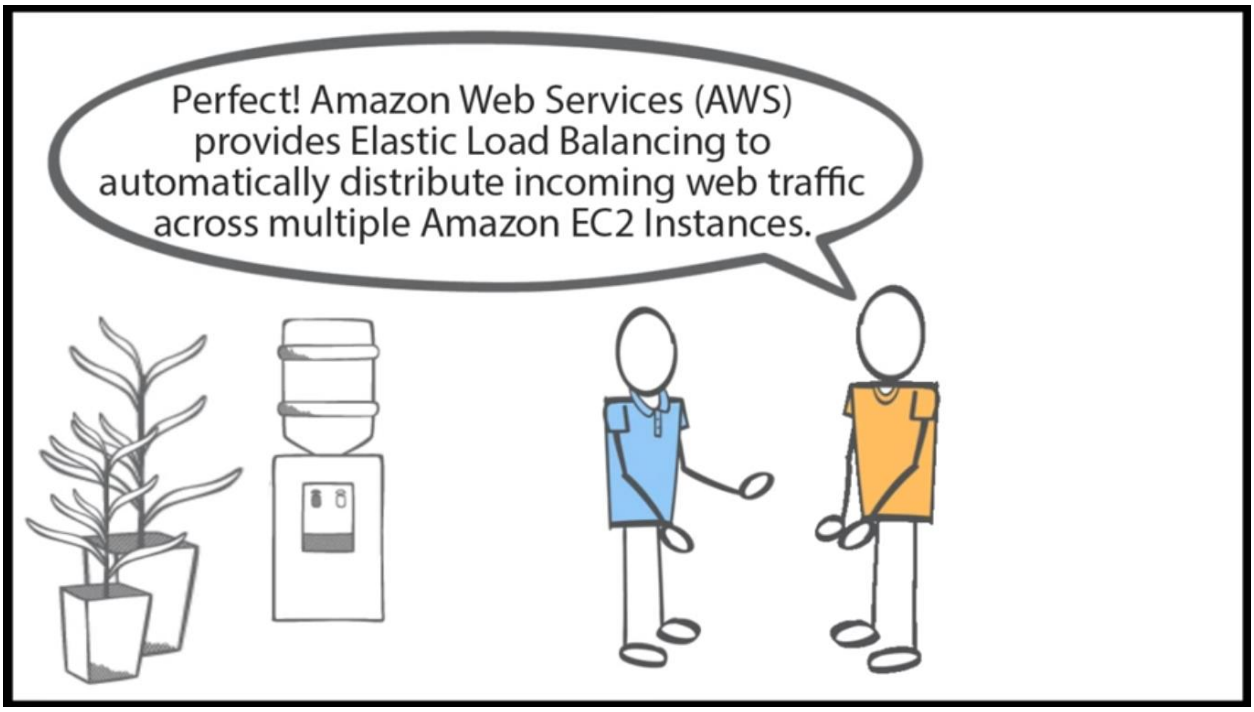
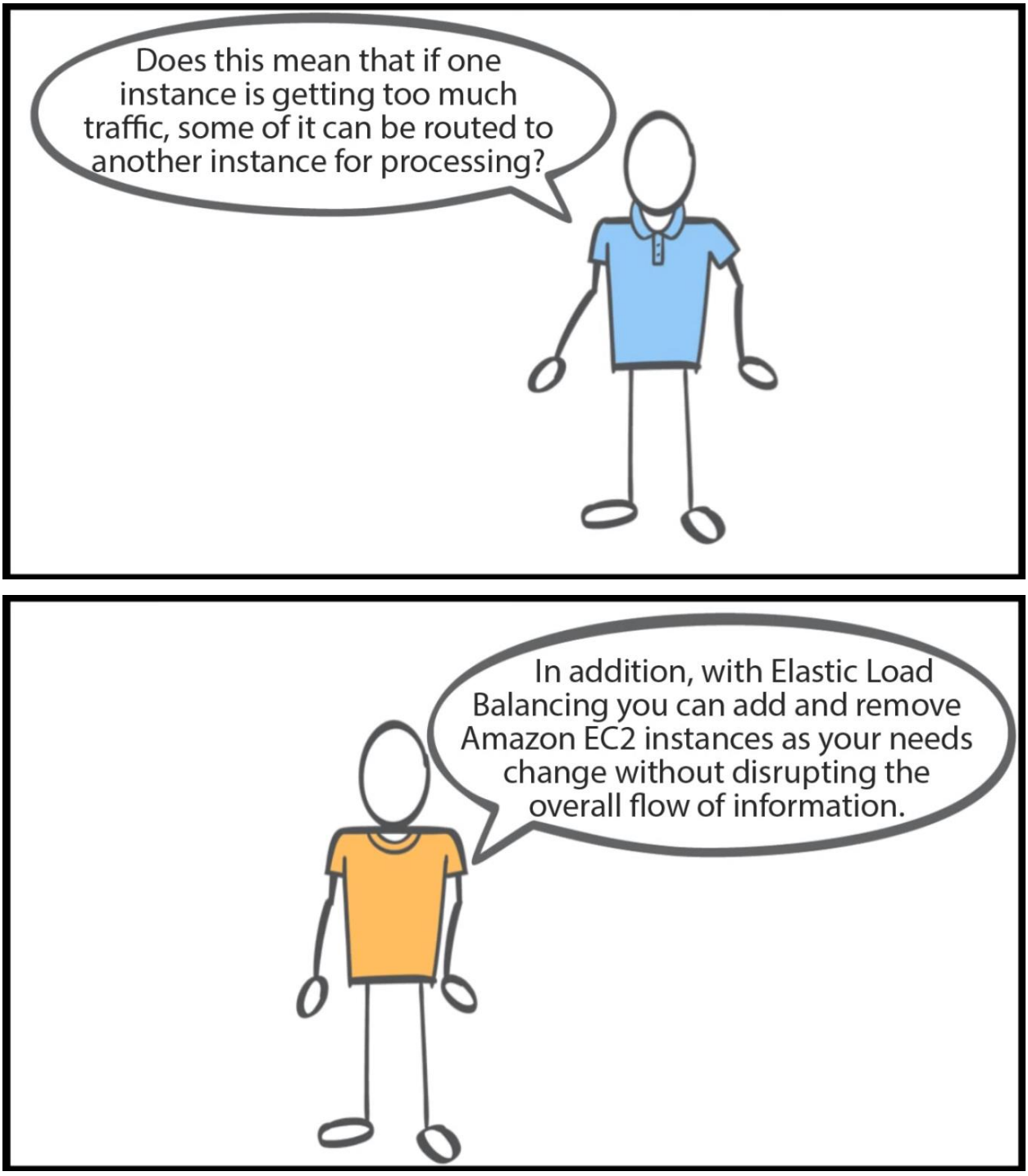
Elastic Load Balancing

Elastic Load Balancing (ELB) automatically distributes incoming application traffic across multiple Amazon EC2 instances. It enables you to achieve greater fault tolerance in your applications and seamlessly provides the correct amount of load balancing capacity needed in response to incoming application traffic.

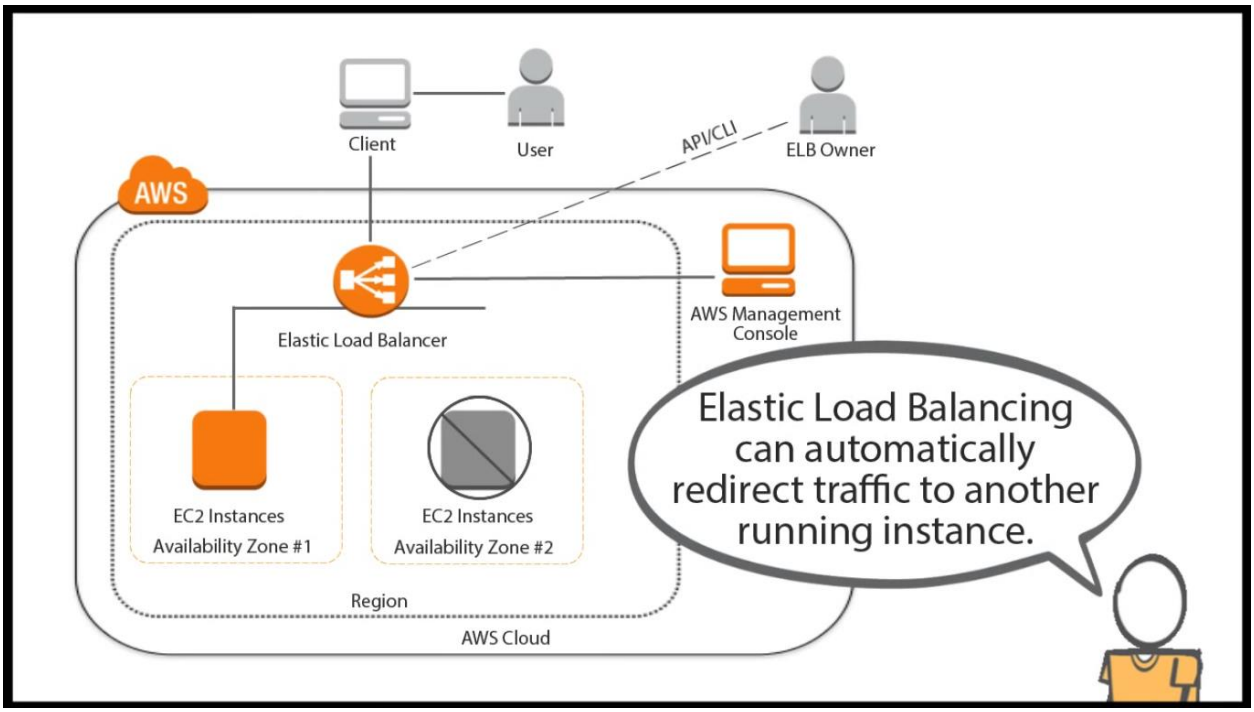
Elastic Load Balancing detects unhealthy instances within a pool and automatically reroutes traffic to healthy instances until the unhealthy instances have been restored to health. Customers can enable Elastic Load Balancing within a single Availability Zone or across multiple zones but not across multiple regions for greater consistent application performance.

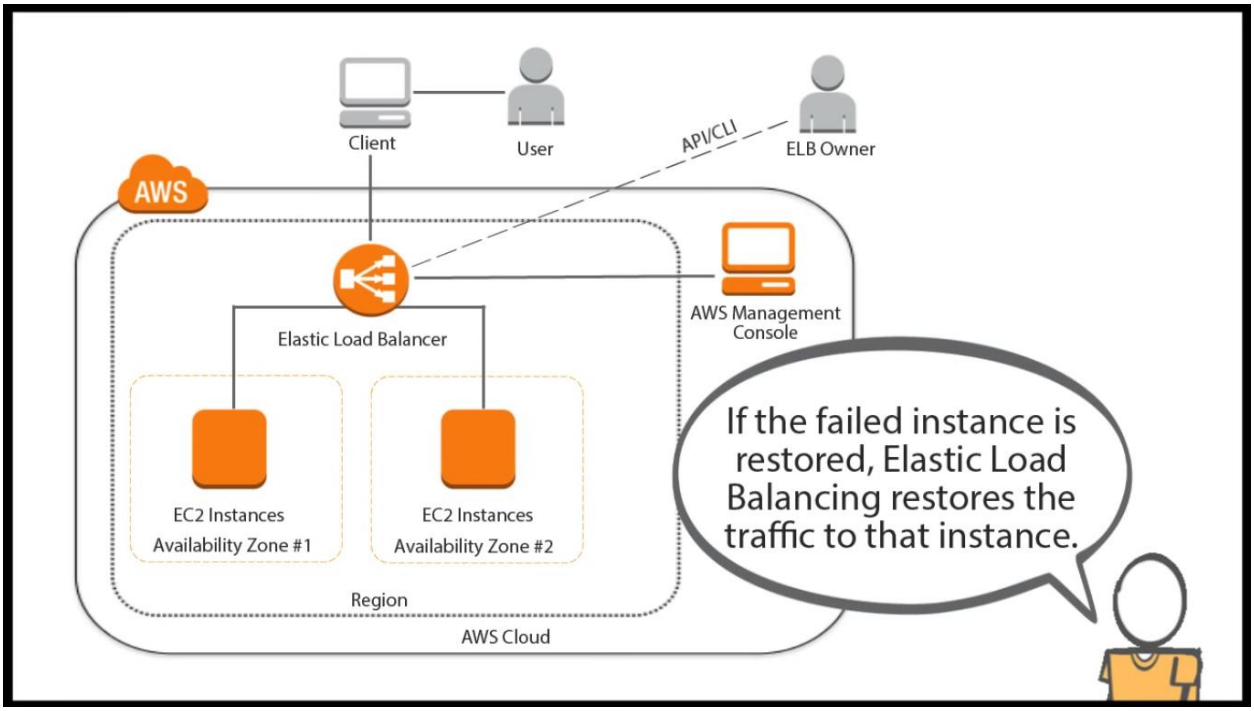
Interactive tutorial:



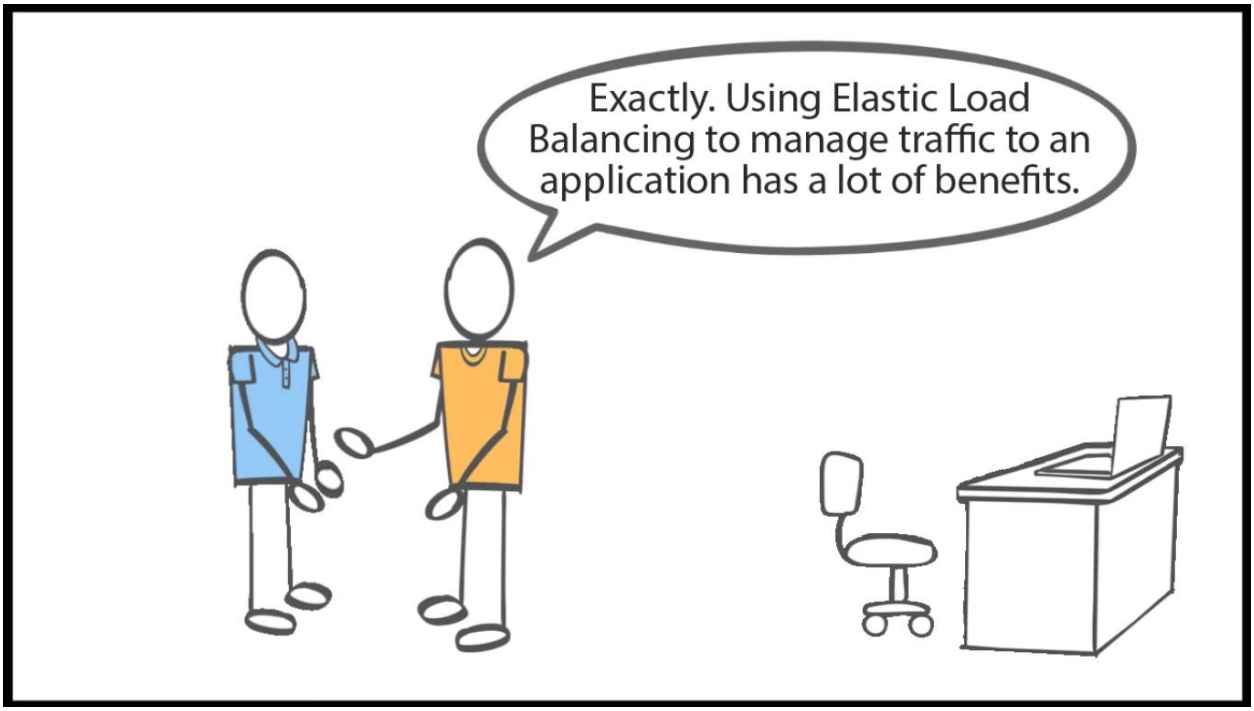

Does this mean that if one instance is getting too much traffic, some of it can be routed to another instance for processing?
In addition, with Elastic Load Balancing you can add and remove Amazon EC2 instances as your needs change without disrupting the overall flow of information.

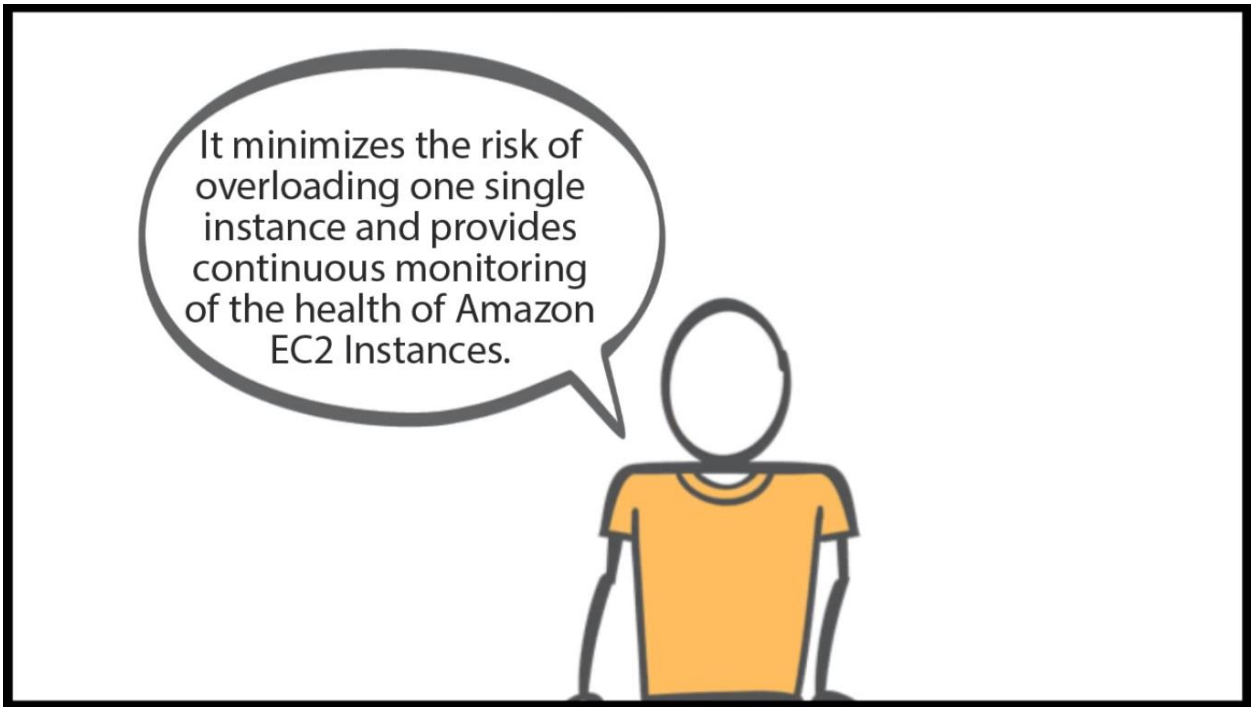

What if the instance happens to fail?

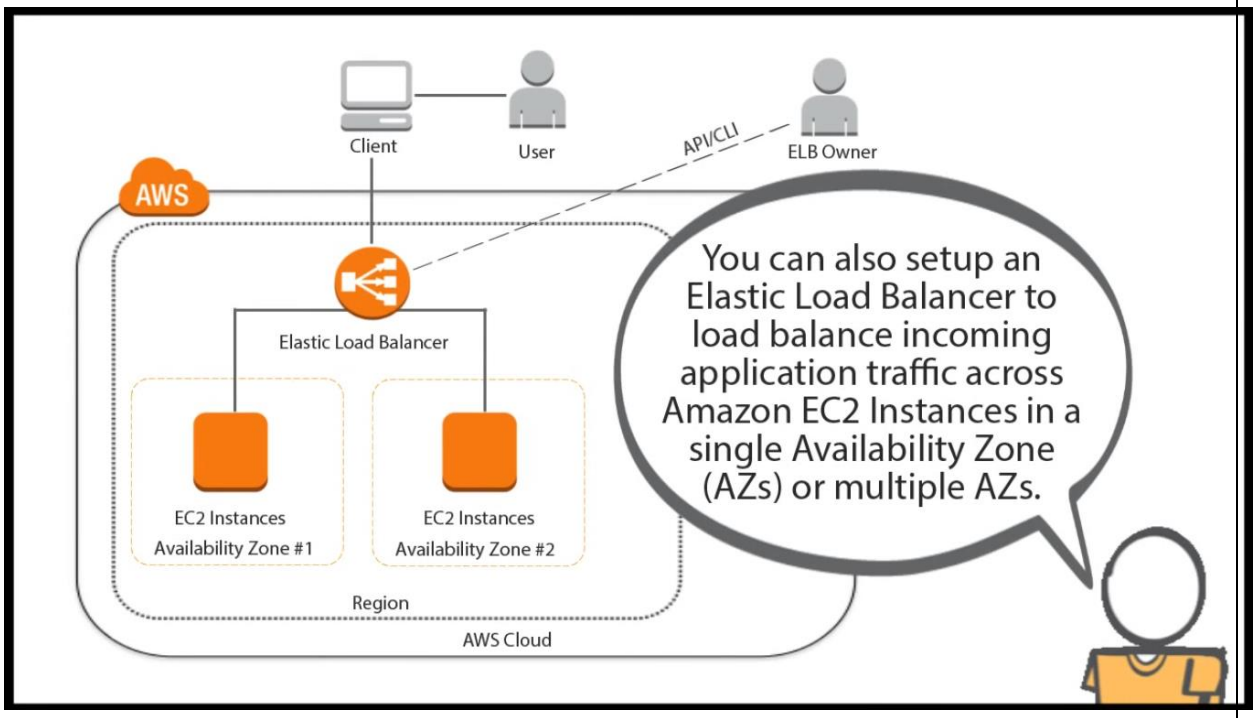

Client
User
API/CLI
ELB Owner
AWS
AWS Management Console
Elastic Load Balancer
EC2 Instances
Availability Zone #1
EC2 Instances
Availability Zone #2
Region
AWS Cloud
Elastic Load Balancing can automatically redirect traffic to another running instance.

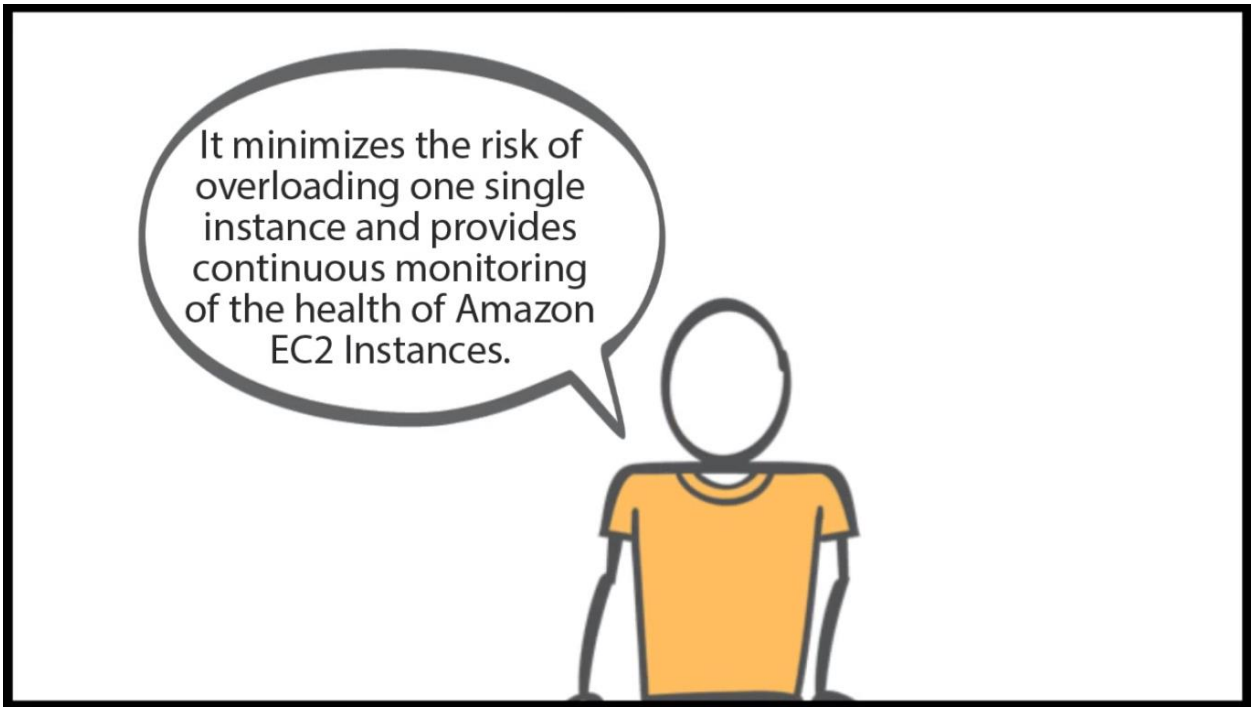

Client
User
API/CLI
ELB Owner
AWS
Elastic Load Balancer
AWS Management Console
EC2 Instances
Availability Zone #1
EC2 Instances
Availability Zone #2
Region
AWS Cloud
If the failed instance is restored, Elastic Load Balancing restores the traffic to that instance.

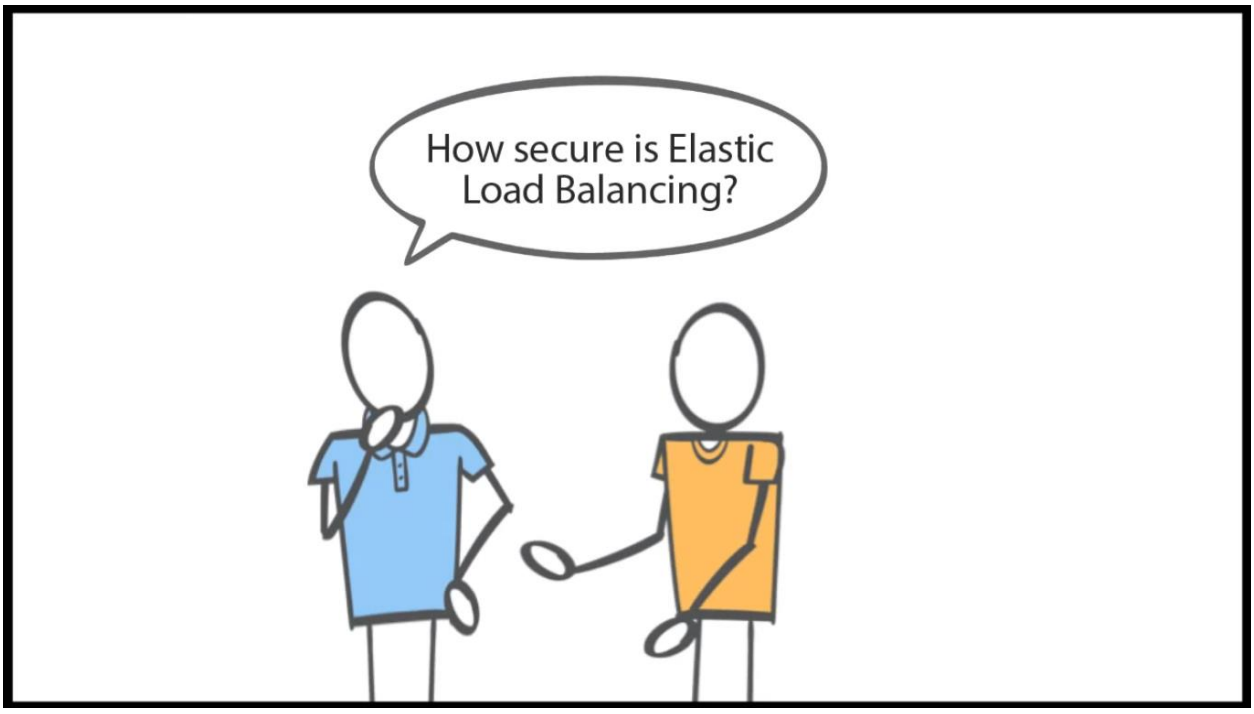

Cool, so Elastic Load Balancing is like a traffic manager?

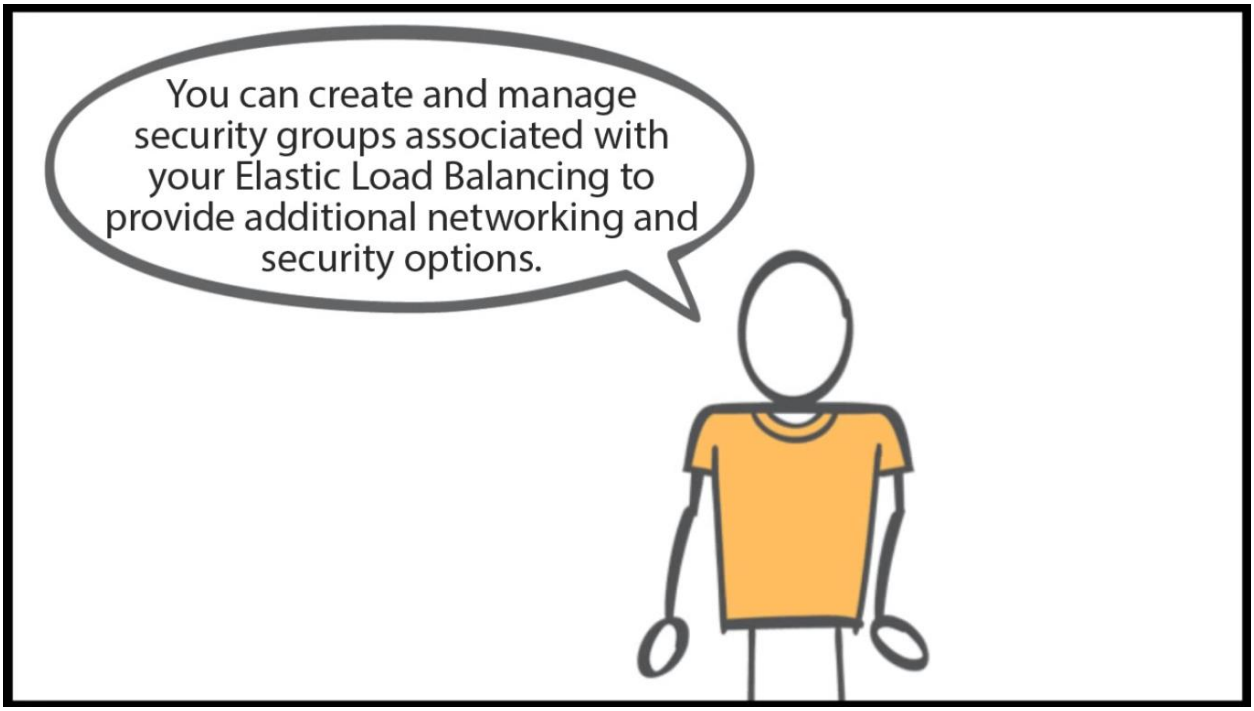

Exactly. Using Elastic Load Balancing to manage traffic to an application has a lot of benefits.

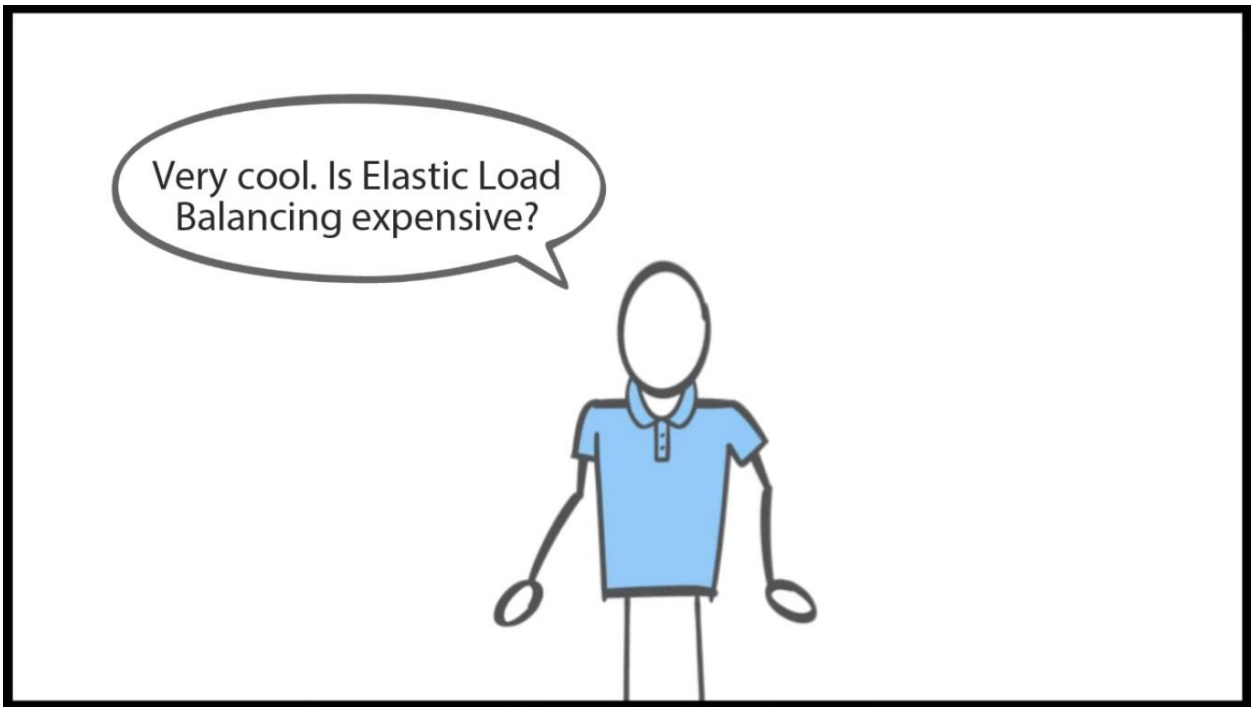

It minimizes the risk of overloading one single instance and provides continuous monitoring of the health of Amazon EC2 Instances.


Client
User
API/CLI
ELB Owner
AWS
Elastic Load Balancer
EC2 Instances
Availability Zone #1
EC2 Instances
Availability Zone #2
Region
AWS Cloud
You can also setup an Elastic Load Balancer to load balance incoming application traffic across Amazon EC2 Instances in a single Availability Zone (AZs) or multiple AZs.


It minimizes the risk of overloading one single instance and provides continuous monitoring of the health of Amazon EC2 Instances.


How secure is Elastic Load Balancing?

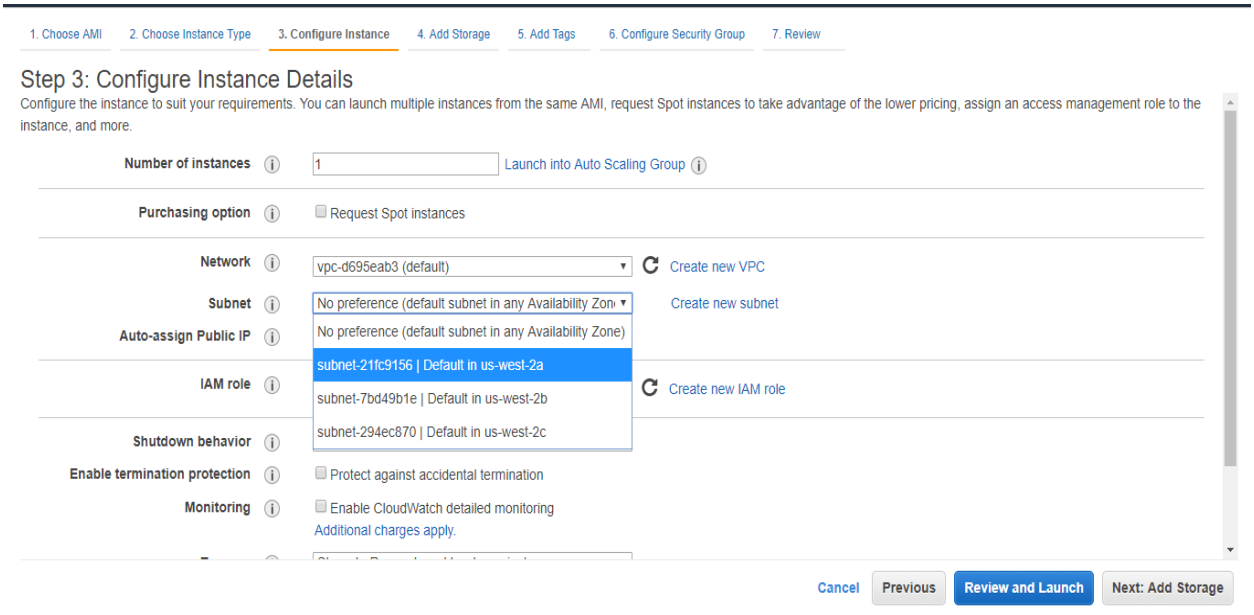

You can create and manage security groups associated with your Elastic Load Balancing to provide additional networking and security options.


Very cool. Is Elastic Load Balancing expensive?

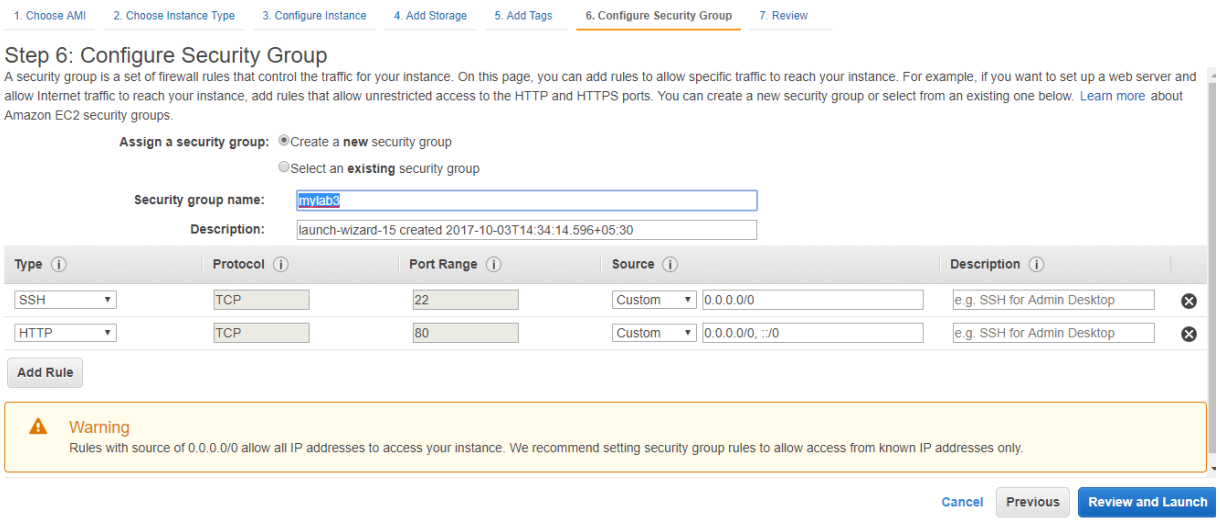
Lab: How an ELB distributes traffic across multiple instances

Step 1.Click on EC2 Dashboard

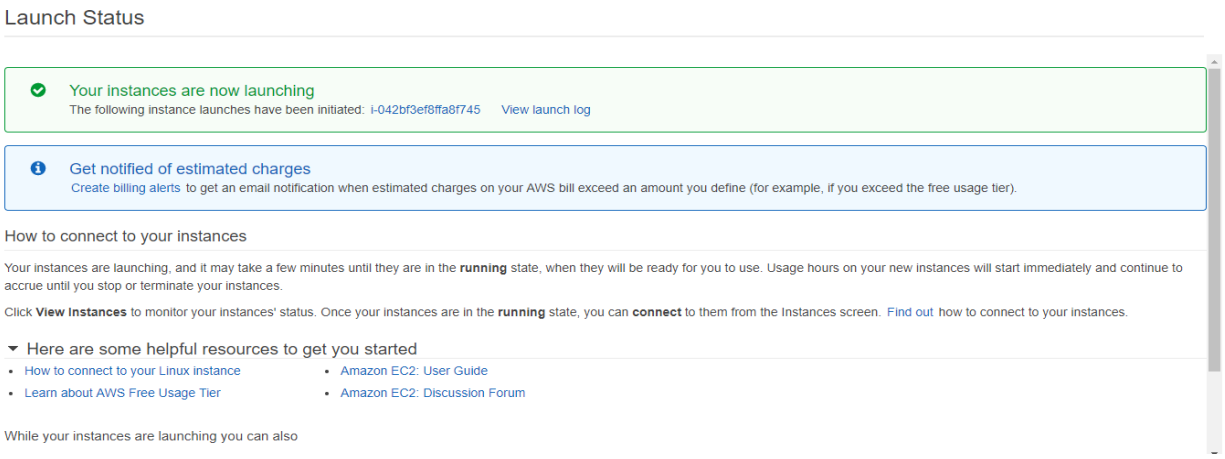
Step 2. From the EC2 dashboard, click on launch instances with usual Amazon Linux AMI and select free tier general purpose of instance type. Click Next and select and in the subnet select us-west-2a (assuming you have selected US west: Oregon region).



Step 3: Then click next:Add storage and then click next:Add tags and then click next: configure security groups. Now configure security groups as shown below and give it a name 'mylab3'



Step 4: Click review and launch and then launch and chose a key pair if you have already created one. Then click 'Launch instances'

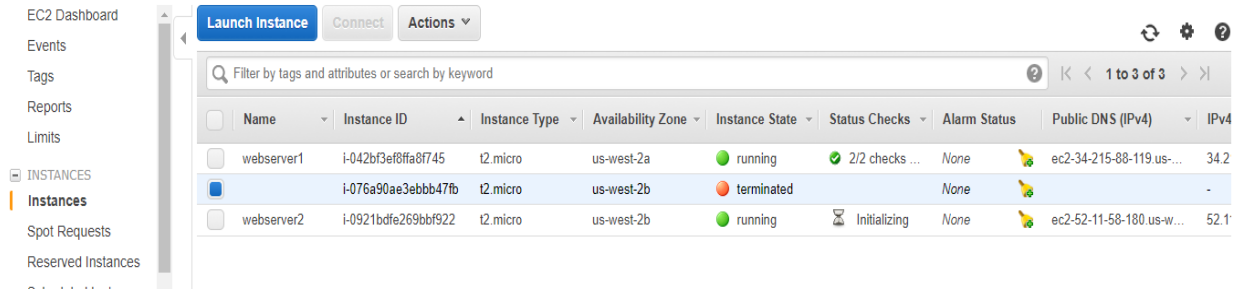


As you can see you've created an instance.

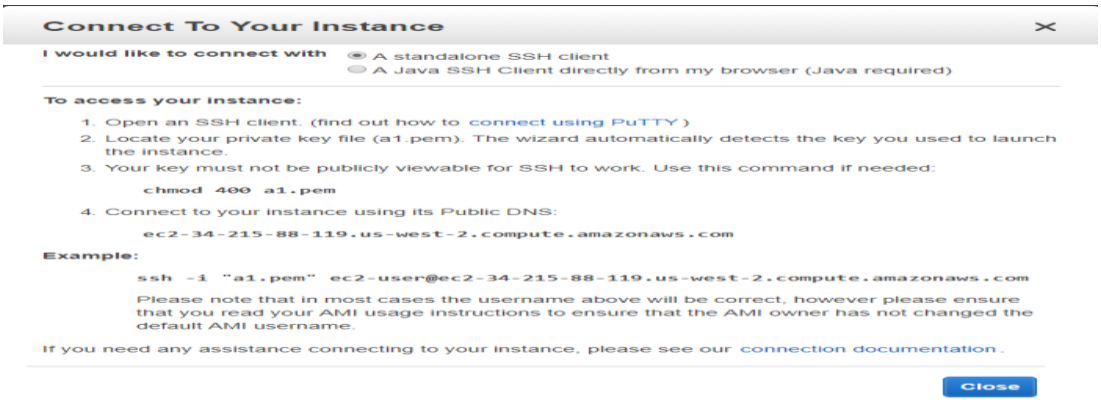
Now repeat the same steps(1-4) and create another instance(you can use the same security group 'mylab3' but make sure that you create that instance in US west-2b or US west-2c but not in US west-2a as we've already created one.

So,far we've created two instances in two availability zones. Now let's connect to each of them and install httpd web server on it and do some stuff

Step 5: I named my 1st instance as webserver 1 and 2nd instance and webserver2.Now I'm going to connect it webserver through putty.

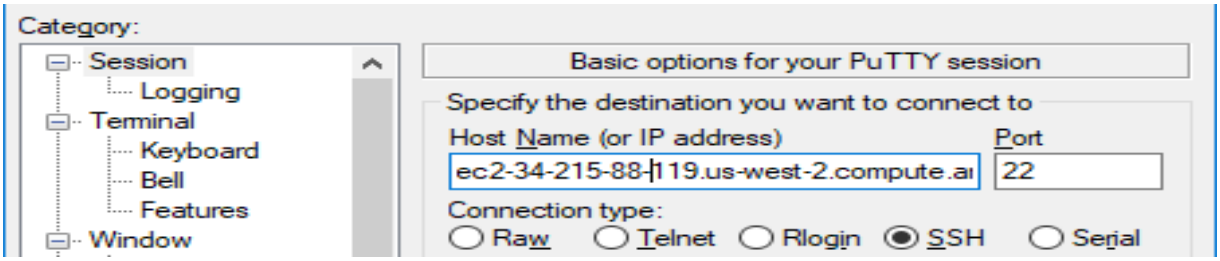


Select webserver 1 and click on 'connect' button to follow instructions:

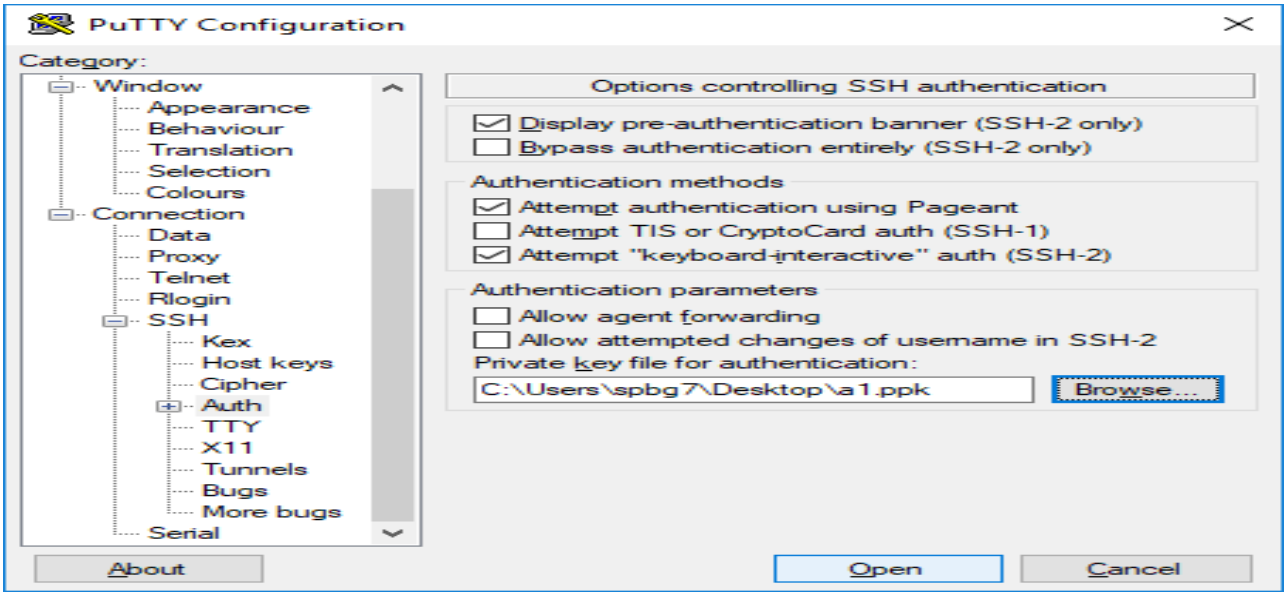


Now open putty

In the hostname copy and paste the Public DNS in Hostname(or IP address) as shown in the pic just right above.

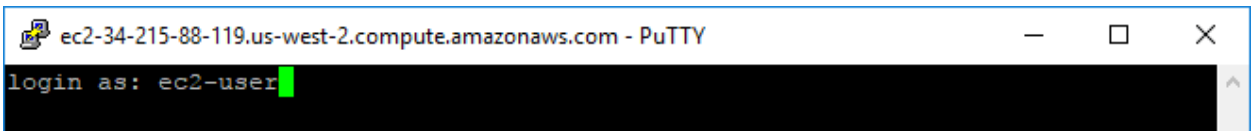


Then on the left column of 'category:' click on SSH ->Auth->browse private key(the one you already created and converted using Puttygen) and click OPEN

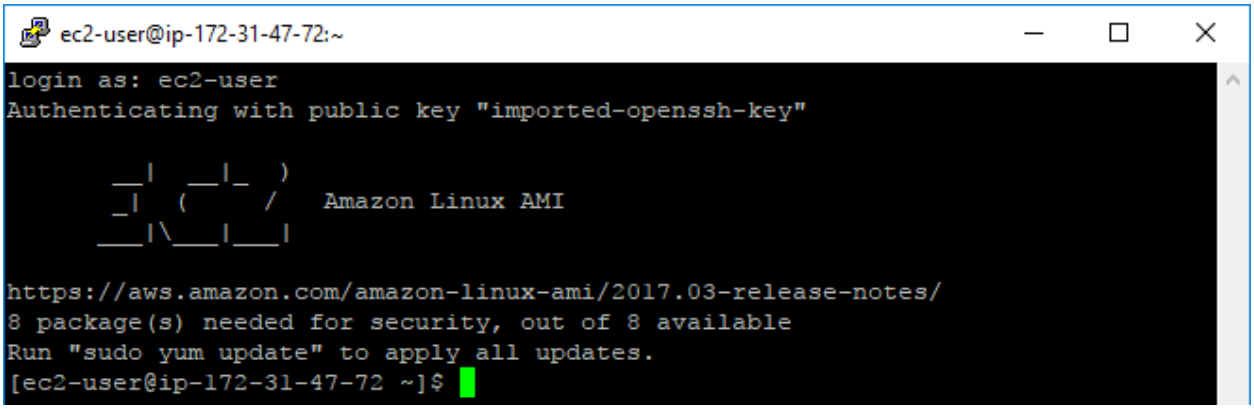


Click 'yes' for security alert and now a dark screen pops up and

enter login as :ec2-user.Then click enter.



Now you're connected to your instance(webserver1):



Step 6:Install httpd server and create a simple static web page

```
[ec2-user@ip-172-31-47-72 ~]$ sudo yum install httpd
```

```
Complete!
[ec2-user@ip-172-31-47-72 ~]$
```

Now start the httpd webserver and make sure it is running:

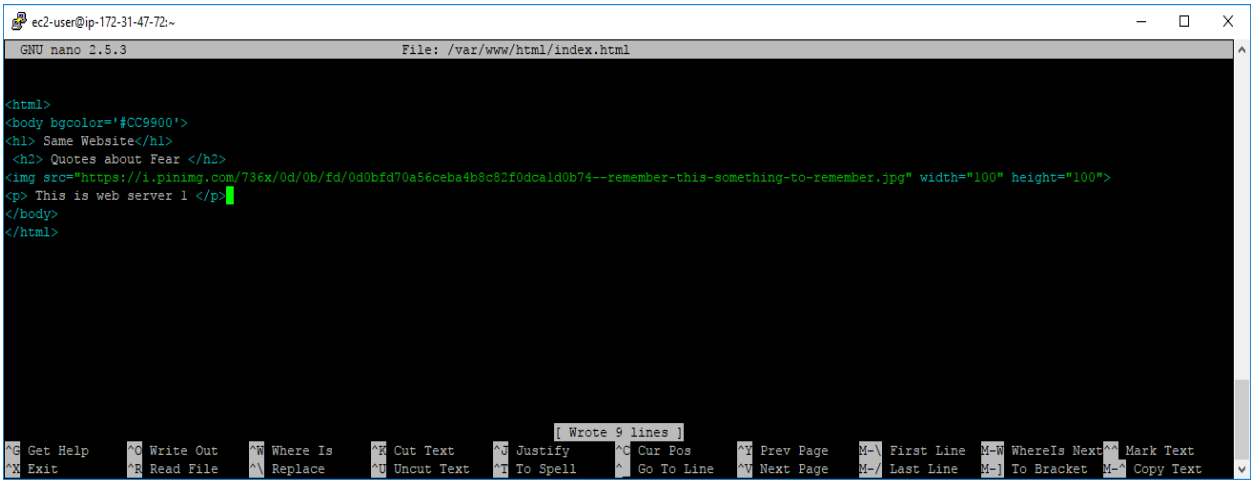
```
ec2-user@ip-172-31-47-72:~
[ec2-user@ip-172-31-47-72 ~]$ sudo service httpd start
Starting httpd:                                            [  OK  ]
[ec2-user@ip-172-31-47-72 ~]$ sudo chkconfig httpd on
[ec2-user@ip-172-31-47-72 ~]$ sudo service httpd status
httpd (pid  2923) is running...
```

Now create a simple webpage index.html

```
ec2-user@ip-172-31-47-72:~
[ec2-user@ip-172-31-47-72 ~]$ sudo nano /var/www/html/index.html
```

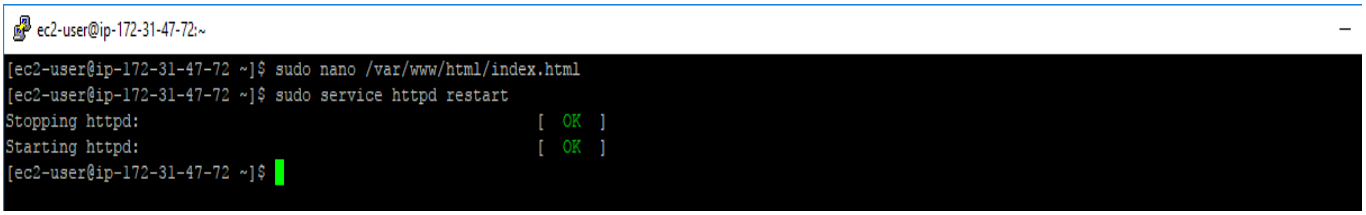
Now an editor will be opened up .Copy and paste this simple HTML script there. Feel free to use if to use if want to create your own

```
<html>

<body bgcolor='#CC9900'>

<h1> Same Website</h1>

 <h2> Quotes about Fear </h2>

<img
src="https://i.pinimg.com/736x/0d/0b/fd/0d0bfd70a56ceba4b8c82f0dca1d0b74--remember-this-something-to-remember.jpg" width="100" height="100">

<p> This is web server 1 </p>

</body>

</html>
```

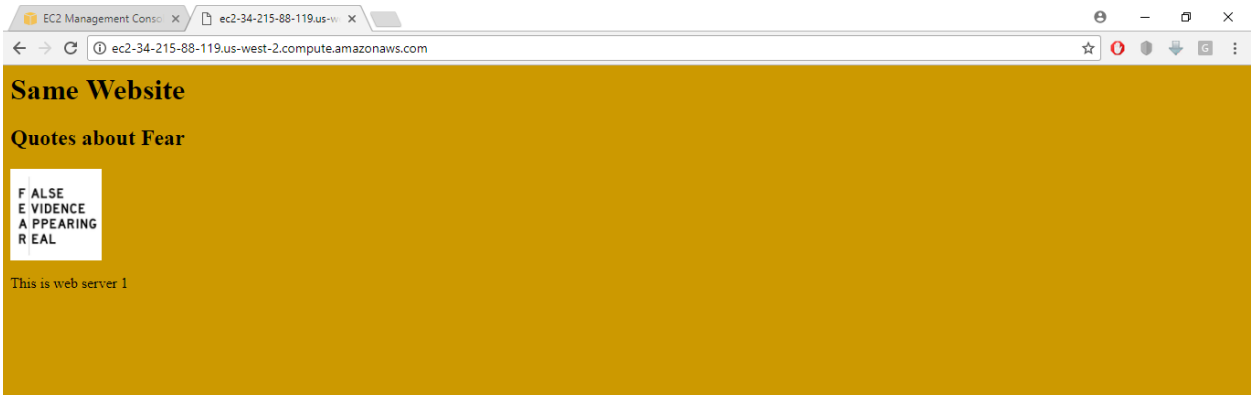


Now click Ctrl O to save and Ctrl X to exit.

Now restart the server



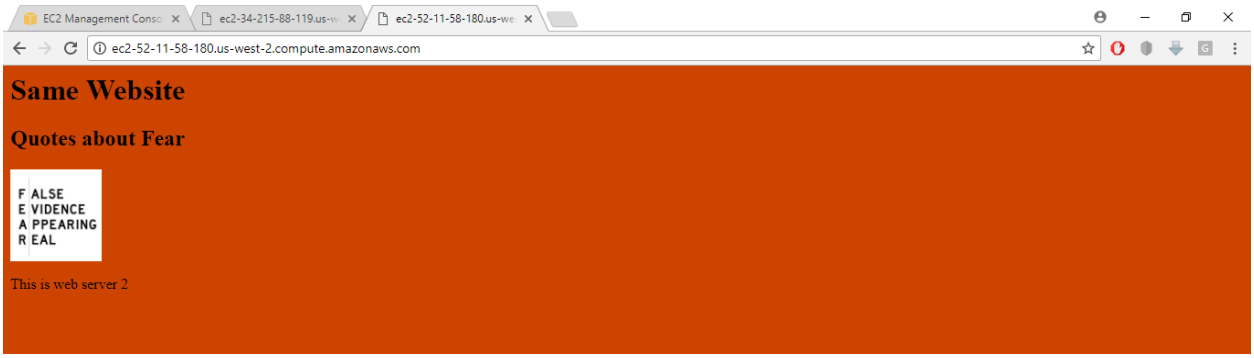
Now copy and paste the public DNS of your instance in a separate browser. You'll see this



Step 7: Connecting instance 2(webserver 2)

Now repeat the steps 5-6 exactly for webserver 2 except for the HTML code. Change the content slightly so that there is some difference in appearance between webserver 1 and webserver2.

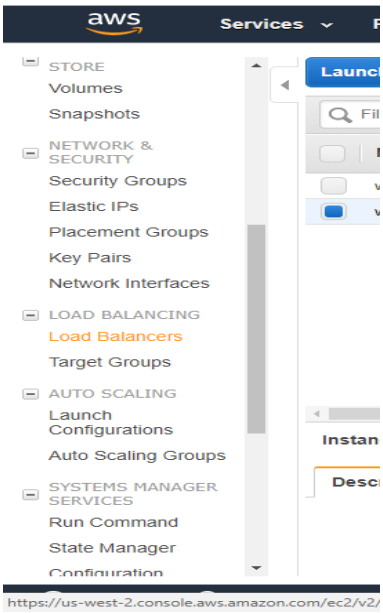
After that load the DNS for webserver2 on separate web browser you'll see this:



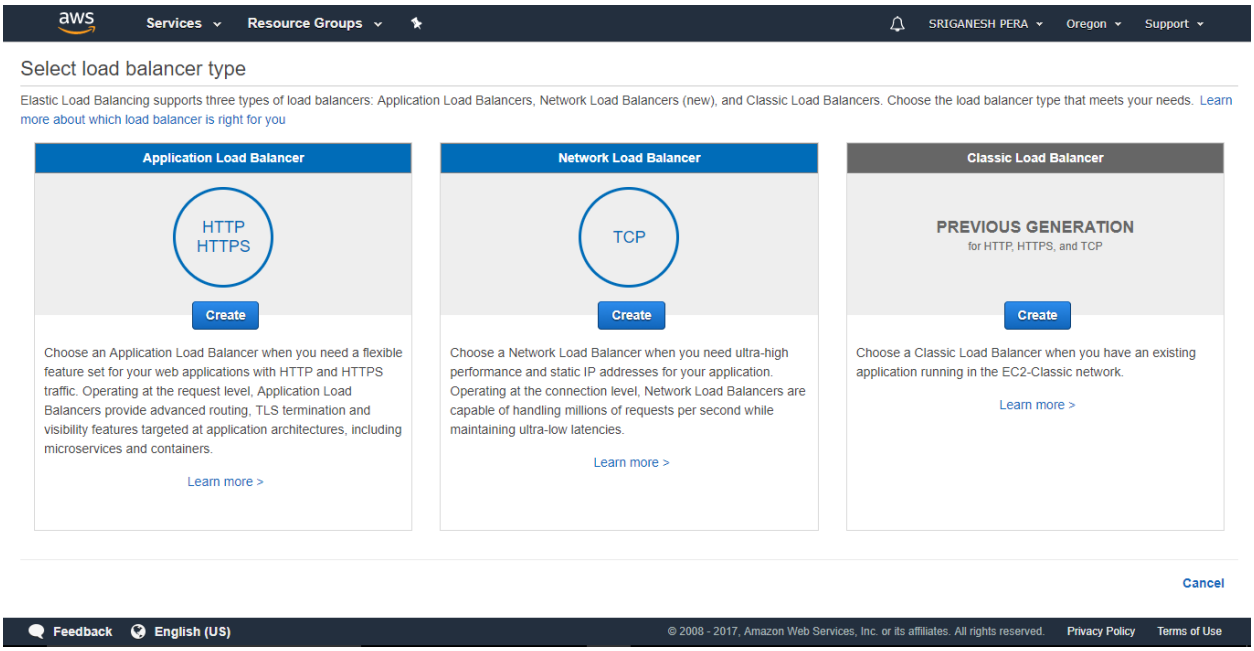
If you notice the 2 static web page images,the only change is the background color and text ("This is web server 2")

Step 7: Now launch an ELB which directs the traffic through it and distribute across the instances

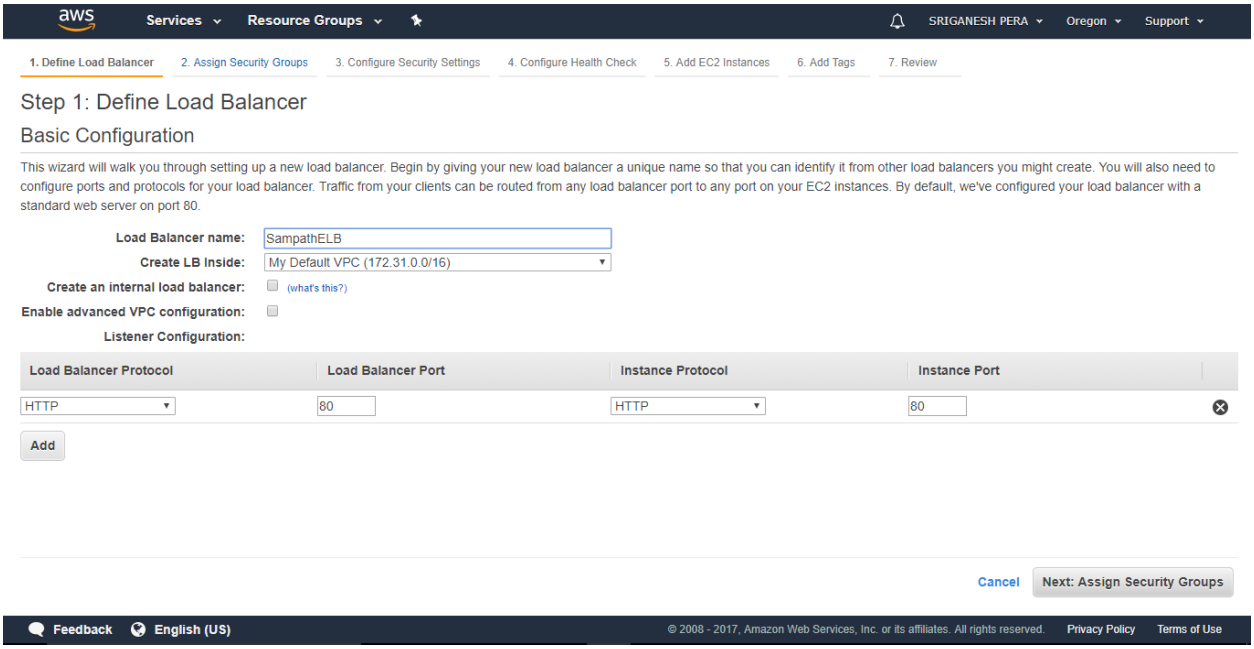
Go to EC2 dashboard and on the left-hand side you will find load balancer column. Now Click on it.



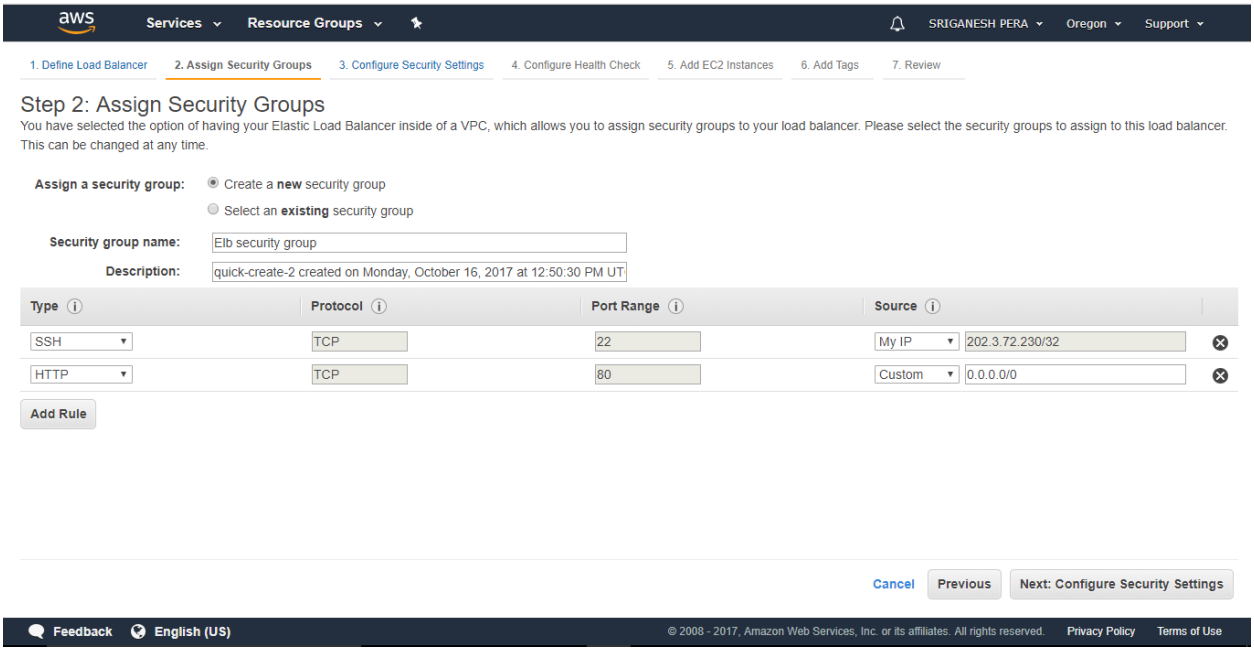
Step 8: Select Classic Load Balancer and click Create



Step 9: Name the load balancer and click Next:Assign Security groups

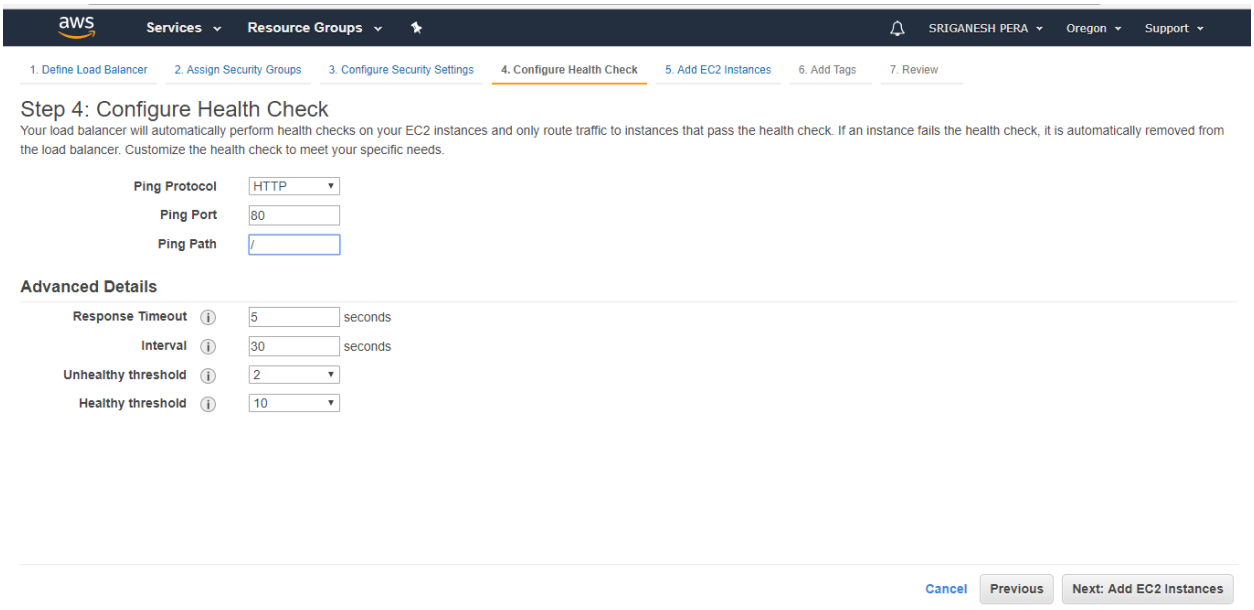


Step 10: Create your own security group or select one if you have one but the setting should be this:



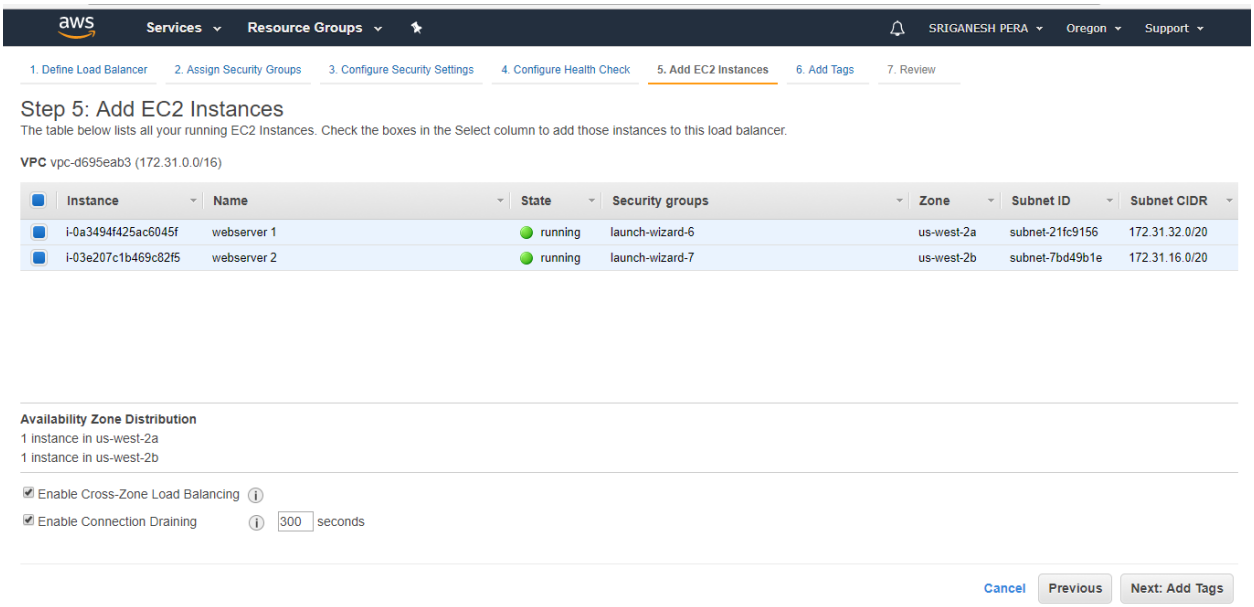
Step 11: Now click Next: Configure security settings and Next: Configure heath checks

In the ping path just keep only the forward slash as show:



Step 12: Now add EC2 instances

Select both the instances and click Next:Add Tags

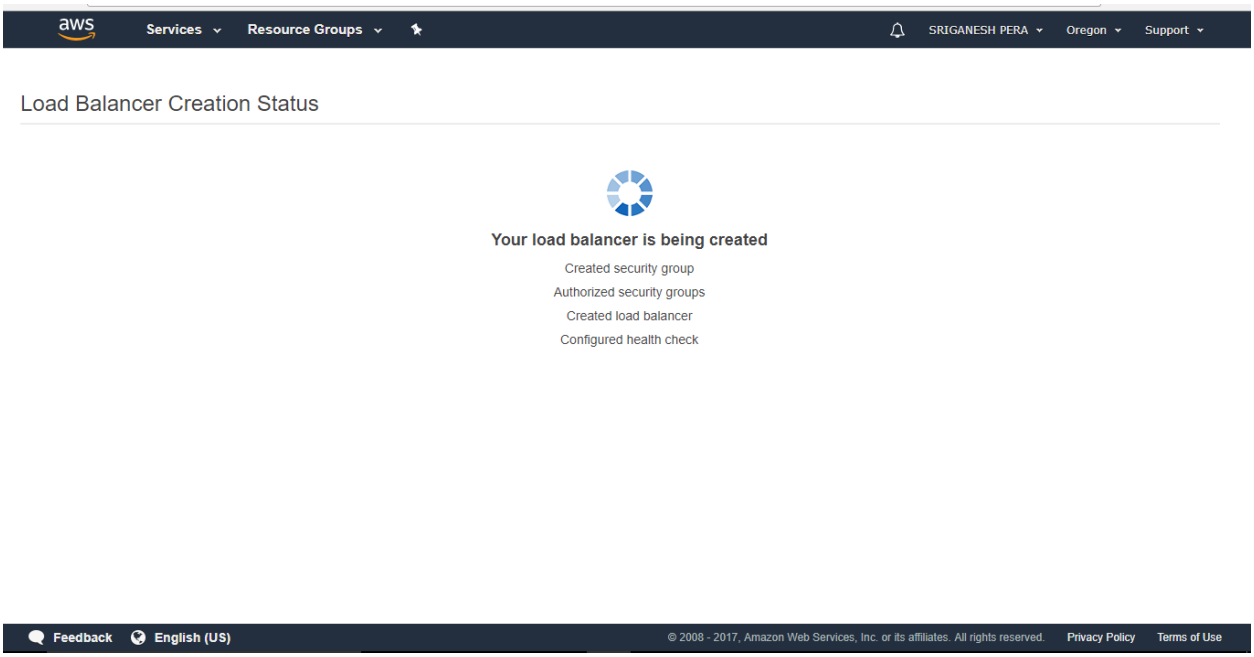


There are 2 config points in here as well:

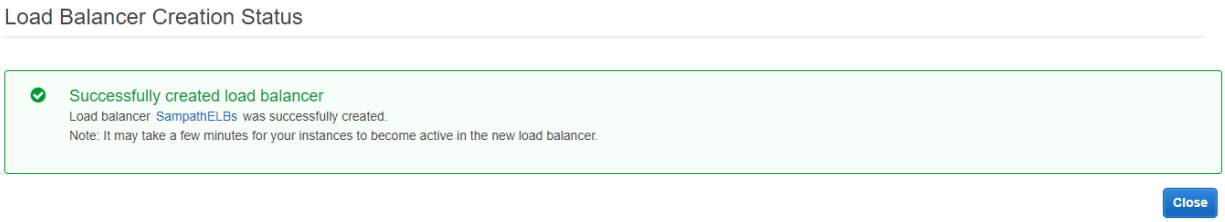
Cross-Zone Load Balancing is used to ensure that your LB distributes incoming requests evenly across all instances in its enabled Availability Zones. That means that the LB will ignore the default of round-robin and will also take into consideration the Availability Zone in which the instance is running. 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.

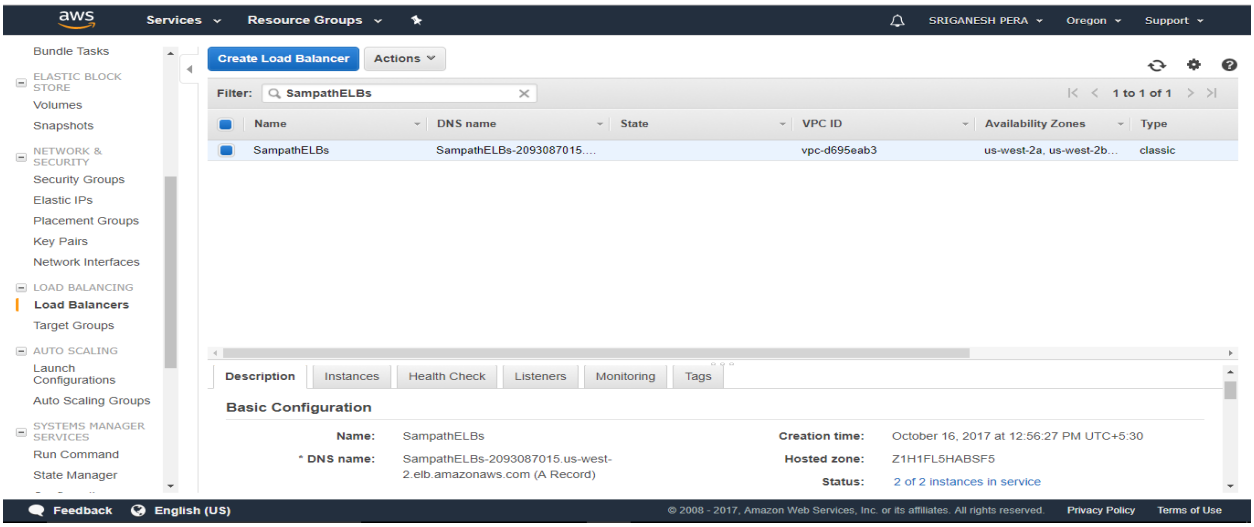
Connection Draining is used to ensure that a Classic Load Balancer stops sending requests to instances that are de-registering or unhealthy while keeping the existing connections open.

Step 13: Now click Review and Create and review all the settings you made and then click Create



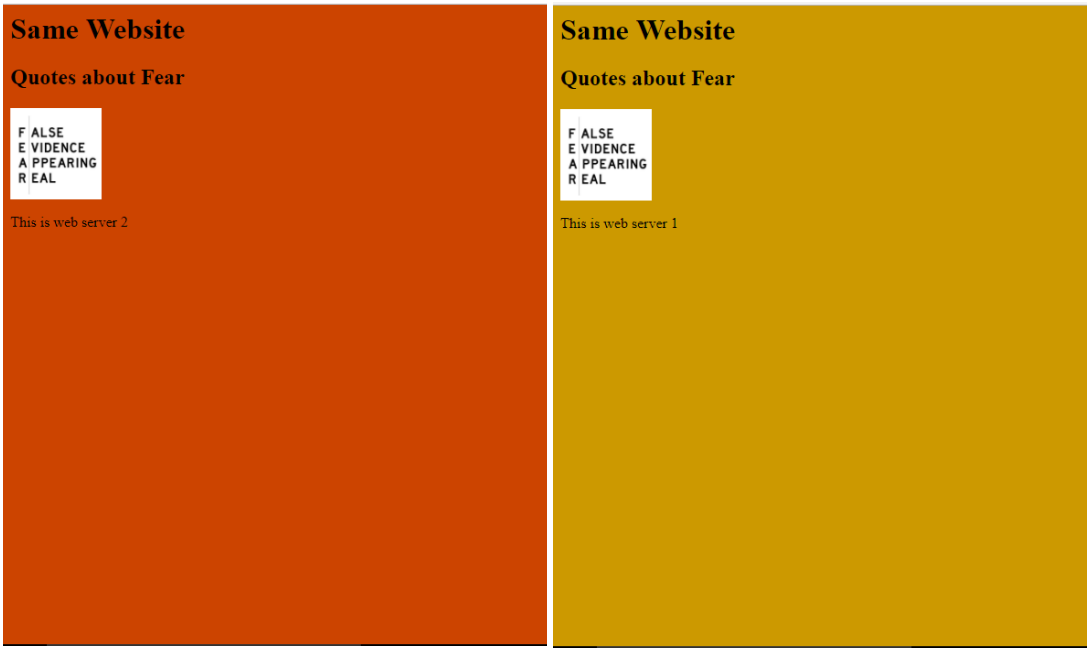
Step 14: Now click on your Load balancer which redirects to your ELB dashboard!





Now click on Description tab and notice the status: it takes at least 5 mins for the load balancer to register those 2 instances. If load balancer is ready you can see the status ‘in service’.

Now copy the DNS name without A record and paste it in your browser and refresh it multiples times to see the magic of ELB and how it routes internet traffic across two instances.



Step 15: Clean up!

Delete your instances and also your load balancer.

Questions to ponder:

1. **What does a load balancer do?**
2. **Can we set up a load balancer in multi AZ's or in multiple regions?**
3. **How does an ELB ensure health of instances ?**

Elastic Load Balancer:

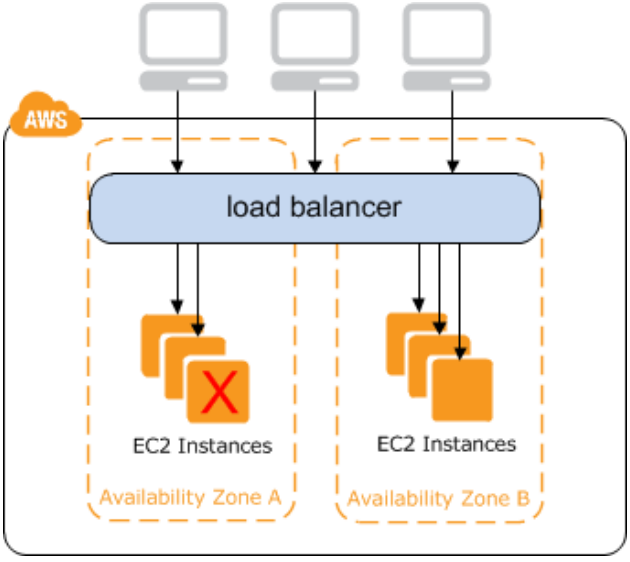
Elastic Load Balancing supports three types of load balancers:

1.Classic Load Balancers

2. Application Load Balancers.

3.Network load balancers

1.Classic Load balancer:



As you can notice from the figure EC2 instances in various availability zone can be handled by a single load balancer.

A classic load balancer serves as a single point of contact for clients. This increases the availability of your application.

You can **add and remove instances** from your load balancer, without disrupting the overall flow of requests to your application.

Elastic Load Balancing scales your load balancer as traffic to your application changes over time.

A *listener* checks for connection requests from clients, using the protocol and port that you configure, and forwards requests to one or more registered instances using the protocol and port number that you configure.

You add one or more listeners to your load balancer.

You can configure health checks, which are used to monitor the health of the of the registered instances so that the load balancer can send requests only to the healthy instances.

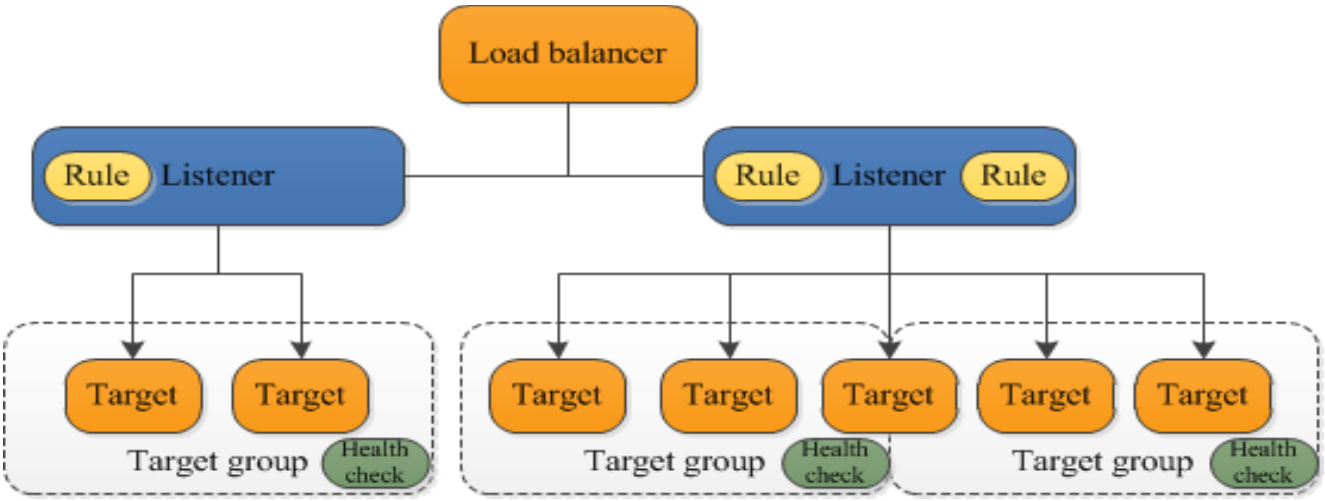
It is important to keep approximately the same number of instances in each Availability Zone registered with the load balancer.

For example, if you have ten instances in Availability Zone us-west-2a and two instances in us-west-2b, the requests are distributed evenly between the two Availability Zones. Thus, the two instances in us-west-2b serve the same amount of traffic as the ten instances in us-west-2a. Instead, you should have six instances in each Availability Zone.

2.Application load balancing:

An Application Load Balancer functions at the application layer of OSI model. The load balancer makes routing decisions based on the content of the application traffic in the HTTP messages.

The load balancer distributes incoming application traffic across multiple targets, such as EC2 instances, in multiple Availability Zones. This increases the fault tolerance of your applications. Elastic Load Balancing detects unhealthy targets and routes traffic only to healthy targets.



Architecture of Application load balancer:

The *load balancer* serves as the single point of contact for clients. You add one or more listeners to your load balancer.

A *listener* checks for connection requests from clients, using the protocol and port that you configure, and forwards requests to one or more target groups.

Each rule specifies a target group, condition, and priority. When the condition is met, the traffic is forwarded to the target group. You must define a default rule for each listener, and you can add rules that specify different target groups based on the content of the request (also known as content-based routing).

Each *target group* routes requests to one or more registered targets, such as EC2 instances, using the protocol and port number that you specify.

You can register a target with multiple target groups. You can configure health checks on a per target group basis. Health checks are performed on all targets registered to a target group that is specified in a listener rule for your load balancer.

Benefits of using an application load balancer:

1. Support for path-based routing. You can configure rules for your listener that forward requests based on the URL in the request.

   For eg: to the URL google.com/images and google.com/videos I can configure the listener to forward requests to those2 URL's based on my requirements

2. Support for containerized applications. Amazon EC2 Container Service (Amazon ECS) can select an unused port when scheduling a task and register the task with a target group using this port. This enables you to make efficient use of your clusters.

3. Support for monitoring the health of each service independently, as health checks are defined at the target group level and many CloudWatch metrics are reported at the target group level. Attaching a target group to an Auto Scaling group enables you to scale each service dynamically based on demand.

4. Access logs contain additional information and are stored in compressed format.

5. Improved load balancer performance.

Network Load Balancer :

A Network Load Balancer functions at the fourth layer of the Open Systems Interconnection (OSI) model. It can handle millions of requests per second. After the load balancer receives a connection request, it selects a target from the target group for the default rule. It attempts to open a TCP connection to the selected target on the port specified in the listener configuration.

When you enable an Availability Zone for the load balancer, Elastic Load Balancing creates a load balancer node in the Availability Zone. Each load balancer node for your Network Load Balancer distributes traffic across the registered targets in its Availability Zone only. If you enable multiple Availability Zones for your load balancer, this increases the fault tolerance of your applications. For example, if all targets in one Availability Zone are unhealthy, we remove the IP address associated with the subnet from DNS, but the load balancer nodes in the other Availability Zones are still available to route traffic. If a client doesn't honor the time-to-live (TTL) and sends requests to the IP address after it is removed from DNS, the requests fail.

Bottomline:

Classic Load Balancer 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, microservices, 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

You can create up to twenty (20) load balancers per region, which includes both Classic and Application Load Balancers and should you require more you can request Amazon Tech support.