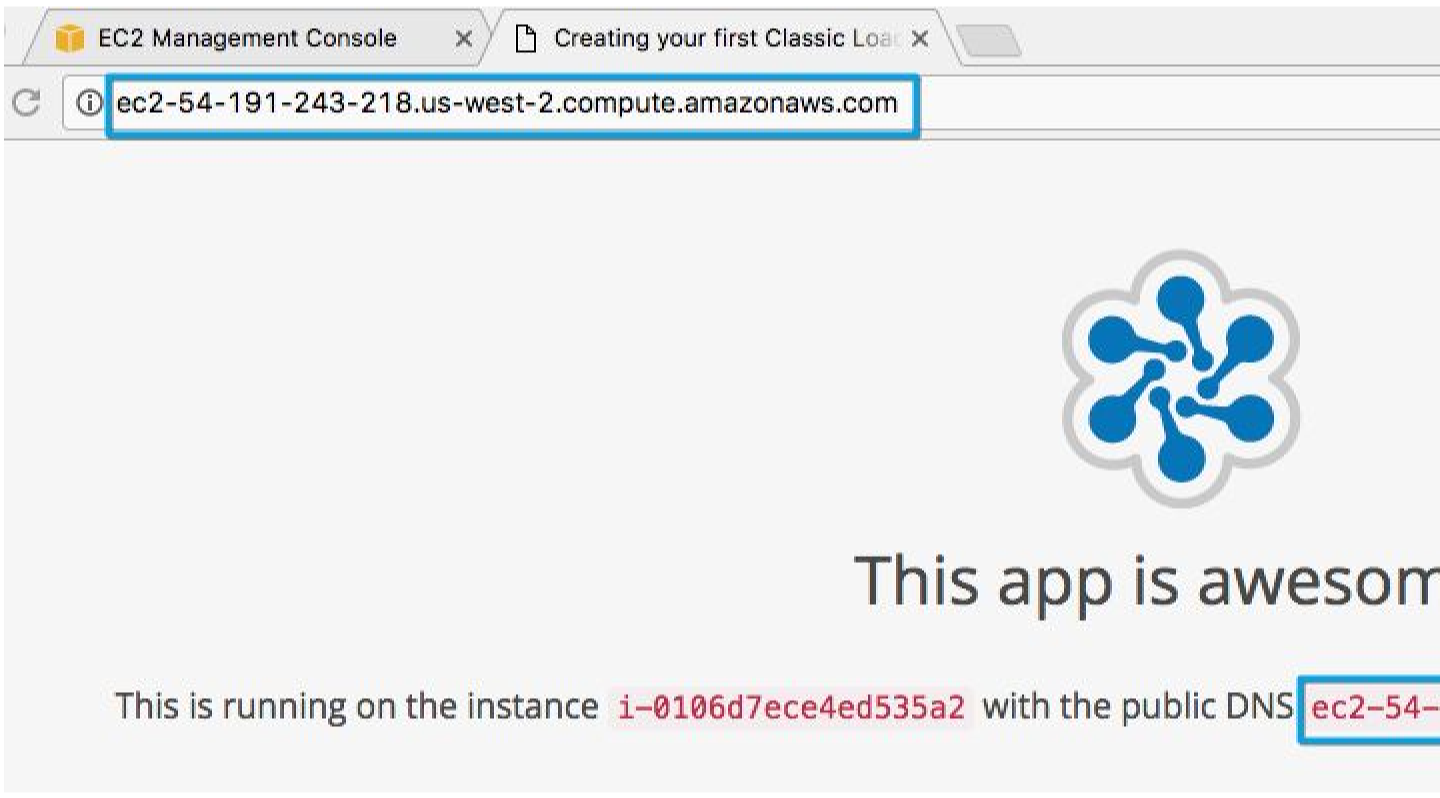
This app i

This is running on the instance `i-0106d7ece4ed535a2` with the pı

This proves that the LB is working as expected. However if you take the public DNS address of the instance, and try to access it directly, you should be able to load the same page.

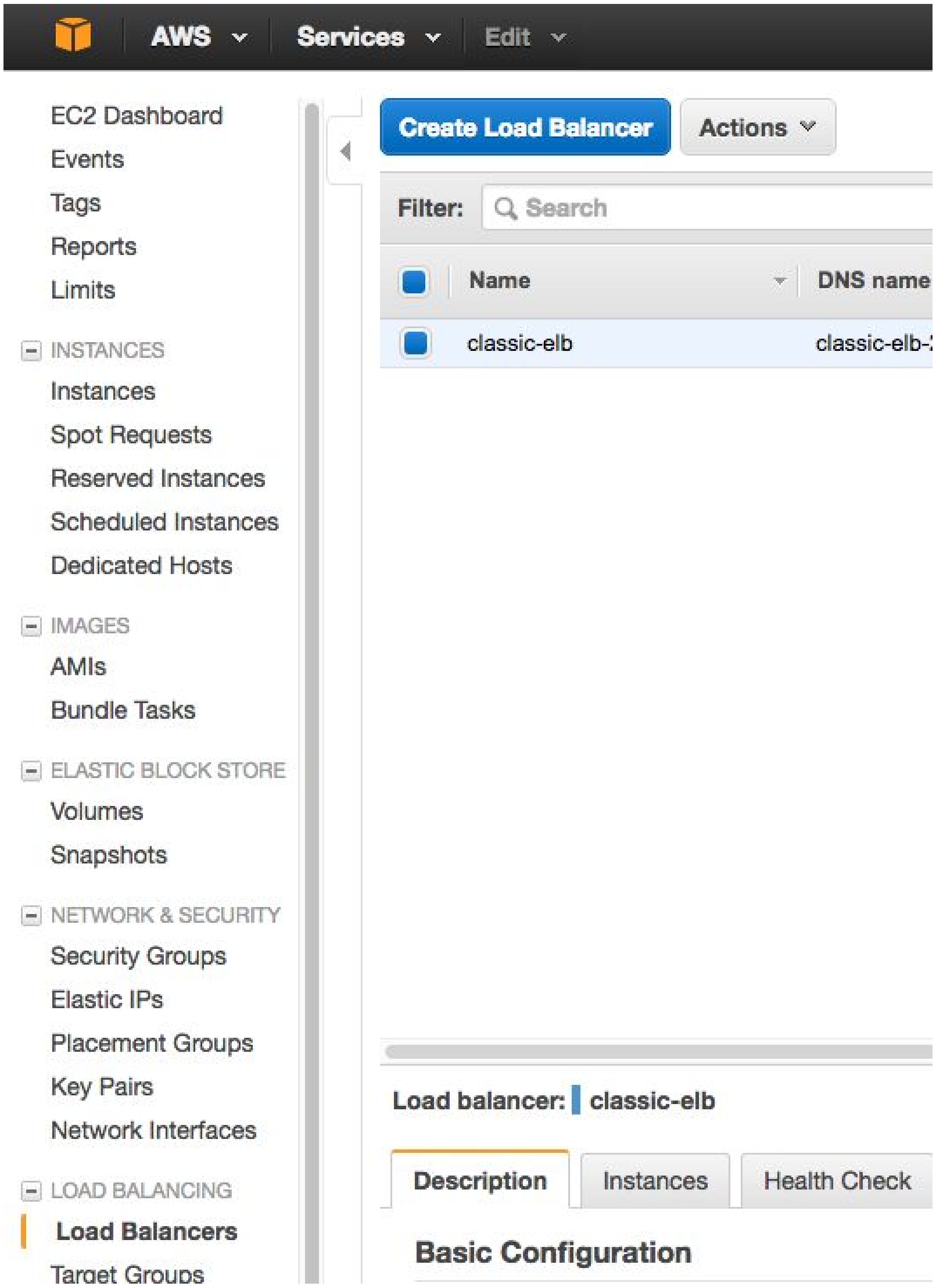

ıesome!

ıS `ec2-54-191-243-218.us-west-2.compute.amazonaws.com` .

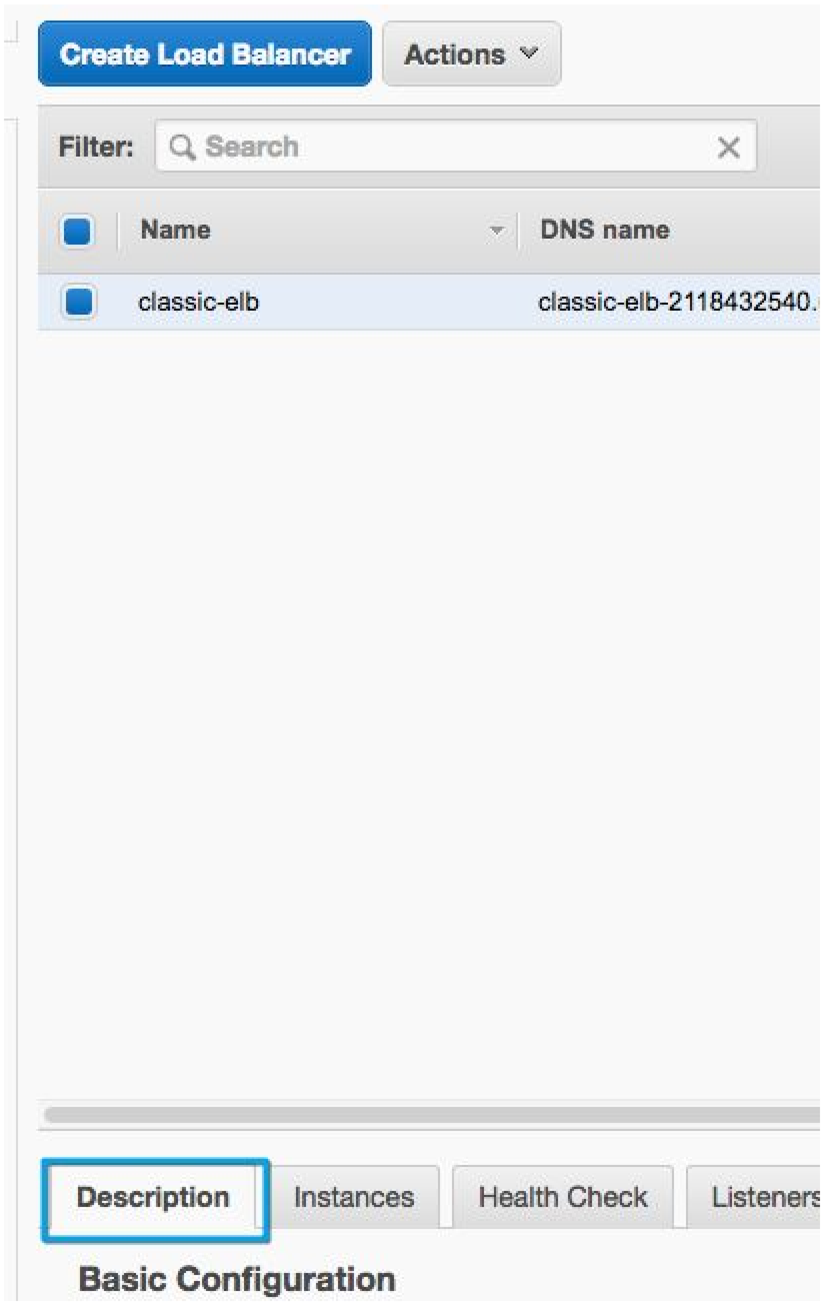

EC2 Management Console
Creating your first Classic Loa
ec2-54-191-243-218.us-west-2.compute.amazonaws.com
This app is awesom

This is running on the instance i-0106d7ece4ed535a2 with the public DNS ec2-54-

Now you are accessing a particular instance directly, and we want to avoid that. To do so we need to change a few things. This is happening because the SG associated with the EC2 instances is allowing access from anywhere to the port 80; we want to change it in a way that the instances will only allow traffic coming from the LB we just created. To configure this go back to the **EC2 management console**on the **Load Balancers** page:

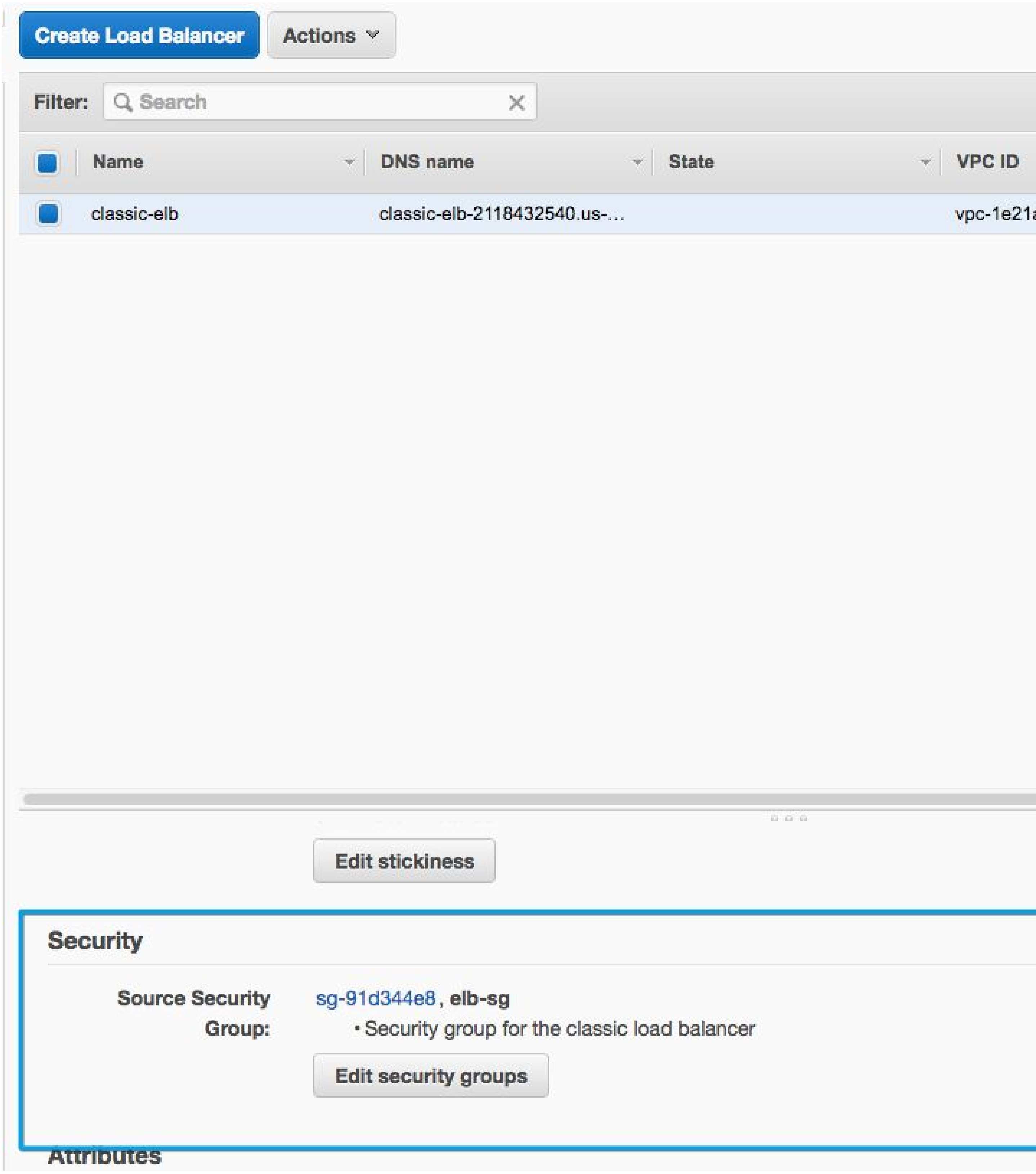


And with the LB selected, go to the **Description** tab:



Name	DNS name
classic-elb	classic-elb-2118432540.

And scroll down until you see **Security**:


Create Load Balancer
Actions

Filter: Search

	Name	DNS name	State	VPC ID
✓	classic-elb	classic-elb-2118432540.us-…		vpc-1e21a

Edit stickiness

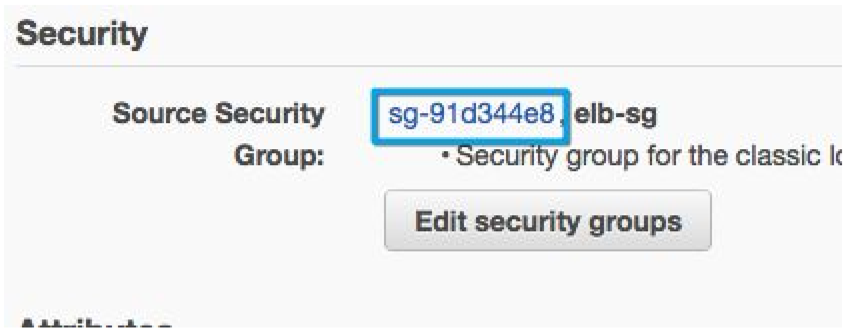
Security

Source Security Group: sg-91d344e8, elb-sg
• Security group for the classic load balancer

Edit security groups

Attributes

Copy the unique identifier of the LB's Security Group:



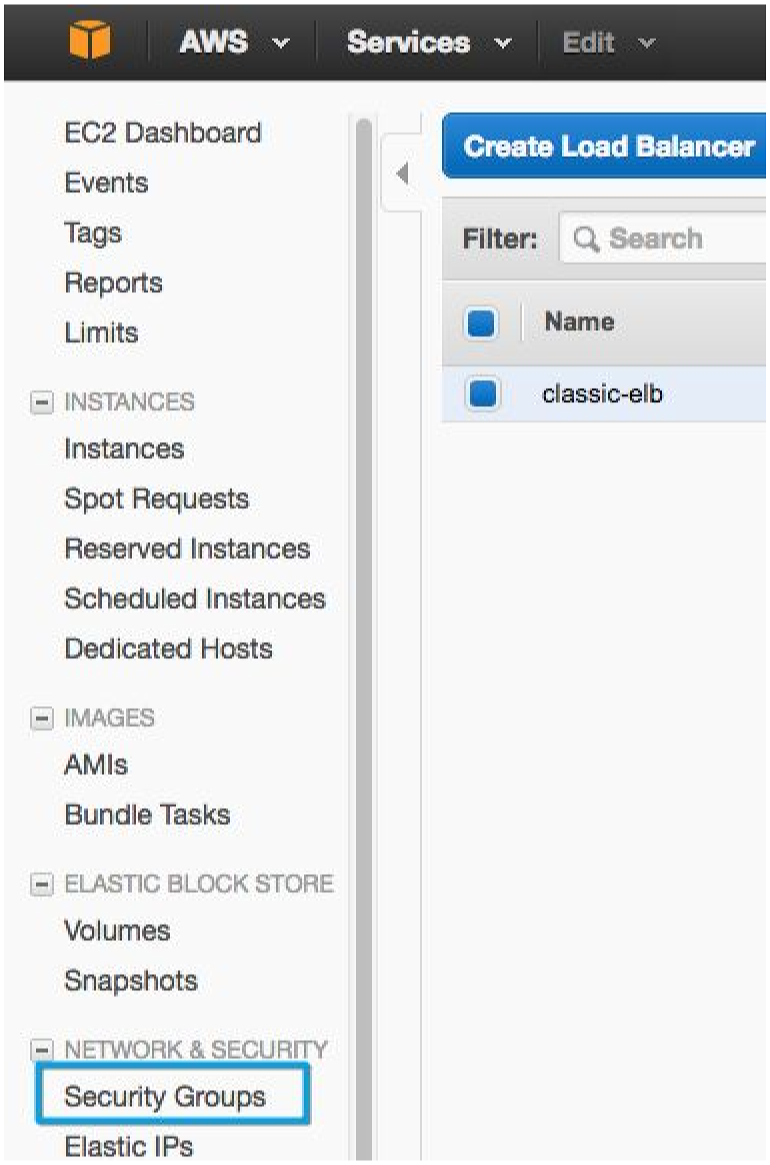
Security

Source Security Group: sg-91d344e8, elb-sg

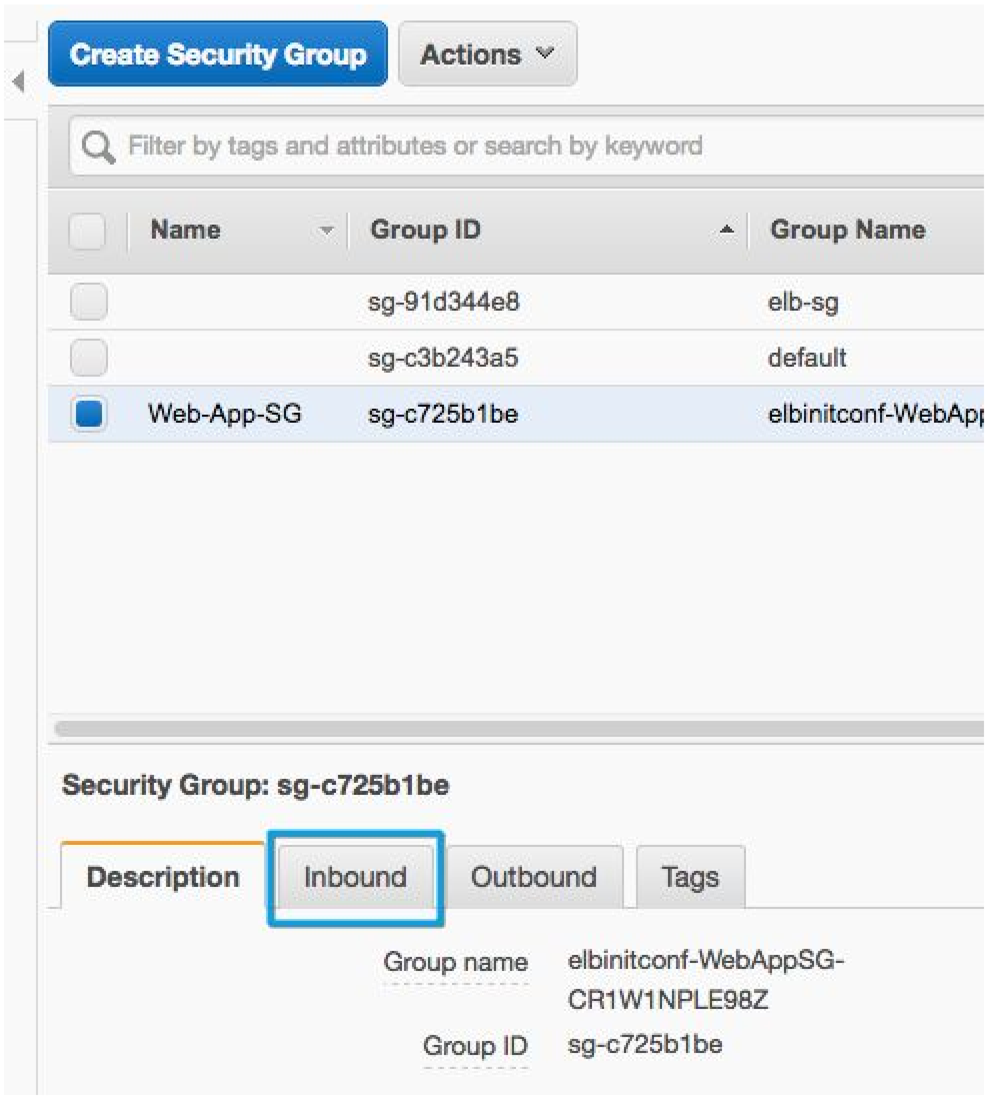
• Security group for the classic l

We will use this information in just a moment.

Now click on **Security Groups:**



Select the SG called `Web-App-SG` and click on the **Inbound** tab:

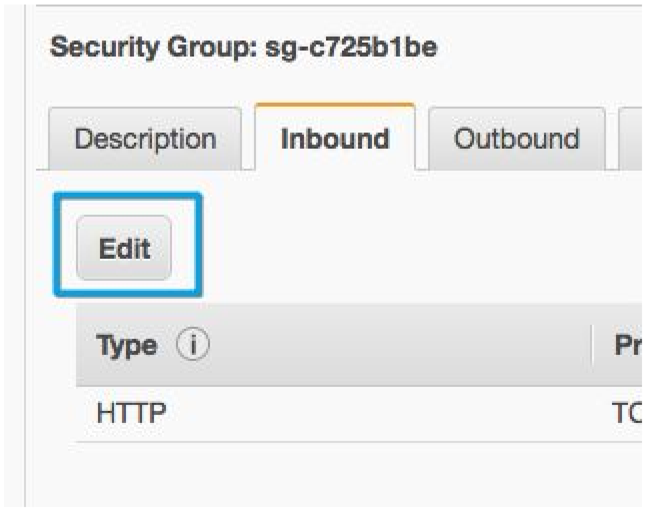


	Name	Group ID	Group Name
		sg-91d344e8	elb-sg
		sg-c3b243a5	default
✓	Web-App-SG	sg-c725b1be	elbinitconf-WebApp

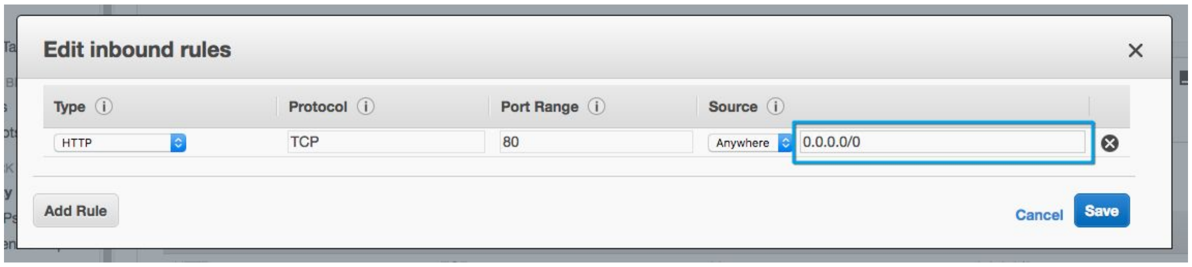
Security Group: sg-c725b1be

Group name	elbinitconf-WebAppSG-CR1W1NPLE98Z
Group ID	sg-c725b1be

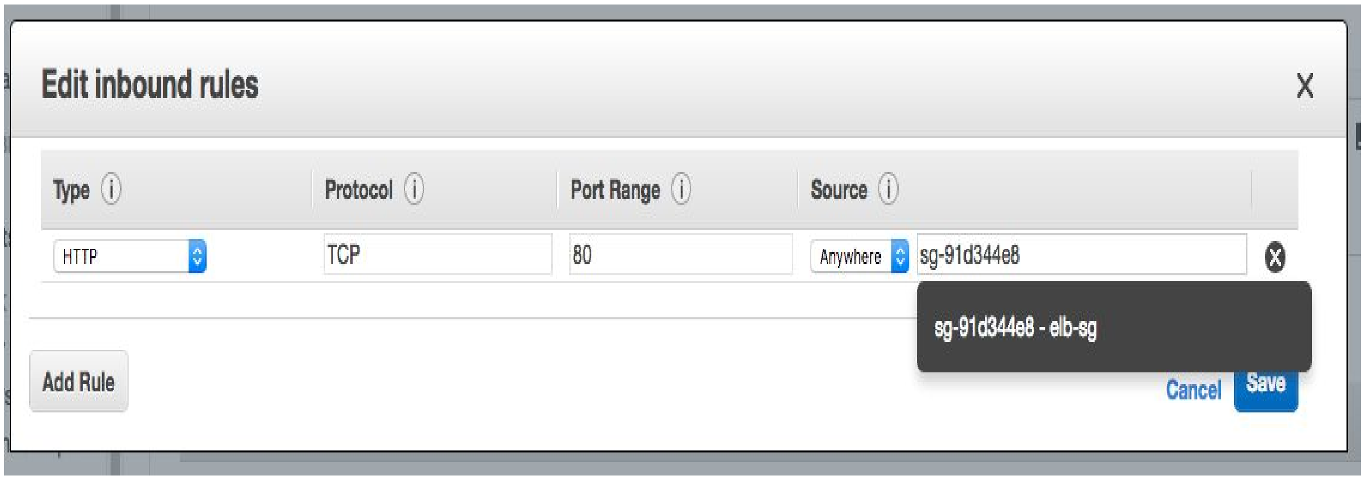
Now, click on **Edit** to change the current rules associated with this SG:



We want to allow only connections coming from the LB to the instances, however, the **LB doesn't have a particular IP address** associated with it so we can't specify an IP address in here. Instead, we will restrict the access by using the SG we just created for the LB. We will change the current rule to deny access to anywhere and allow it only to members of the LB's security group. The process is very straight forward, simply replace the source in the HTTP rule that is already created with the LB's security group identifier that you just copied:


Edit inbound rules

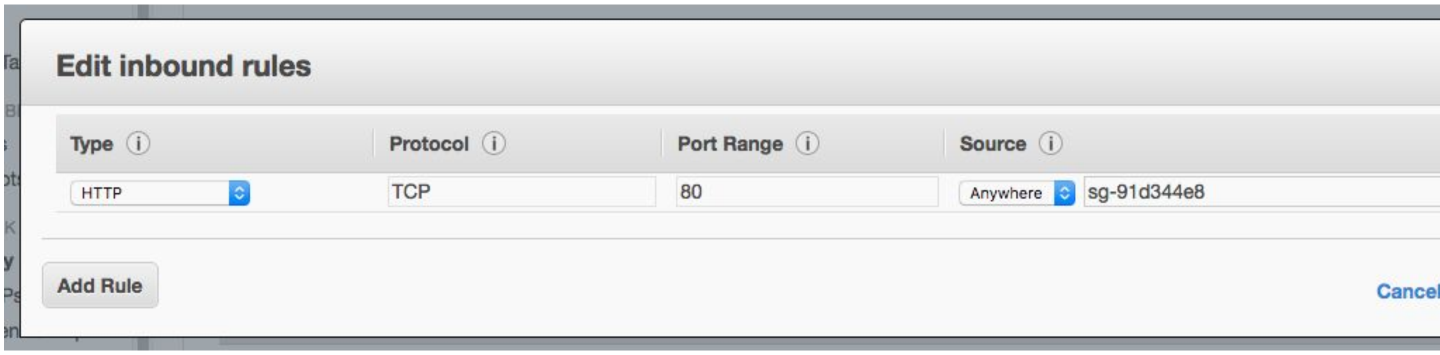
Type	Protocol	Port Range	Source	
HTTP	TCP	80	Anywhere	0.0.0.0/0



Type	Protocol	Port Range	Source
HTTP	TCP	80	Anywhere sg-91d344e8

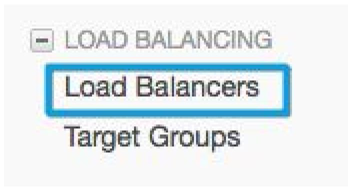
sg-91d344e8 - elb-sg

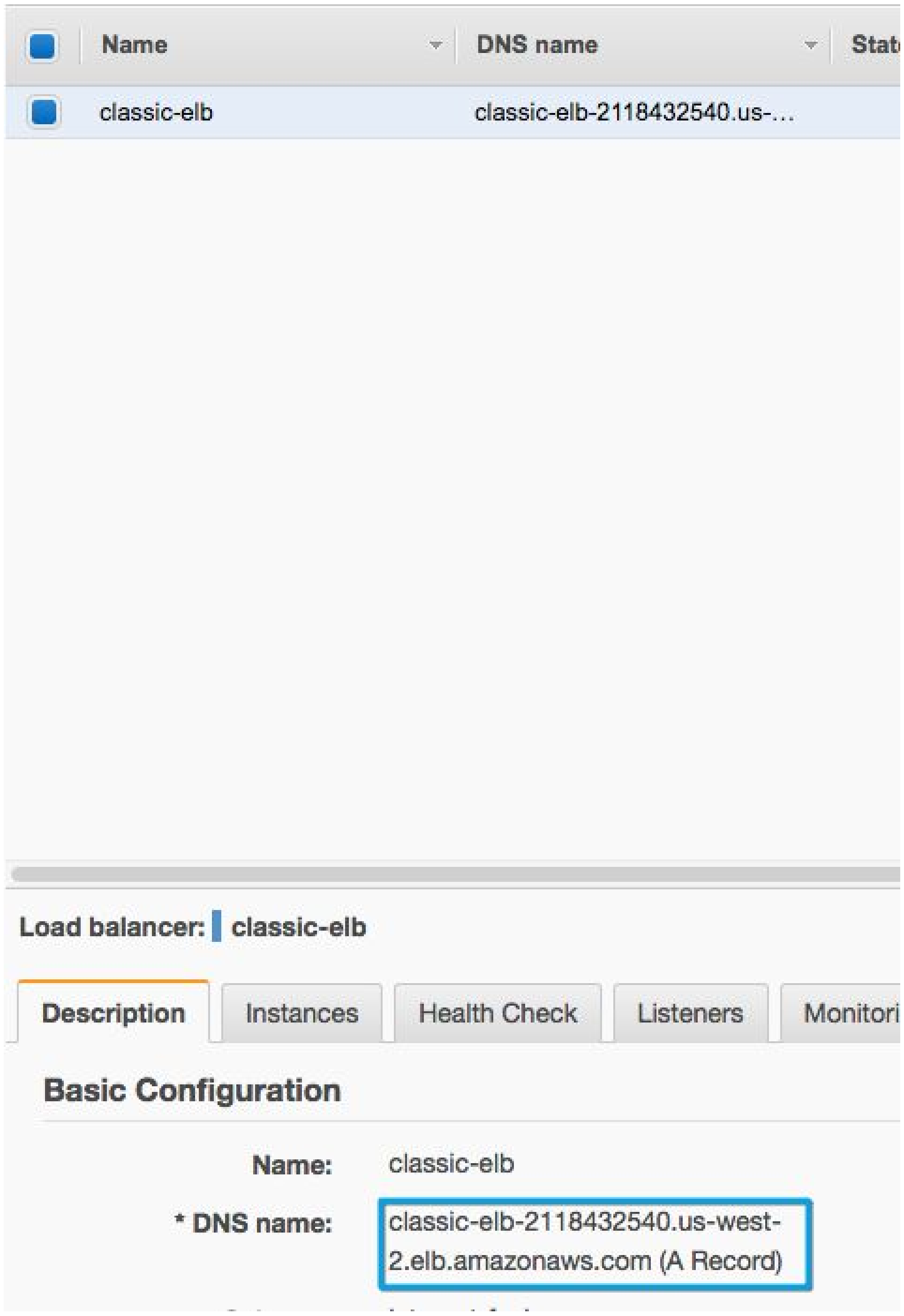
You can also start typing "sg-" and select the correct SG identifier in the list that will appear. Then click **Save**.



Type	Protocol	Port Range	Source
HTTP	TCP	80	Anywhere sg-91d344e8

Now let's test the rule. Click on **Load Balancers** to be able to copy the URL for the LB again:





Name	DNS name	Stat
classic-elb	classic-elb-2118432540.us-…	

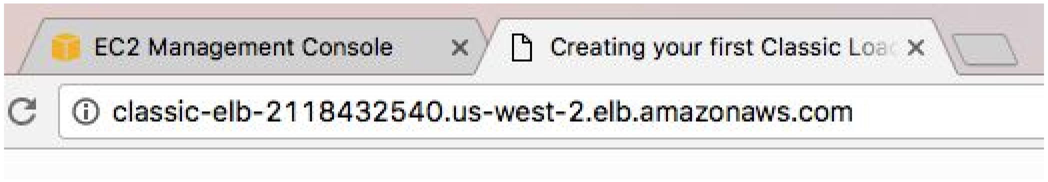
Load balancer: classic-elb

Basic Configuration

Name: classic-elb

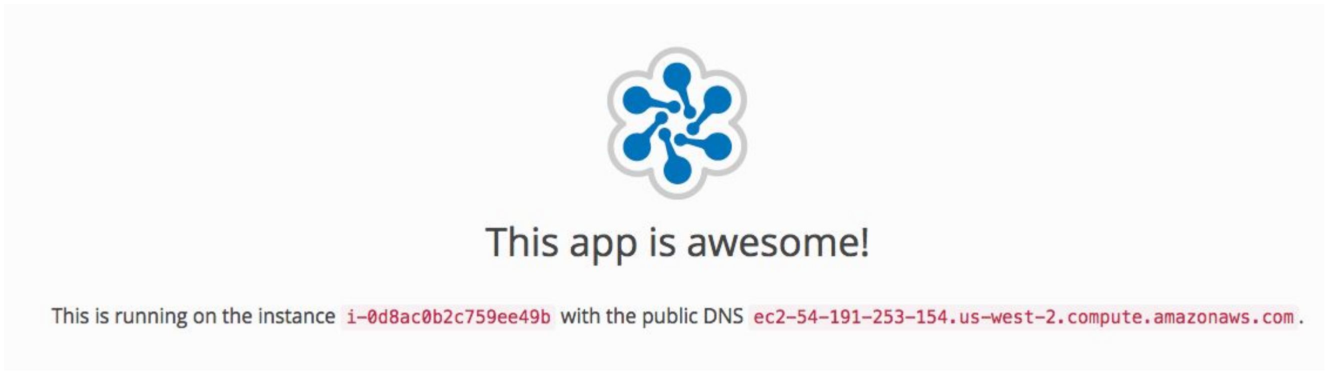
* DNS name: classic-elb-2118432540.us-west-2.elb.amazonaws.com (A Record)

Copy the **DNS Name** address and past it into a new tab in your browser:



classic-elb-2118432540.us-west-2.elb.amazonaws.com

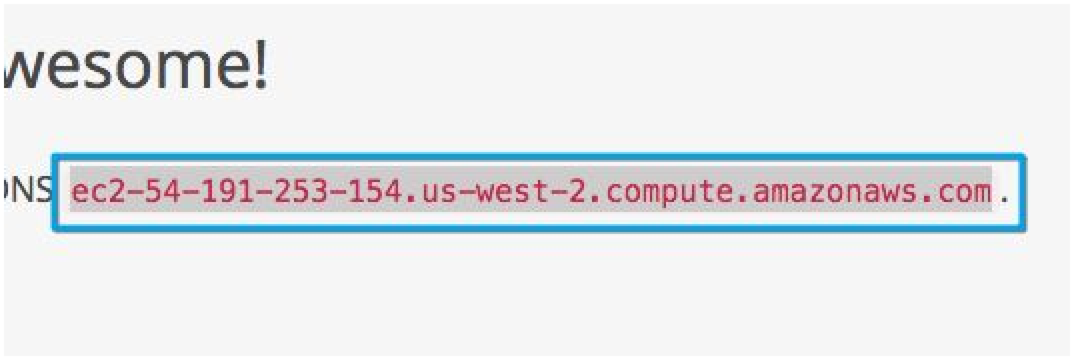
And check your results:



This app is awesome!

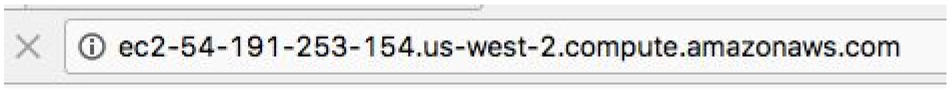
This is running on the instance `i-0d8ac0b2c759ee49b` with the public DNS `ec2-54-191-253-154.us-west-2.compute.amazonaws.com`.

Nothing new here. However, if you try to copy and past the public DNS address of the instance you are accessing, and paste it into your browser, you shouldn't be able to access the app - the connection will timeout.

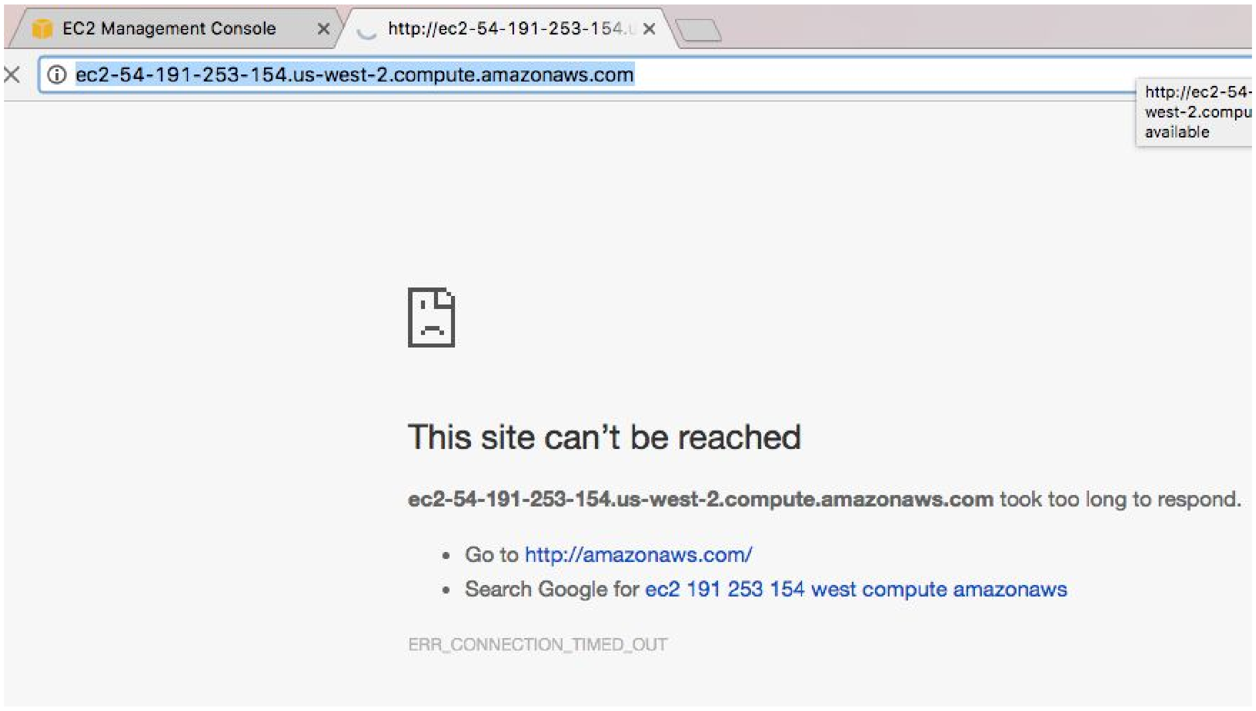


wesome!

NS `ec2-54-191-253-154.us-west-2.compute.amazonaws.com`.



ec2-54-191-253-154.us-west-2.compute.amazonaws.com


EC2 Management Console
http://ec2-54-191-253-154.
ec2-54-191-253-154.us-west-2.compute.amazonaws.com
http://ec2-54-
west-2.compu
available

This site can't be reached

ec2-54-191-253-154.us-west-2.compute.amazonaws.com took too long to respond.

- Go to http://amazonaws.com/
- Search Google for ec2 191 253 154 west compute amazonaws

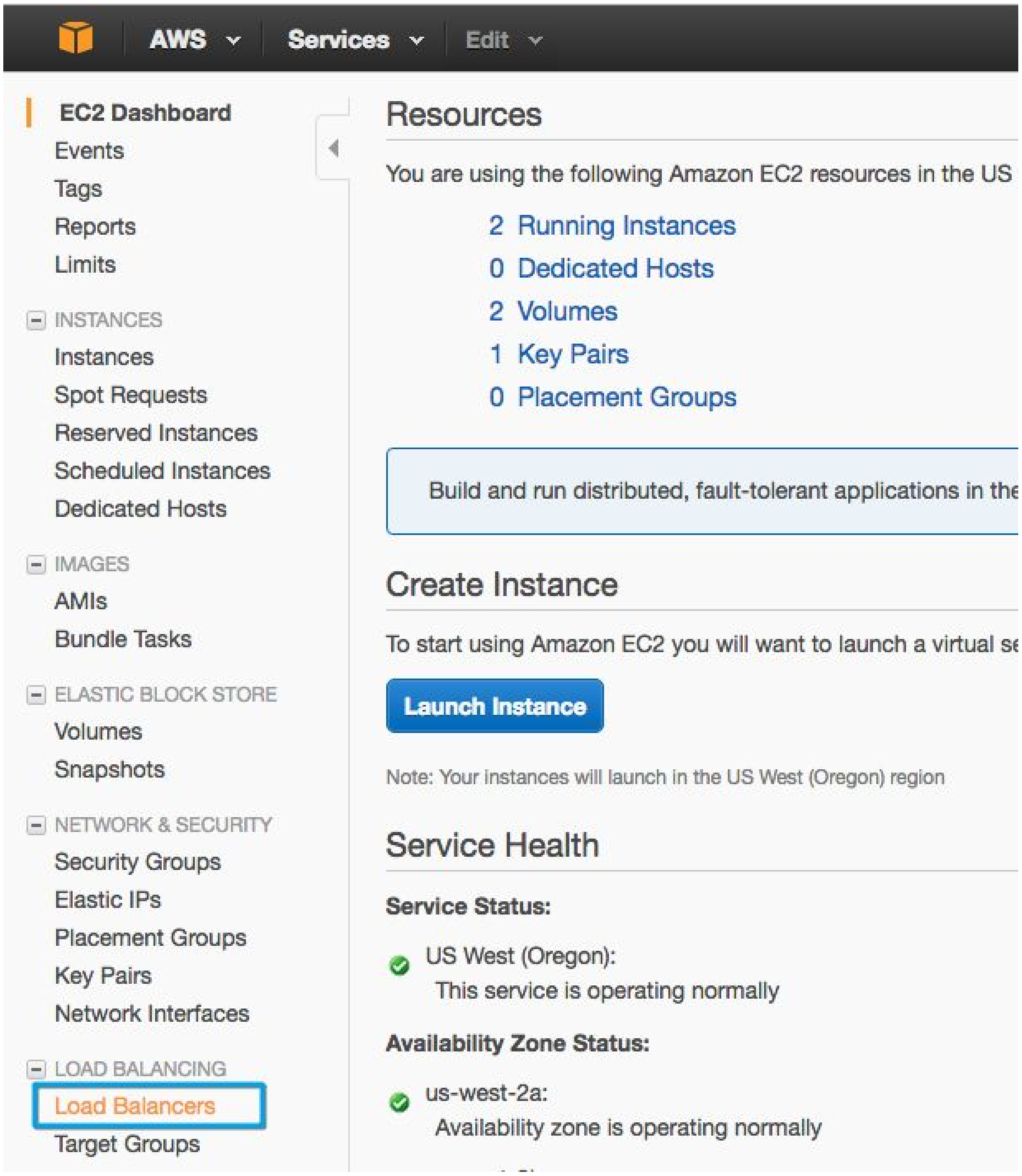
ERR_CONNECTION_TIMED_OUT

Now you can move ahead

tep 5 Checking a load balancer's behavior during instance failures

In this step, we will make an EC2 instance fail and see how the ELB service responds to that. To do so, go to the EC2 console:

And click on Load Balancers:



Resources

You are using the following Amazon EC2 resources in the US

2 Running Instances
0 Dedicated Hosts
2 Volumes
1 Key Pairs
0 Placement Groups

Build and run distributed, fault-tolerant applications in the

Create Instance

To start using Amazon EC2 you will want to launch a virtual se

Note: Your instances will launch in the US West (Oregon) region

Service Health

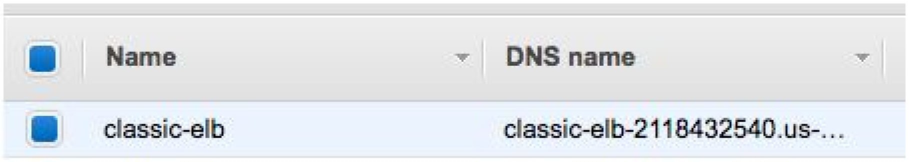
Service Status:

US West (Oregon):
This service is operating normally

Availability Zone Status:

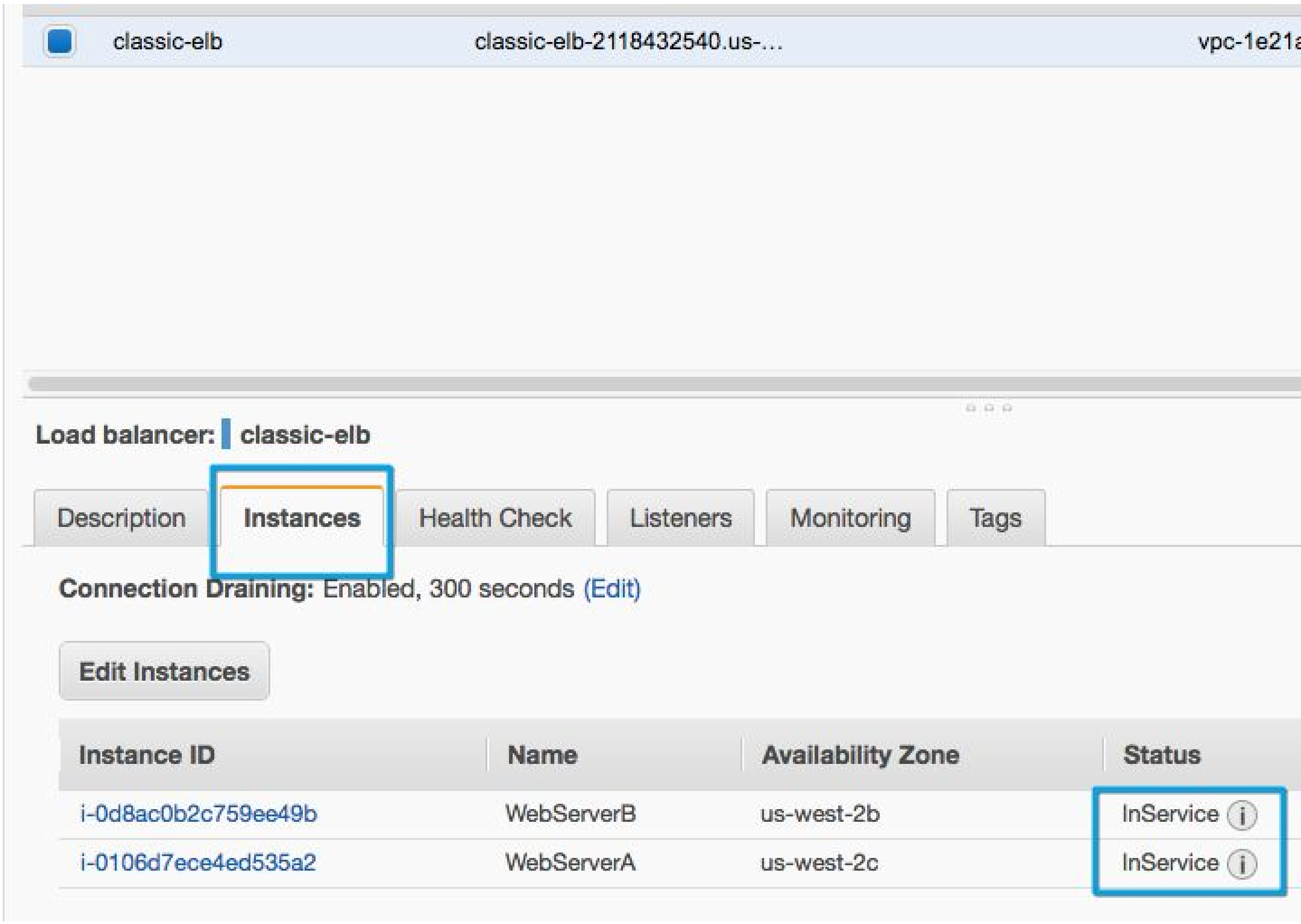
us-west-2a:
Availability zone is operating normally

Then select the LB you just created:



Name	DNS name
classic-elb	classic-elb-2118432540.us-…

If you click on **Instances** you will be able to see that both instances are currently on service:



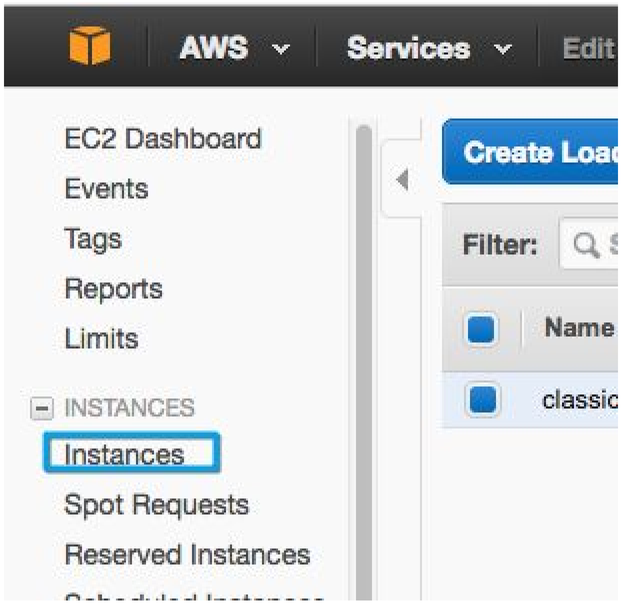
classic-elb classic-elb-2118432540.us-… vpc-1e21a

Load balancer: classic-elb

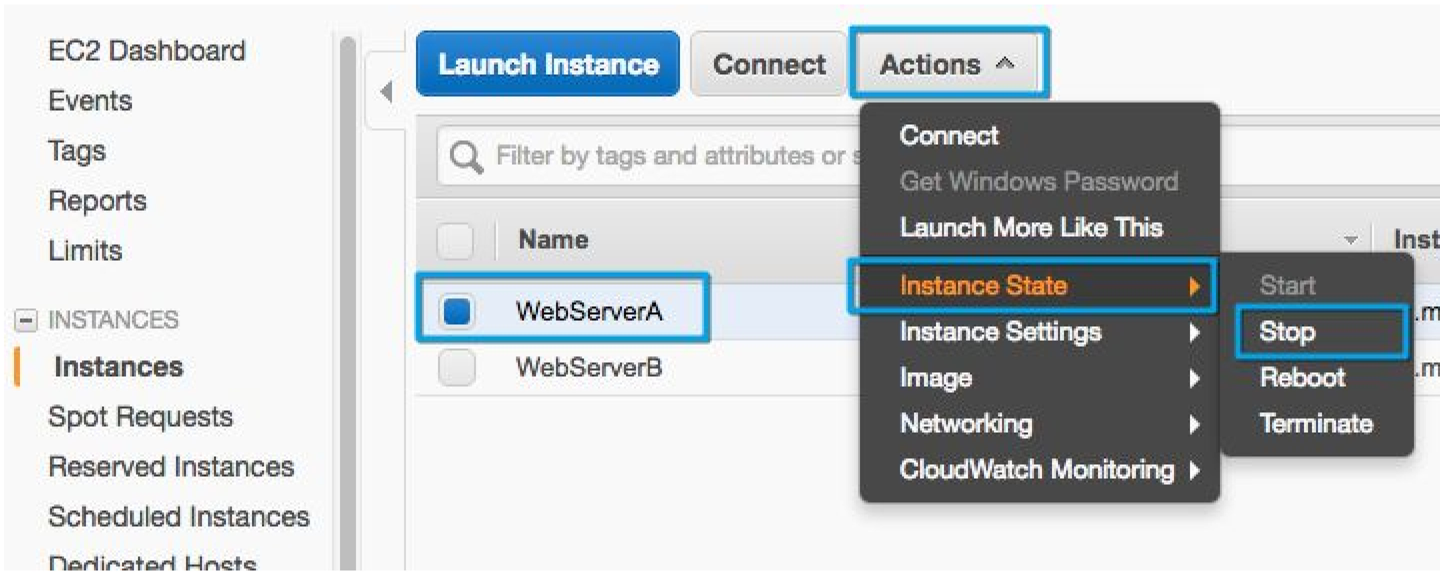
Connection Draining: Enabled, 300 seconds (Edit)

Instance ID	Name	Availability Zone	Status
i-0d8ac0b2c759ee49b	WebServerB	us-west-2b	InService
i-0106d7ece4ed535a2	WebServerA	us-west-2c	InService

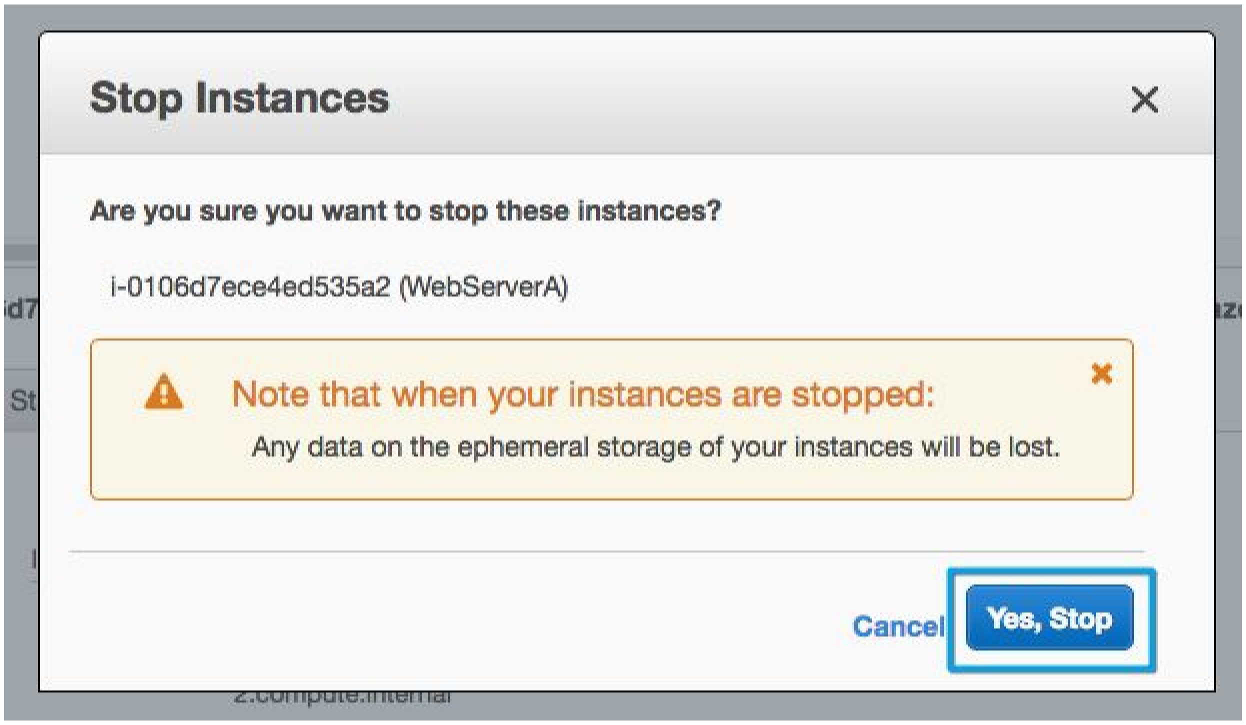
This means that the instances are answering the health checks we configured during the LB's creation, and thus the ELB service has decided to put them **InService.** To see what happens with the LB when there is a problem in a particular instance, let's simulate an instance failure. The easiest way to do this is to stop a running instance, so let's do that. Click on **Instances**:



And choose any of the running instances, click on **Actions, Instance State,** then on **Stop**:



Then click on **Yes, Stop** on the dialog box that pops up:


Stop Instances

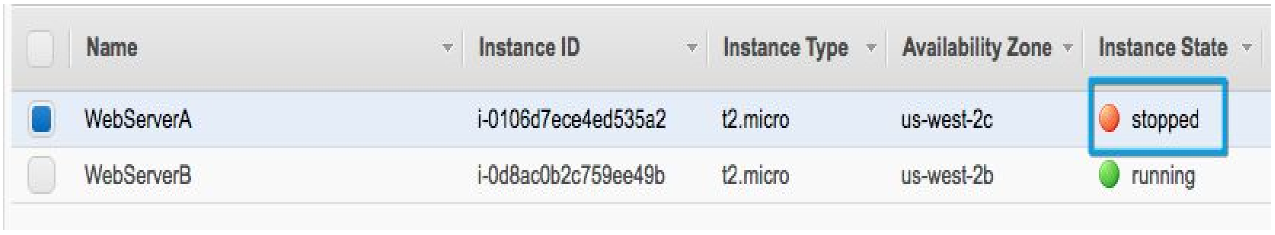
Are you sure you want to stop these instances?

i-0106d7ece4ed535a2 (WebServerA)

Note that when your instances are stopped:
Any data on the ephemeral storage of your instances will be lost.

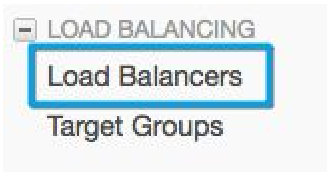
Doing this will stop a particular instance, which will make it fail the ELB's health checks.

Once the **Instance State** changes to **stopped** you can proceed:

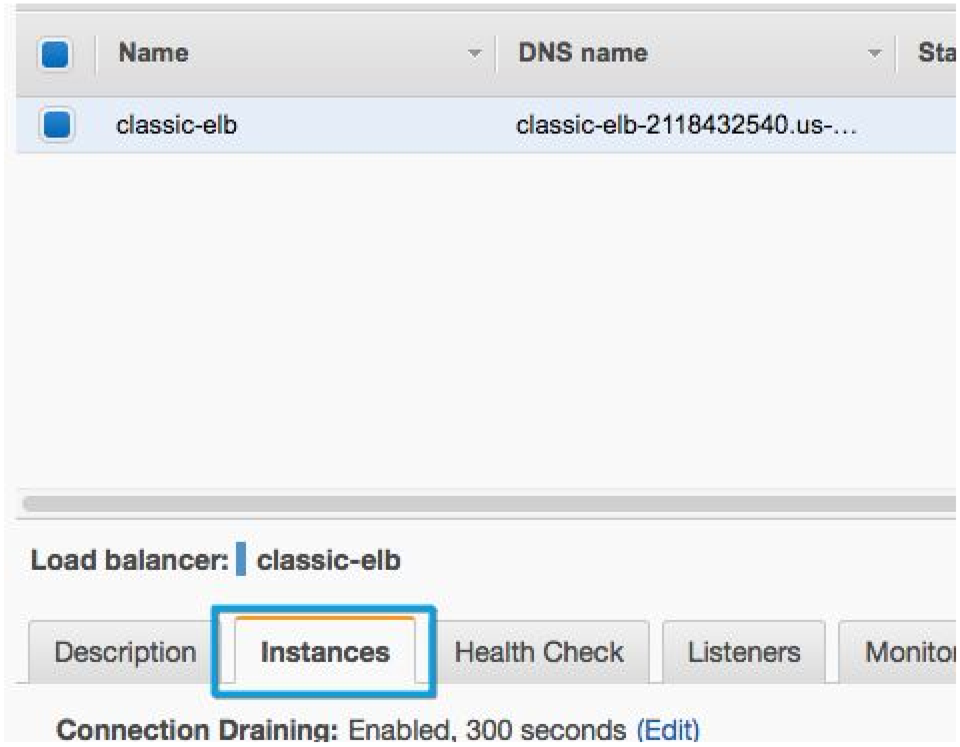


Name	Instance ID	Instance Type	Availability Zone	Instance State
WebServerA	i-0106d7ece4ed535a2	t2.micro	us-west-2c	🟥 stopped
WebServerB	i-0d8ac0b2c759ee49b	t2.micro	us-west-2b	🟩 running

Now click on **Load Balancers**:



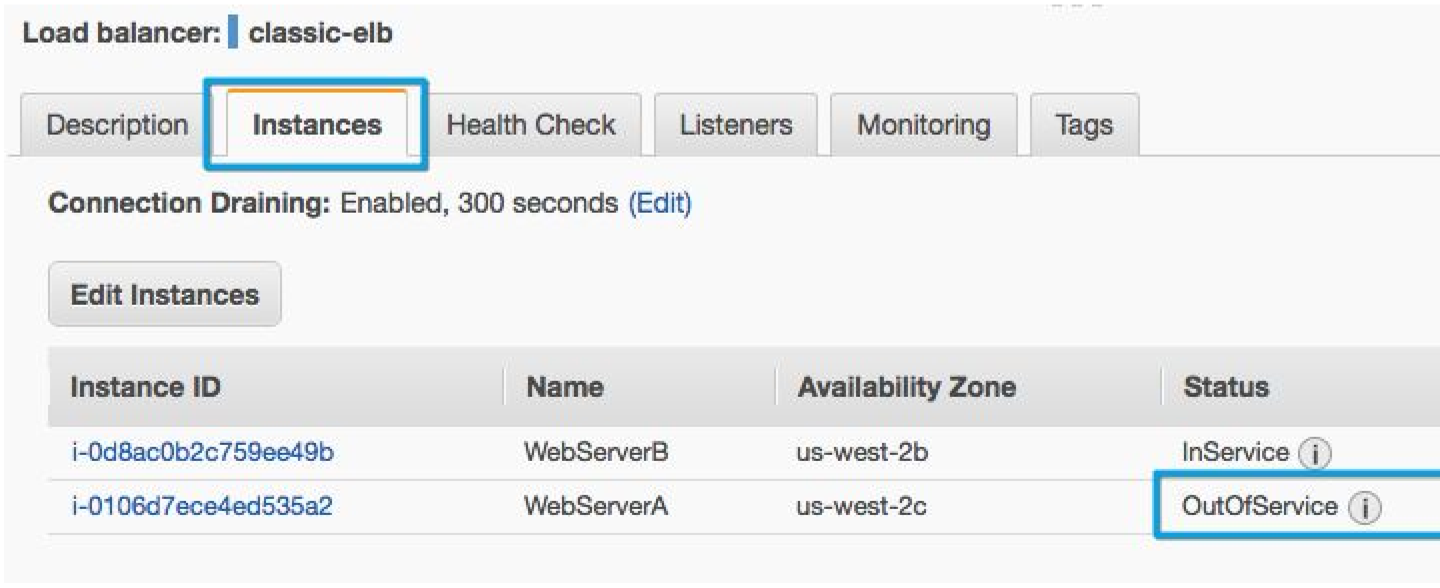
Select the LB and click on **Instances**:



Name	DNS name
classic-elb	classic-elb-2118432540.us-…

Connection Draining: Enabled, 300 seconds (Edit)

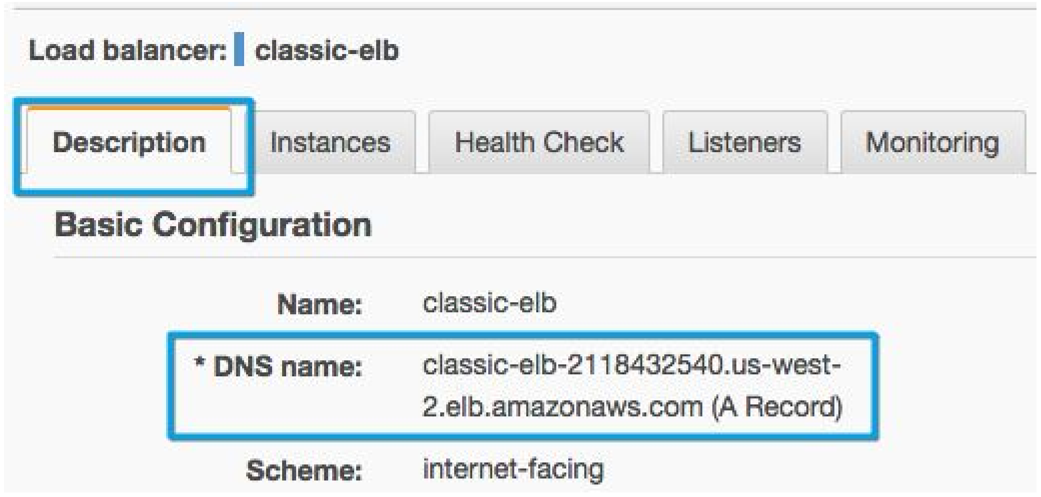
Now you should be able to see an instance with the status **OutOfService**



Connection Draining: Enabled, 300 seconds (Edit)

Instance ID	Name	Availability Zone	Status
i-0d8ac0b2c759ee49b	WebServerB	us-west-2b	InService
i-0106d7ece4ed535a2	WebServerA	us-west-2c	OutOfService

This means that there is only one instance serving the application, and therefore all the requests will be forwarded to the same instance. You can test this behavior by clicking on **Description** and accessing the **DNS name** of the LB:

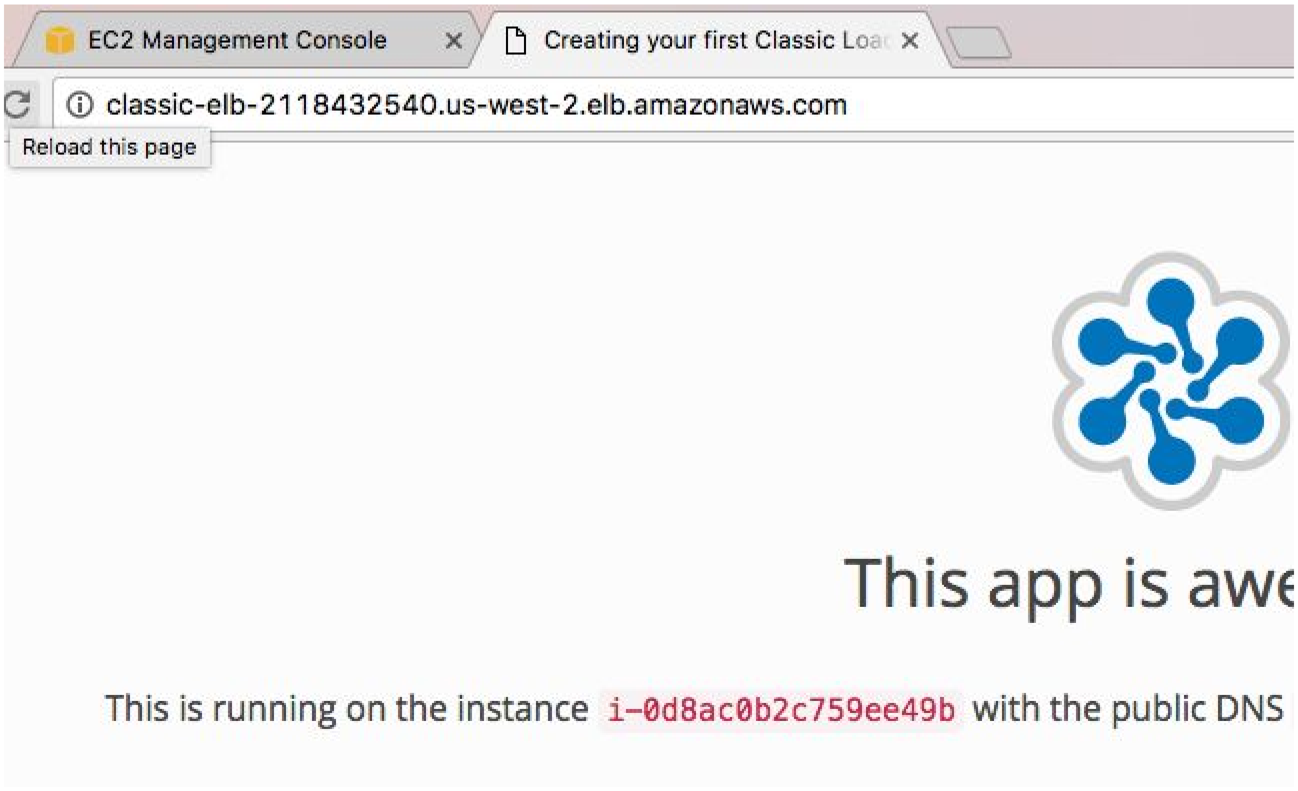

Load balancer: classic-elb

Basic Configuration

Name: classic-elb

* DNS name: classic-elb-2118432540.us-west-2.elb.amazonaws.com (A Record)

Scheme: internet-facing

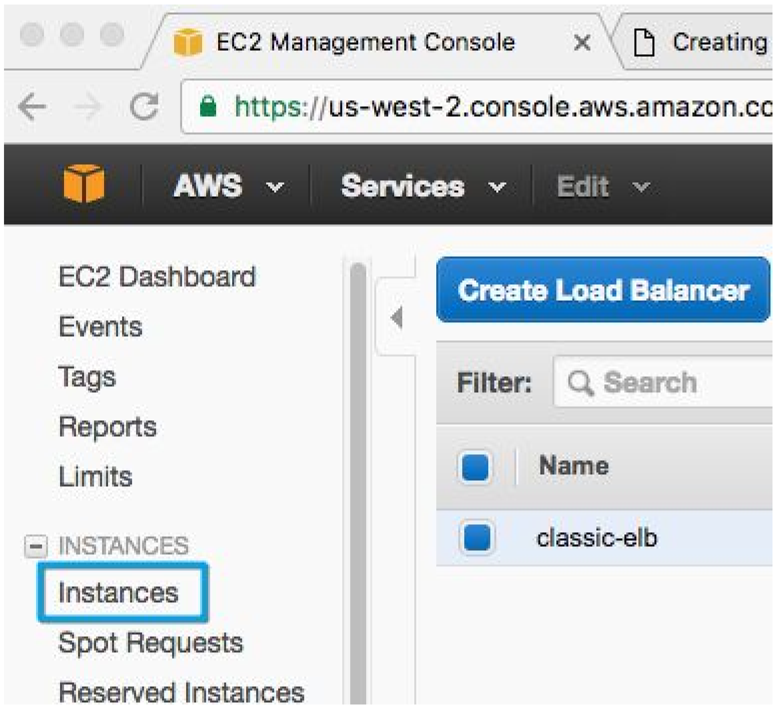


classic-elb-2118432540.us-west-2.elb.amazonaws.com

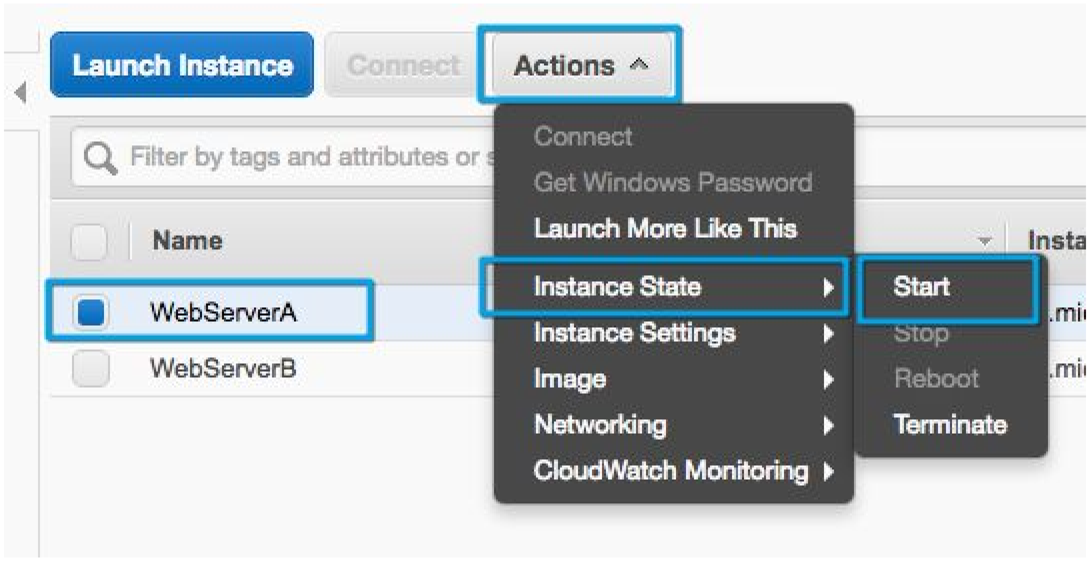
This app is awe

This is running on the instance `i-0d8ac0b2c759ee49b` with the public DNS

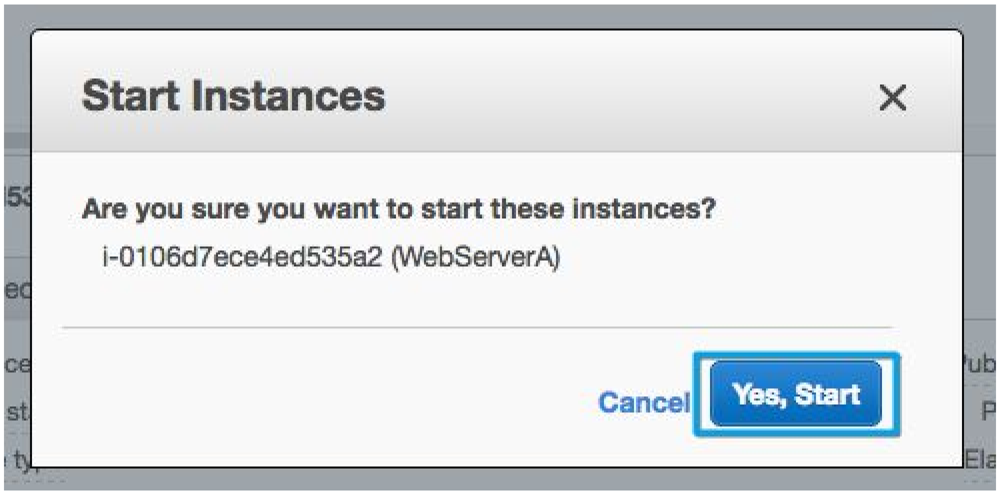
At this stage, no matter how many times you hit the refresh button, you will always be forwarded to the same instance. Hit refresh a few times, then go back to the **EC2 Management Console**, and click on **Instances**:



Start the stopped instance again by selecting it, clicking on **Actions, Instance State,** then on **Start**:



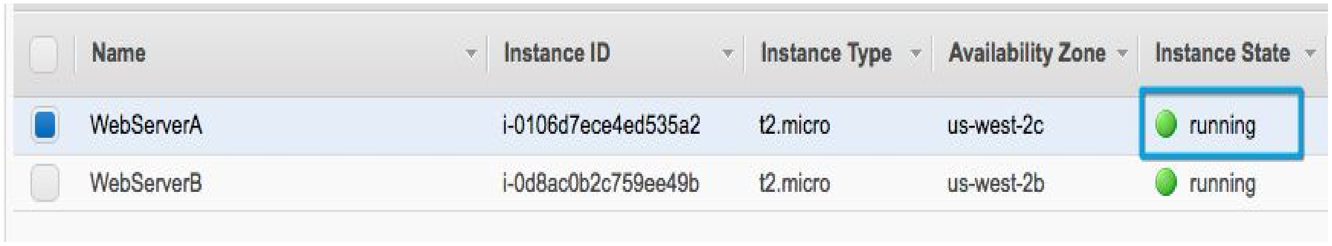
Click on **Yes, Start** in the dialogue box that pops up:


Start Instances

Are you sure you want to start these instances?

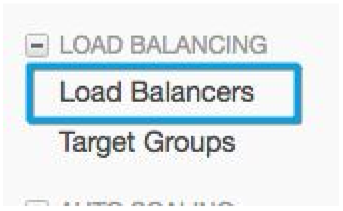
i-0106d7ece4ed535a2 (WebServerA)

Wait until the **Instance State** changes to **running**:

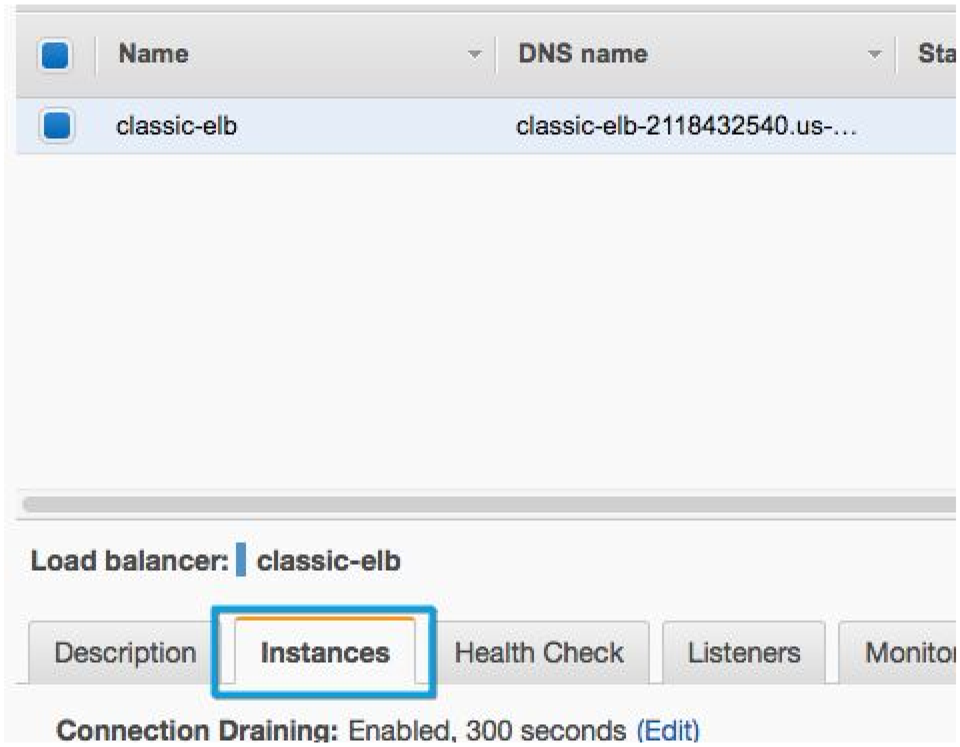


Name	Instance ID	Instance Type	Availability Zone	Instance State
WebServerA	i-0106d7ece4ed535a2	t2.micro	us-west-2c	running
WebServerB	i-0d8ac0b2c759ee49b	t2.micro	us-west-2b	running

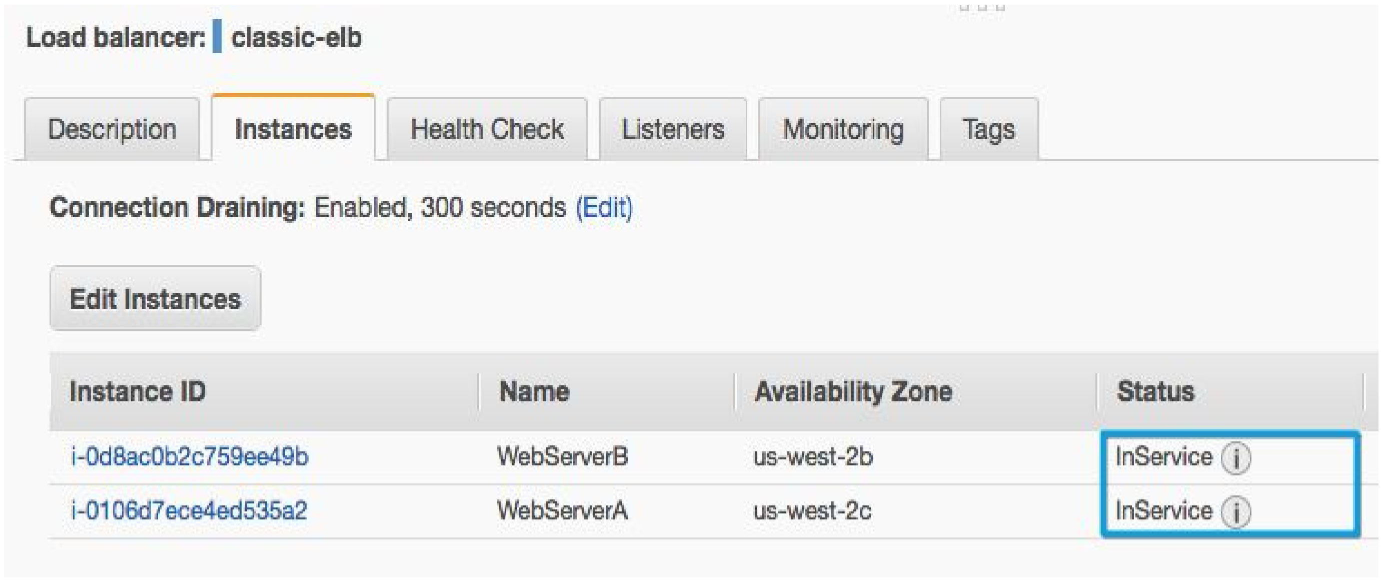
Then click on **Load Balancers**:



Select the LB and click on **Instances**:


Name
DNS name
Sta
classic-elb
classic-elb-2118432540.us-...
Load balancer: classic-elb
Description
Instances
Health Check
Listeners
Monitor
Connection Draining: Enabled, 300 seconds (Edit)

And you should be able to see that all the instances have an **InService** status again:



Connection Draining: Enabled, 300 seconds (Edit)

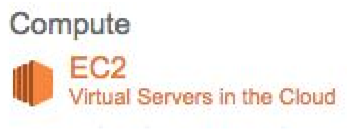
Instance ID	Name	Availability Zone	Status
i-0d8ac0b2c759ee49b	WebServerB	us-west-2b	InService
i-0106d7ece4ed535a2	WebServerA	us-west-2c	InService

You can test the LB's behavior again by accessing its **DNS Name** in your browser. Once you're done testing move to the next lab step.

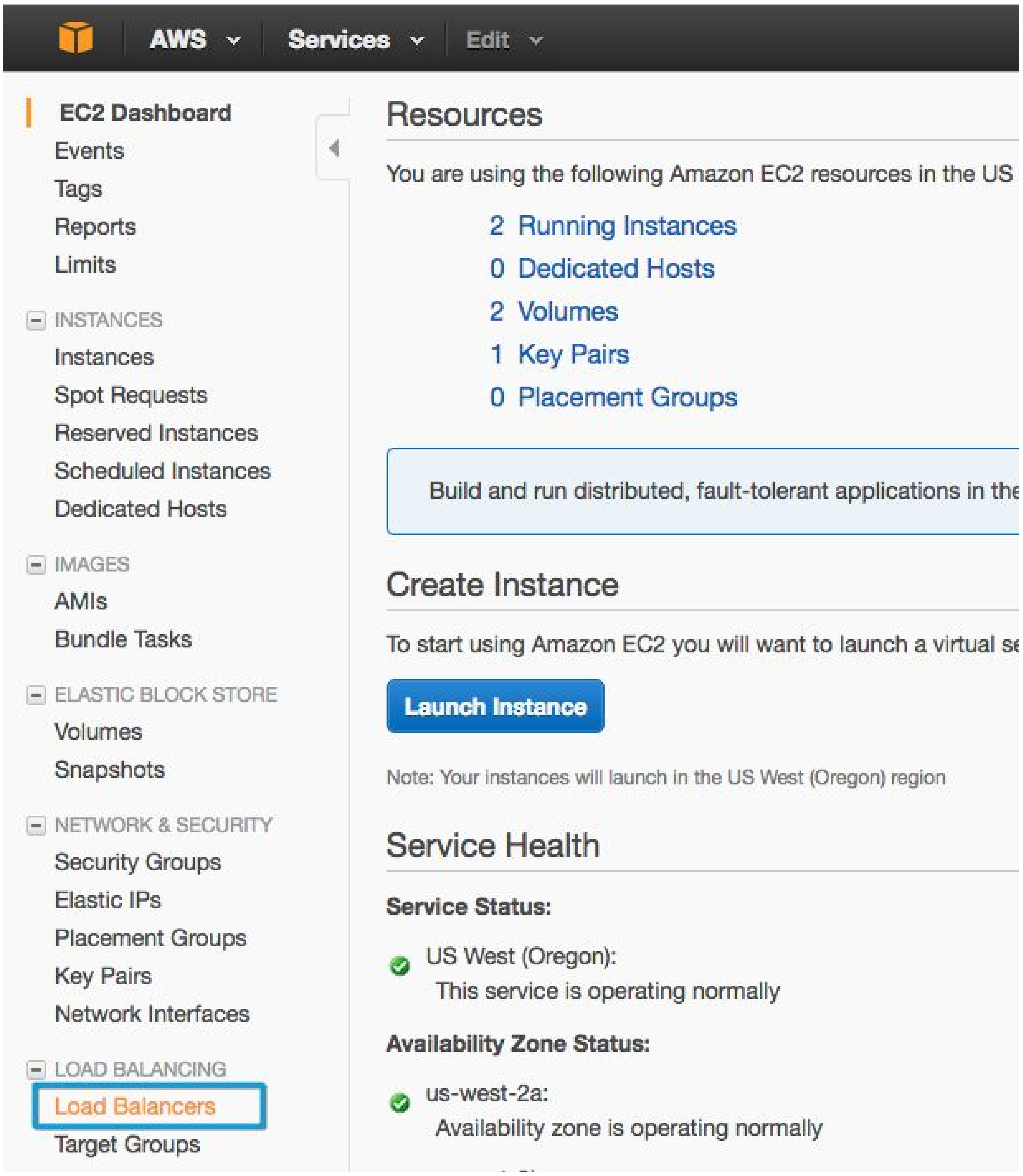
tep 6 Monitoring your Classic Load Balancer

In this step, we will take a quick look at the most common metrics for troubleshooting problems with your Classic Load Balancer and the EC2 instances running behind it.

There are two ways of doing that. One is on the CloudWatch console, which can be a bit frustrating for newcomers because it will hold metrics for ALL LBs that existed within the past 2 weeks in the AWS account you're using, and you might get lost with so many metrics. With that in mind, we will take a different approach, and we will take a look at the metrics related to our LB in the EC2 console. To do so, go to the EC2 console:



And click on Load Balancers:



Resources

You are using the following Amazon EC2 resources in the US

2 Running Instances
0 Dedicated Hosts
2 Volumes
1 Key Pairs
0 Placement Groups

Build and run distributed, fault-tolerant applications in the

Create Instance

To start using Amazon EC2 you will want to launch a virtual se

Note: Your instances will launch in the US West (Oregon) region

Service Health

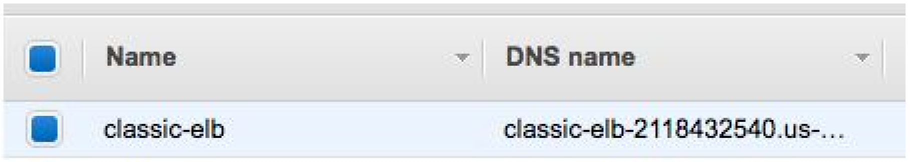
Service Status:

US West (Oregon):
This service is operating normally

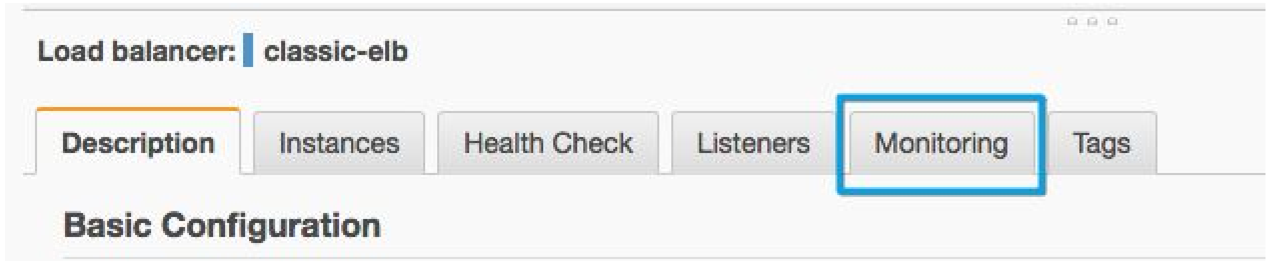
Availability Zone Status:

us-west-2a:
Availability zone is operating normally

Then select the LB you just created:



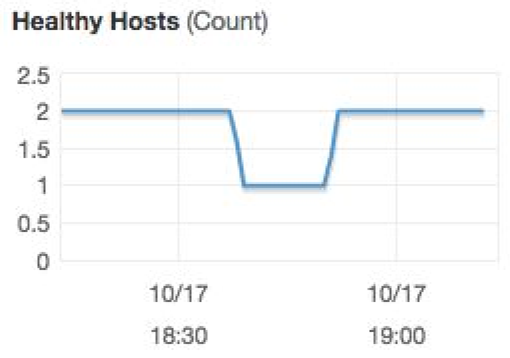
Click on the **Monitoring** tab to see the metrics of the LB you selected:



The ELB service reports metrics to CloudWatch only when requests are flowing through the LB. If there are requests flowing through the LB, ELB measures and sends its metrics in 60-second intervals. If there are no requests flowing through the load balancer, or no data for a metric, the metric is not reported. There are a few metrics related to a Classic Load Balancer, and in general they are self-explanatory. However, some of them may be unfamiliar to you, in which case you can take a look at the description for all metrics in here: http://docs.aws.amazon.com/elasticloadbalancing/latest/classic/elb-cloudwatch-metrics.html

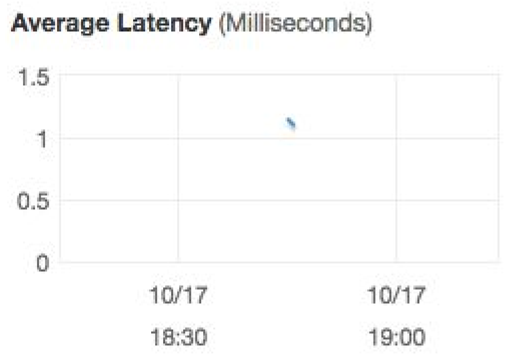
The metrics called **HealthyHostCount**, and **UnHealthyHostCount** will count the number of Healthy and Unhealthy instances respectively. These metrics can be useful for you to identify a major problem in your AWS account. For example, you could set up some CloudWatch Alarms to notify you when you have less than 2 instances running your application, though to be clear this is not a general rule: the number of instances that might identify a problem will vary depending on your environment.

Also notice that in these metrics, there is no way of seeing the Availability Zone to which the Healthy/Unhealthy instance belongs. In our lab, we stopped an instance for a few minutes, therefore you should be able to see something like this:



If the **Healthy Hosts** metric reaches 0, that means that people won't see anything when accessing your LB, and it is probable that you have a big problem in your infrastructure.

The **Average Latency** metric might be useful to identify potential issues in your setup. Maybe everything is working in your application, but you notice an increase in this metric. If you haven't changed anything in your application, that can be a potential issue - maybe you haven't provisioned enough EC2 instances, or you even have lots of instances but they don't have enough power to serve your increasing traffic.



The other metrics can be very useful for troubleshooting specific scenarios and will vary depending on your setup.

You have now covered all the Learning Objectives in this lab. You can take some time to play around with the metrics. You can do things such as:

- Stop ALL instances and make requests to the LB
- Make several requests to paths that don't exist, such as <dns name>/app, and <dns name>/users to generate some errors in the responses

Once you have explored enough, you can move to the next lab step. Please also be sure to rate this lab.