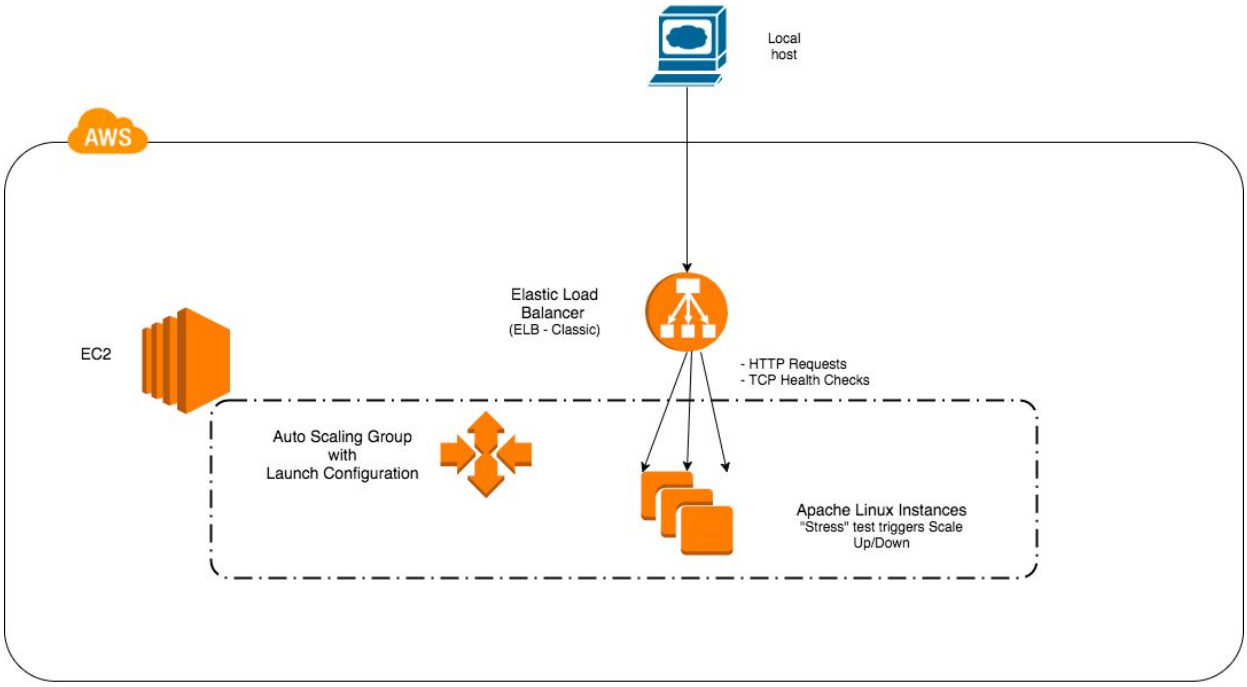
Lab Environment

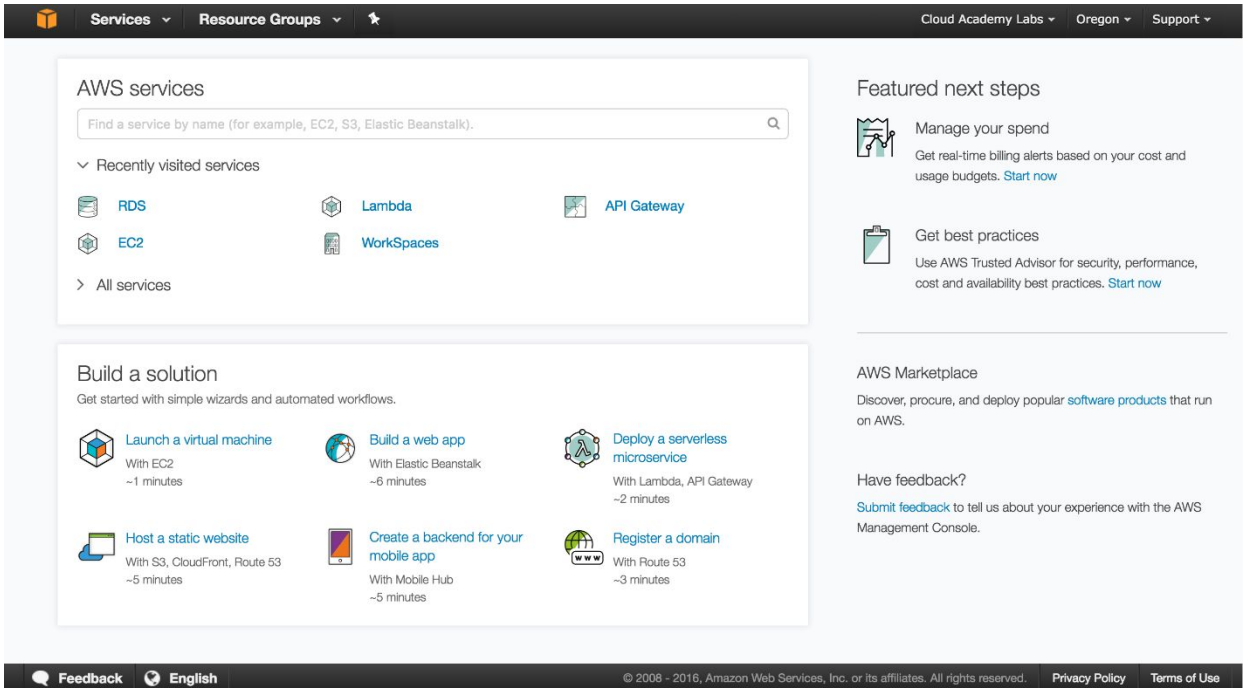
After completing the lab instructions the environment will look similar to:



Logging in to the Amazon Web Services Console

Introduction

This Lab experience involves Amazon Web Services, and you will use the AWS Management Console to complete all the Lab Steps. Please note that you will have a space storage limit of 100GB for this lab, which will be more than sufficient to complete it.



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.

Instructions

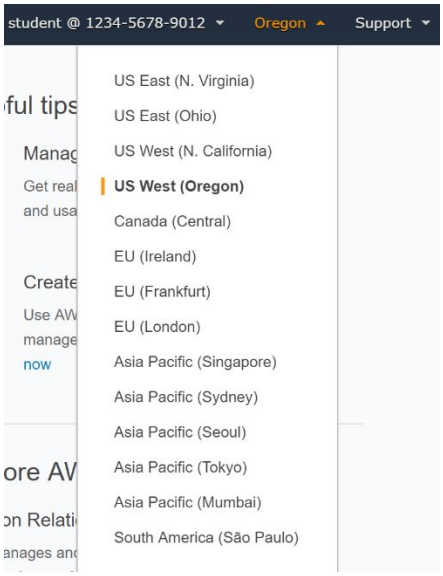
1. To start the Lab experience, open the Amazon Console by clicking this button:

Open AWS Console

2. Enter the following credentials created just for your Lab session, and click **Sign In**:

- **Account ID or alias**: Keep the pre-populated value
- **IAM user name**: *student*
- **Password**: *$passw0rd*

3. Select the **US West (Oregon)** region using the upper right drop-down menu on the AWS Management Console:



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 **US West (Oregon)** for this Lab.

Create a load balancer using ELB

Introduction

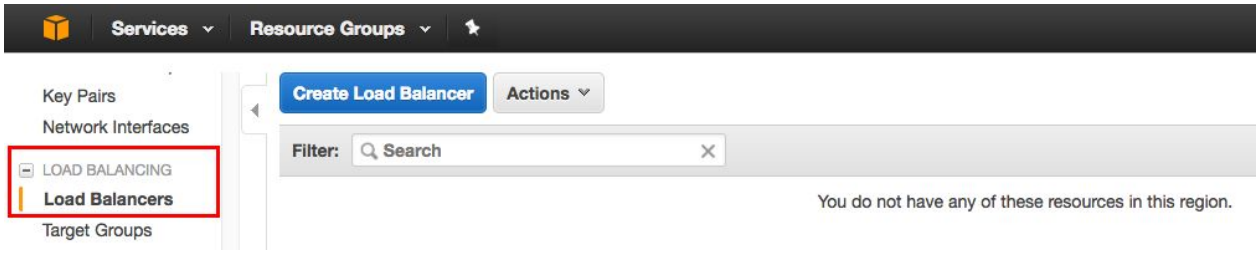
Elastic Load Balancers automatically distribute incoming application traffic across multiple Amazon EC2 instances. It enables you to achieve greater fault tolerance in your applications and seamlessly provide the correct amount of load balancing capacity needed in response to incoming application requests.

ELB detects unhealthy instances within a pool and automatically reroutes traffic to healthy instances until the unhealthy instances have been restored. Elastic Load Balancers can be enabled within a single Availability Zone or across multiple zones for greater consistent application performance.

Instructions

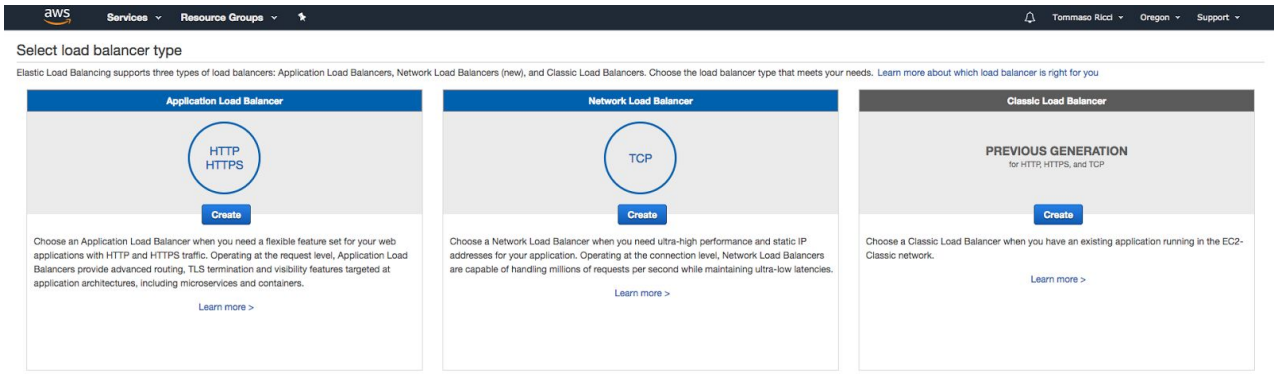
1. Select **EC2** from the AWS **Services** menu.

2. In the left pane, click **Load Balancers** in the **Load Balancing** section:



3. Click the **Create Load Balancer** button.

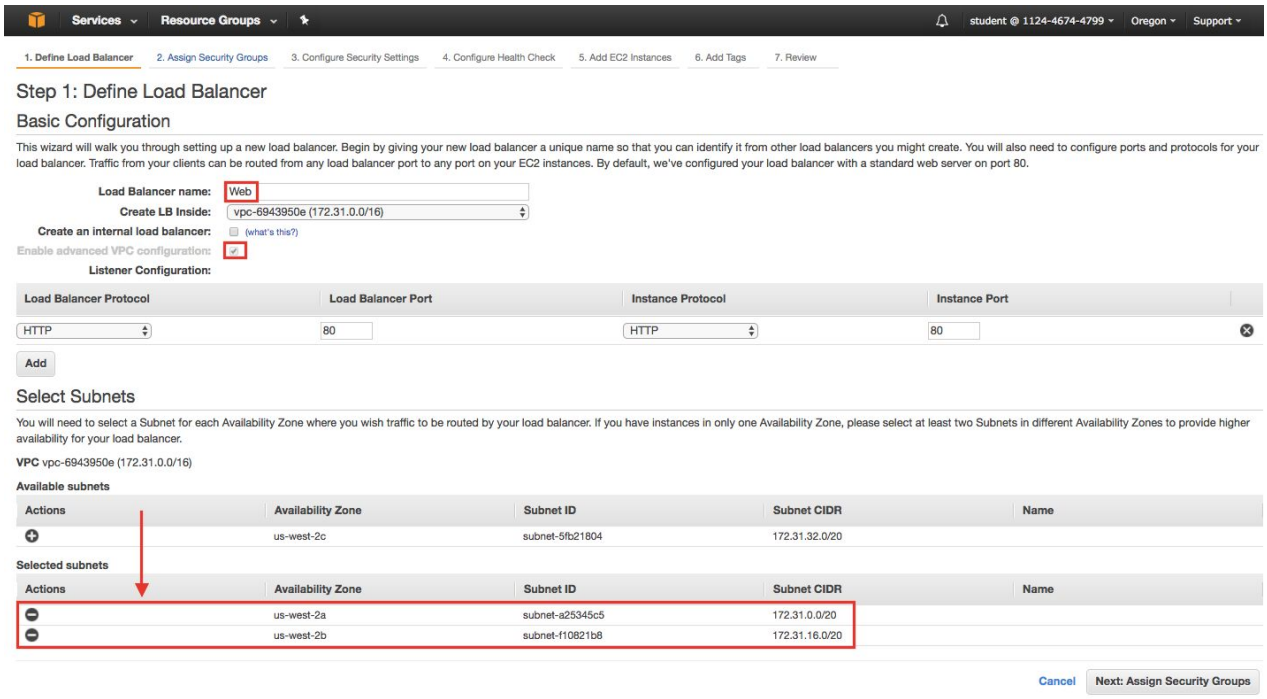
4. Take a moment to read the information for both load balancer types. Click **Create** in the Classic Load Balancer square:



A multi-step wizard starts for creating a load balancer.

5. On the **Define Load Balancer** screen, configure it in the following manner:

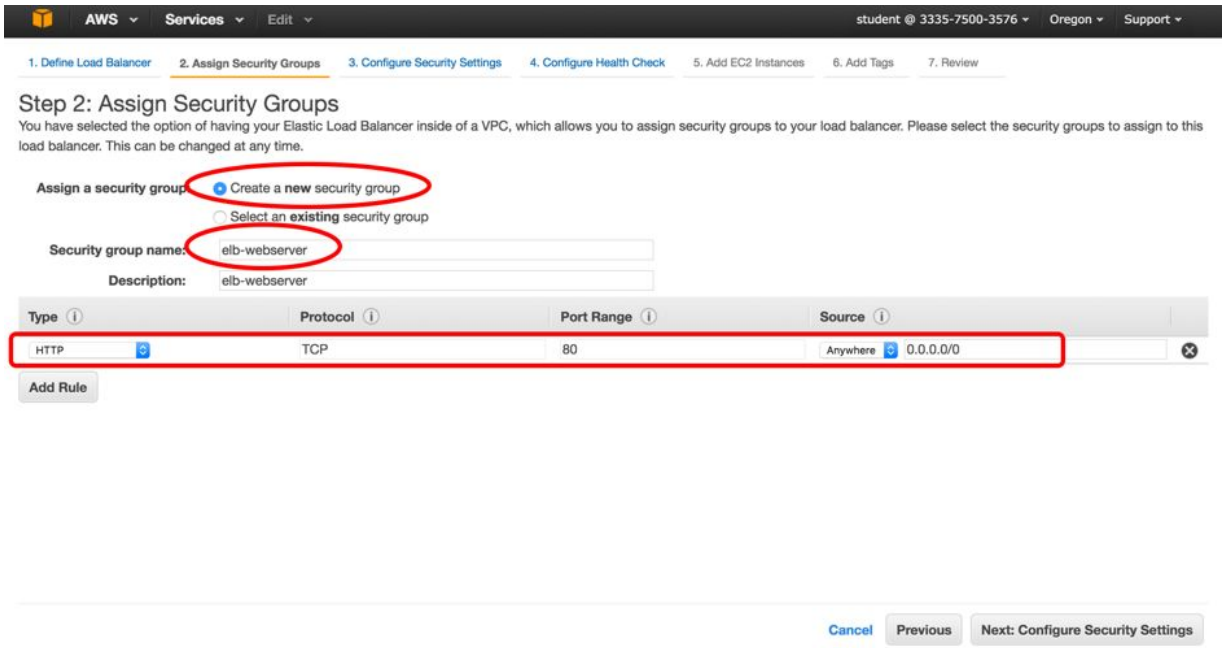
- **Load Balancer name**: Enter *Web*
- **Create LB Inside**: Select the vpc-*<AlphaNumeric>* from the drop-down menu. Note the CIDR notation for the VPC. (For example, 172.31.0.0/16)
- **Enable advanced VPC configuration:** This field should be checked for you. If not, please check the box.
- **Listener Configuration:** Leave the default values (HTTP/80/HTTP/80)
- **Select Subnets > Available subnets:** Click the **+** action icon to select **us-west-2a** *and* **2b Availability Zones.**



Notice when you select an **Available subnet** it transfers to the **Selected subnets** section. Click **Next: Assign Security Groups** when ready.

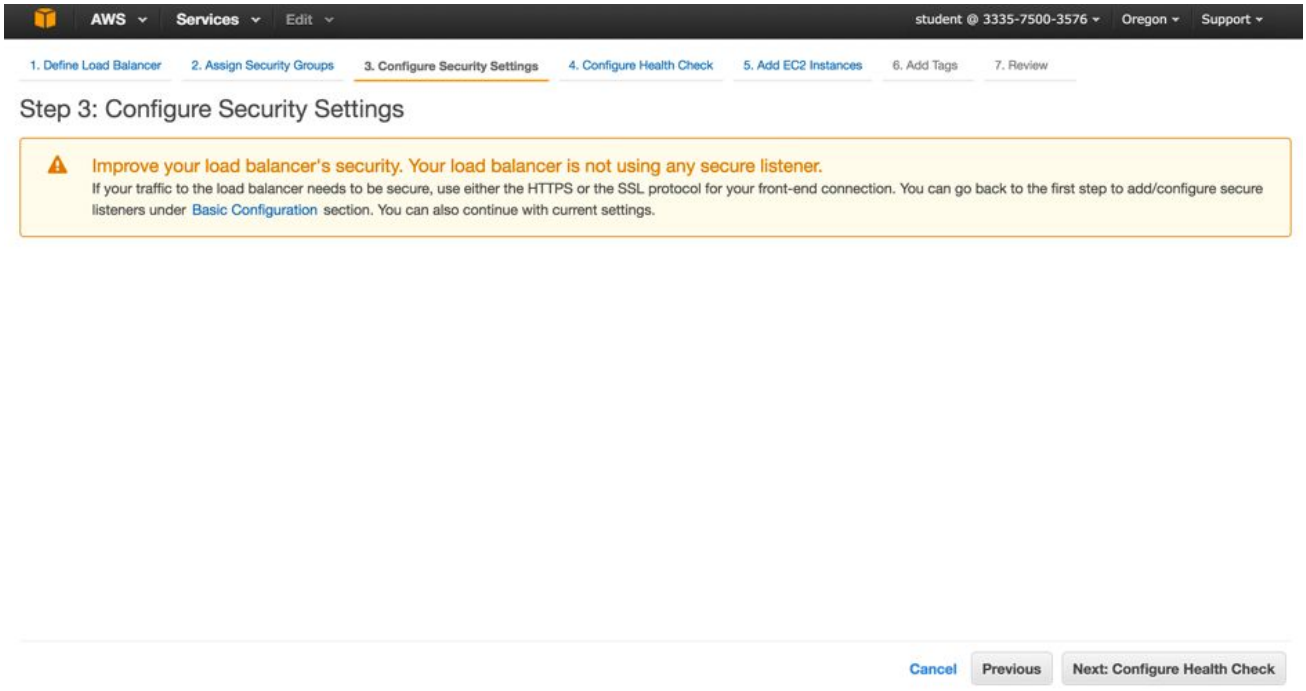
6. On the **Assign Security Group**s page:

- Select the **Create a new security group** radio button
- Enter *elb-webserver* for the **Security group name**
- Enter *ELB for a webserver cluster* for the **Description**
- Create a single firewall rule of **Type HTTP**; **Protocol TCP**; **Port Range 80** and **Source Anywhere** (0.0.0.0/0).



This will rule will allow inbound HTTP traffic on port 80 from any source IP address.
Click **Next: Configure Security Settings** when ready.

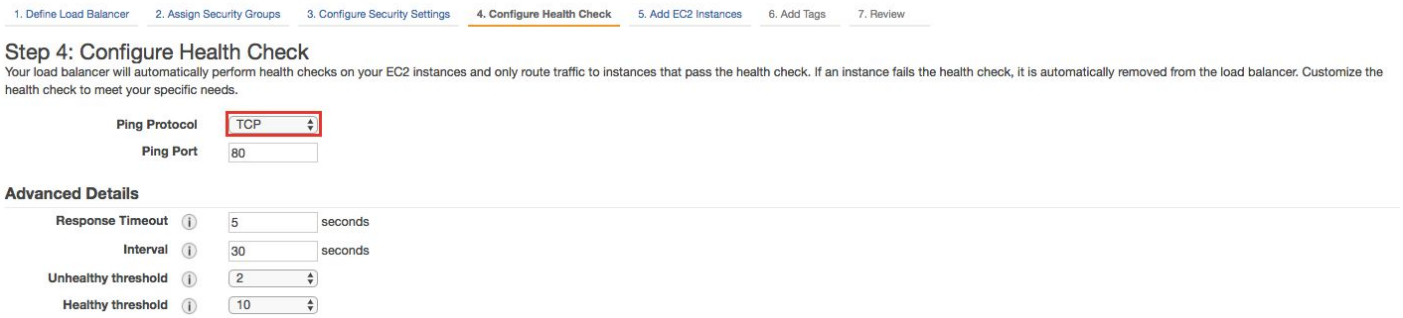
7. Ignore the warning in the **Configure Security Settings** page. You are only serving the HTTP protocol in this lab. Security is not a concern (implemented via HTTPS or SSL protocols) so these settings are not required:



Click **Next: Configure Health Check**.

8. On the **Configure Health Check** page:

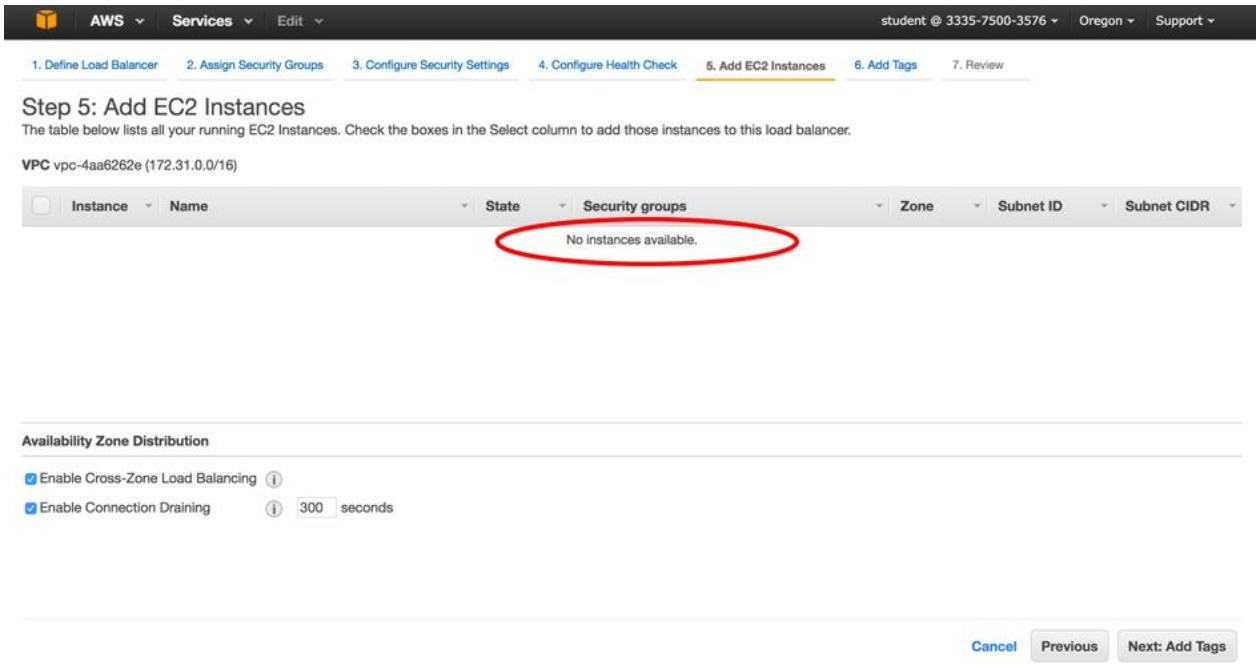
- **Ping Protocol:** Select **TCP**
- **Ping Port**: *80*



In a production environment you would likely set this to HTTP. However, this Lab is not concerned with how well your web server is doing serving HTML pages, it just wants to make sure it is up and running. Again, set the **Ping Protocol** to **TCP**.

Click **Next: Add EC2 Instances** when ready to proceed (the defaults for the advanced section will suffice).

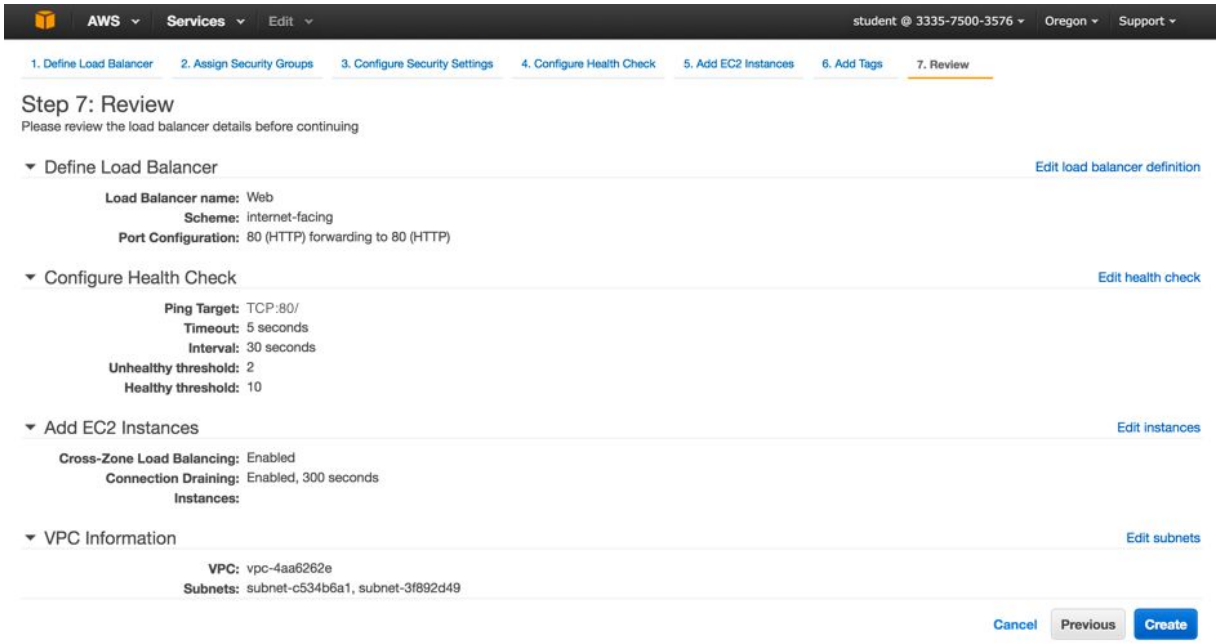
9. On the **Add EC2 Instances** page of the wizard, notice the "No instances available" message:



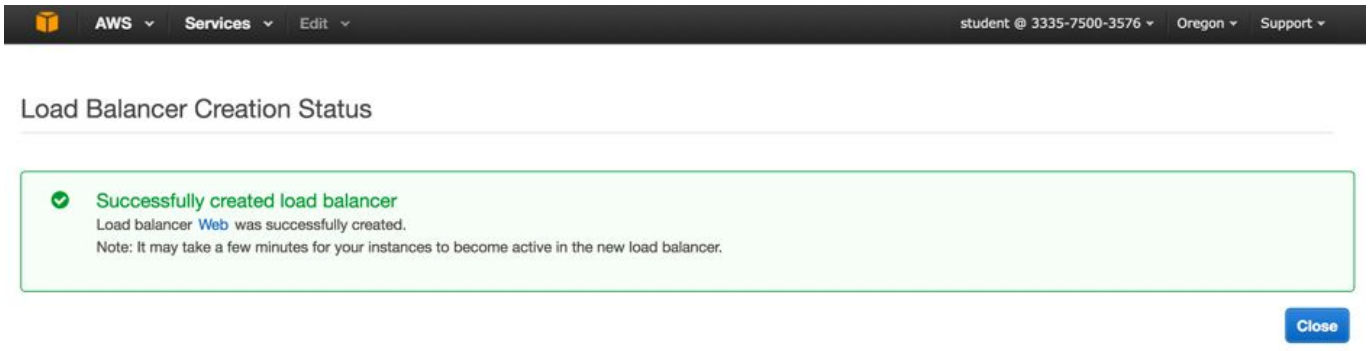
The message is because you have not created and launched an Auto Scaling Group yet, which is used to launch and maintain one or more instances.

10. Click **Next: Add Tags** to continue. Leave the fields blank in the **Add Tags** page. Click **Review and Create**.

11. Review your settings for correctness and then click **Create** when ready:



12. Wait for the **Load Balancer Creation Status** to display a successful creation message:



13. Click **Close**.

Summary

ELB is ready to service requests! In this Lab Step you created an Elastic Load Balancer (ELB) to service HTTP requests on port 80 from any IP source address. This ELB will be used on the front-end to direct requests to several instances running a web server. This is a very common use case for ELBs.

Create a Launch Configuration

Introduction

A Launch Configuration is a template that the Auto Scaling group uses to launch Amazon EC2 instances. If you've launched an individual EC2 instance before, you've already walked through the process of defining compute characteristics such as the instance type, security groups, and configuration scripts. A launch configuration allows you to define these same characteristics, which are then applied to any instances launched in the Auto Scaling group that references the Launch Configuration. The Launch Configuration essentially contains the blueprint or DNA for the exact type of instance that should be launched. Hence, when auto scaling, each instance is guaranteed to be just like the last one. It's repeatable, scalable, and reliable.

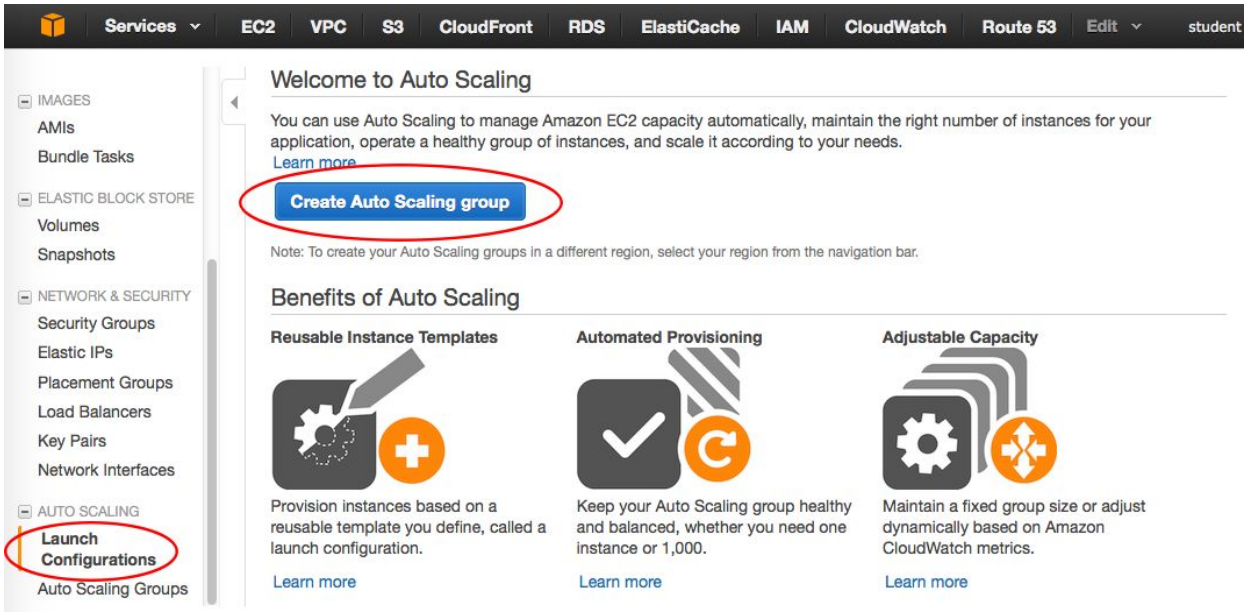
When you create the Launch Configuration you will include information such as the Amazon machine image ID (AMI) to use for launching the EC2 instance, the instance type, key pairs, security groups, and block device mappings, among other configuration settings. When you create your Auto Scaling group, you must associate it with a Launch Configuration. You can attach only one Launch Configuration to an Auto Scaling group at a time and it cannot be modified.

First you will create a Launch Configuration, then the Auto Scaling group.

Instructions

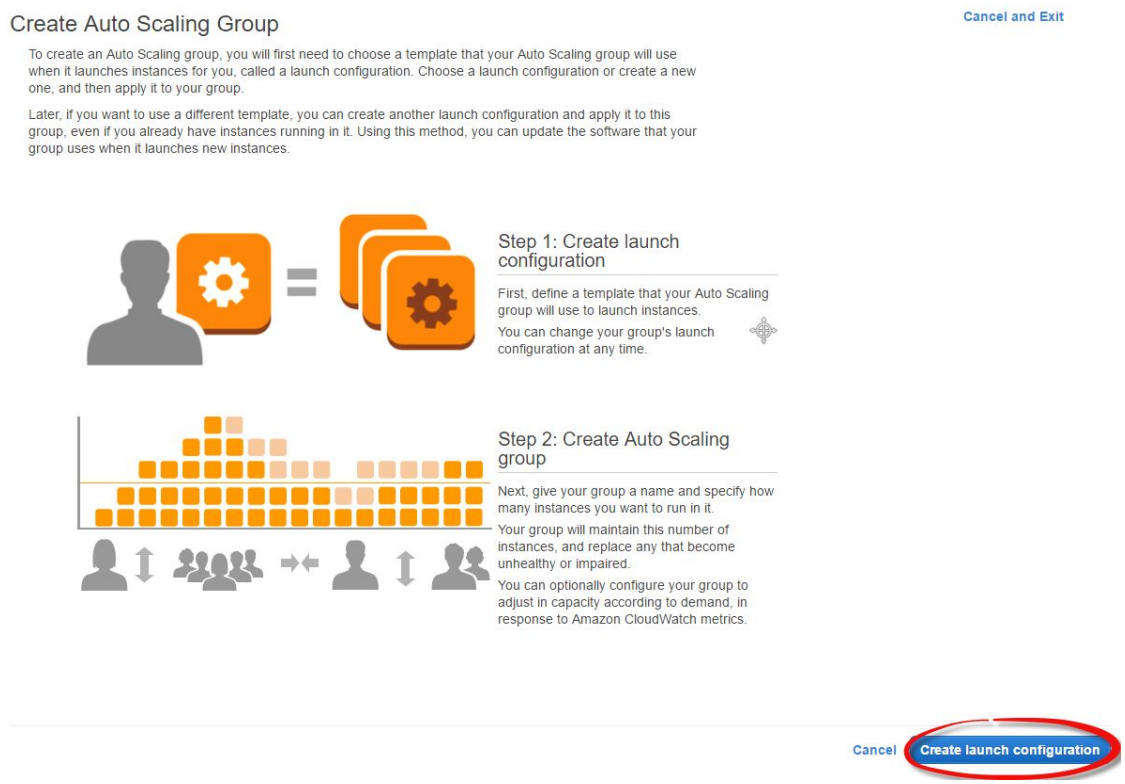
1. Navigate to the **EC2** service from the AWS dashboard:

2. Open the **Launch Configurations** page and click the **Create Auto Scaling group** button:



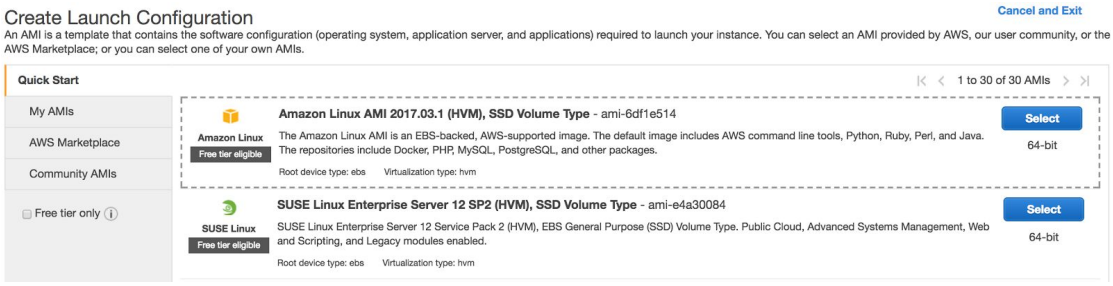
The Create Auto Scaling group wizard starts.

3. Take a moment to read the helpful **Create Auto Scaling Group** information that is part of the wizard. *Tip*: Reading the information within various AWS wizards helps with the learning curve and is a good habit to get into:



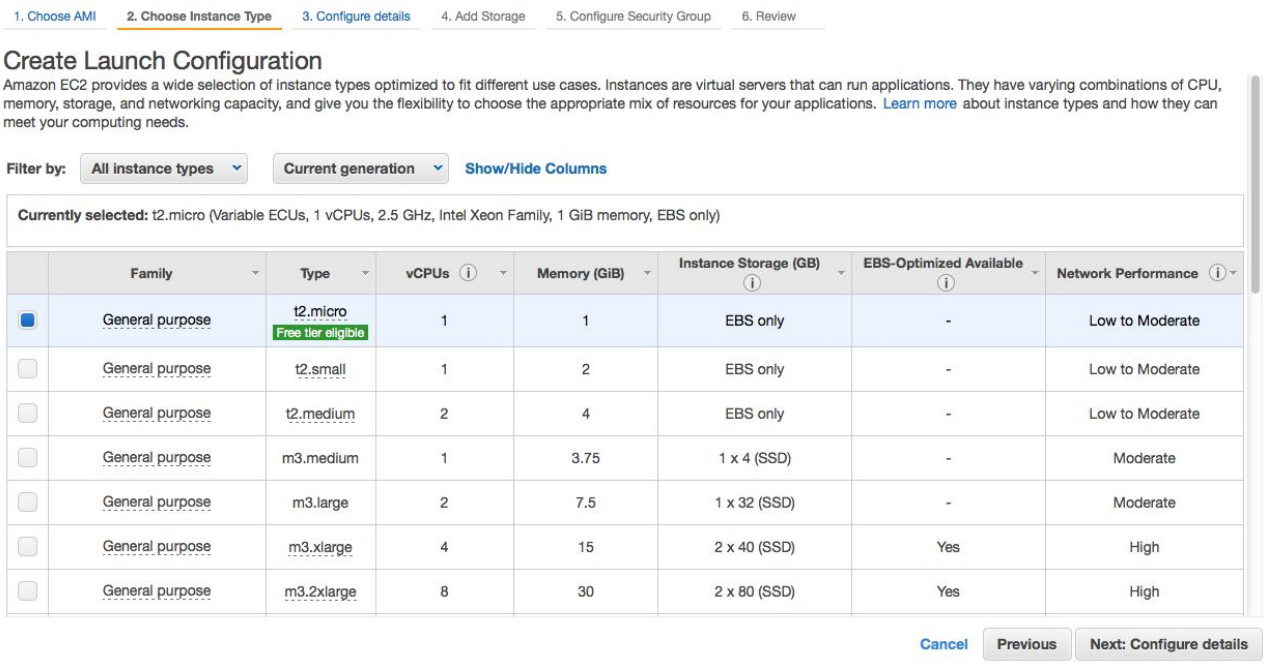
Click the **Create Launch configuration** button. Creating a launch configuration is the first step in creating a Auto Scaling group. The wizard guides you through each required step and displays a graphical interface that is similar to the Launch Instance wizard.

4. On the **Choose AMI** page of the wizard you must select the AMI that will be used by all the EC2 instances of the Auto Scaling group. Select the Amazon Linux AMI:



Click **Select** when ready. The next step is choosing the instance type.

5. On the **Choose Instance Type** page, select the **t2.micro type** and click **Next: Configure details:**



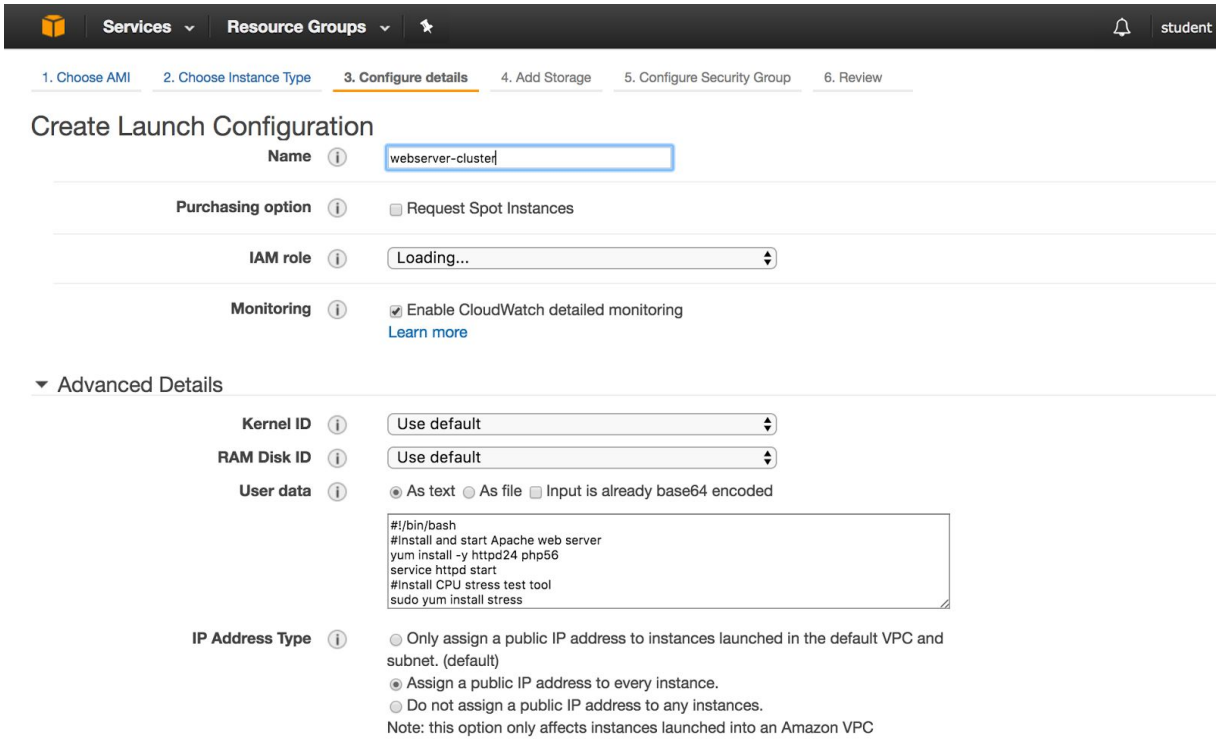
6. On the **Configure details** page:

- Enter *webserver-cluster* for the **Name**
- Check **Enable CloudWatch detailed monitoring**
- Expand **Advanced Details**

- Select the **Assign a public IP address to every instance** radio button
- Paste the following bash snippet in the **User data** field:

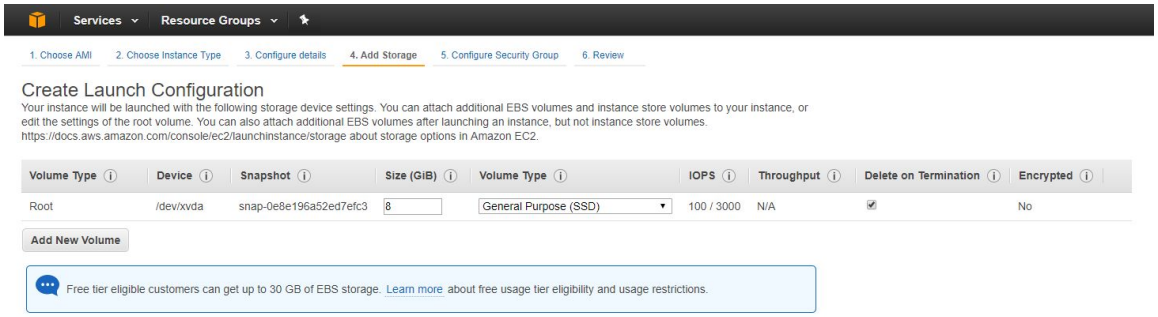
```
#!/bin/bash
#Install and start Apache web server
yum install -y httpd24 php56
service httpd start
touch /var/www/html/index.html
chmod 644 /var/www/html/index.html
#Install CPU stress test tool
sudo yum install stress
```

The Configure details screen of the wizard should look similar to the following:



By default, CloudWatch monitors EC2 instances approximately every 5 minutes. Detailed monitoring enables monitoring more often (each minute). *Note*: Detailed monitoring does have an associated cost.Click **Next: Add Storage** when ready.

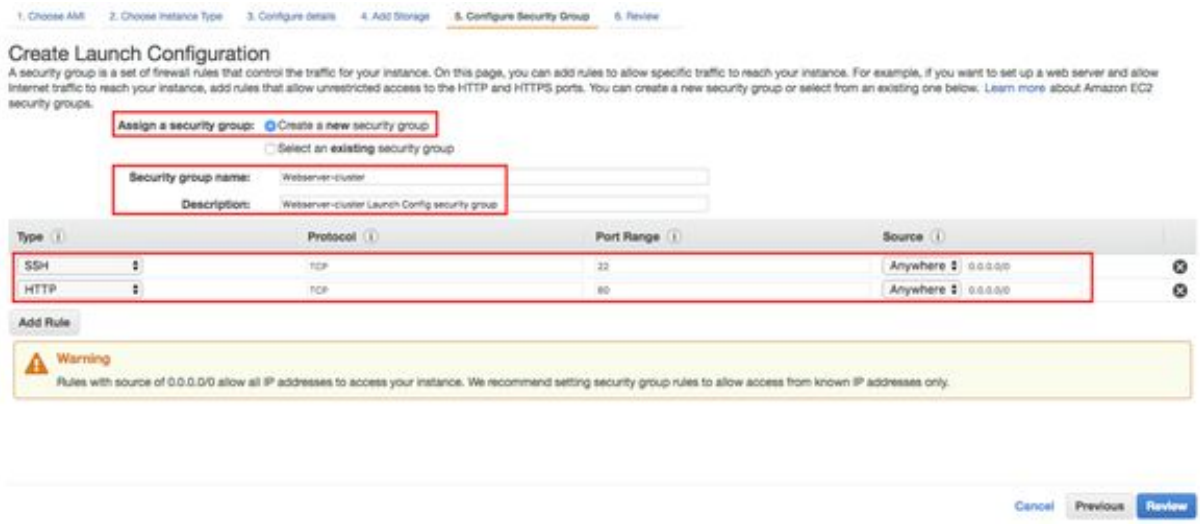
7. The **Add Storage** page of the wizard allows you to add or increment the size of any EBS volume attached to each EC2 instance started by the Auto Scaling group. Leave the defaults and do not add any EBS volumes.



Typically large EBS volumes are only needed if your software requires storage space to process the application data. Many applications store raw or processed data with Amazon S3, Redshift, DynamoDB or another storage/database service provided by Amazon. When that is the use-case, large EBS volumes are usually not required. This lab environment definitely does not need extra disk space. Click **Next: Configure Security** when ready.

8. On the **Configure Security Group** page there are several configurations for your Launch Configuration:

- Select **Create a new security group**
- Enter *Webserver-cluste*r for the **Name**. Enter a **Description** as well.
- Add two rules. For the first rule configure:
    - **Type**=SSH
    - **Protocol**=TCP
    - **Port Range**=22
    - **Source**=Anywhere (0.0.0.0/0) *Warning*: In production the source should be more restrictive to account for corporate security policies. For example, your corporate external public IP range.
- For the second rule configure:
    - **Type**=HTTP
    - **Protocol**=TCP
    - **Port Range**=80
    - **Source**=Anywhere (0.0.0.0/0) *Warning*: In production the source should be more restrictive. For example, only the ELB should be able to connect to port 80 on the web servers. Then the ELB allows remote access from anywhere, but is the only component that can access the instances directly in the auto scaling group.



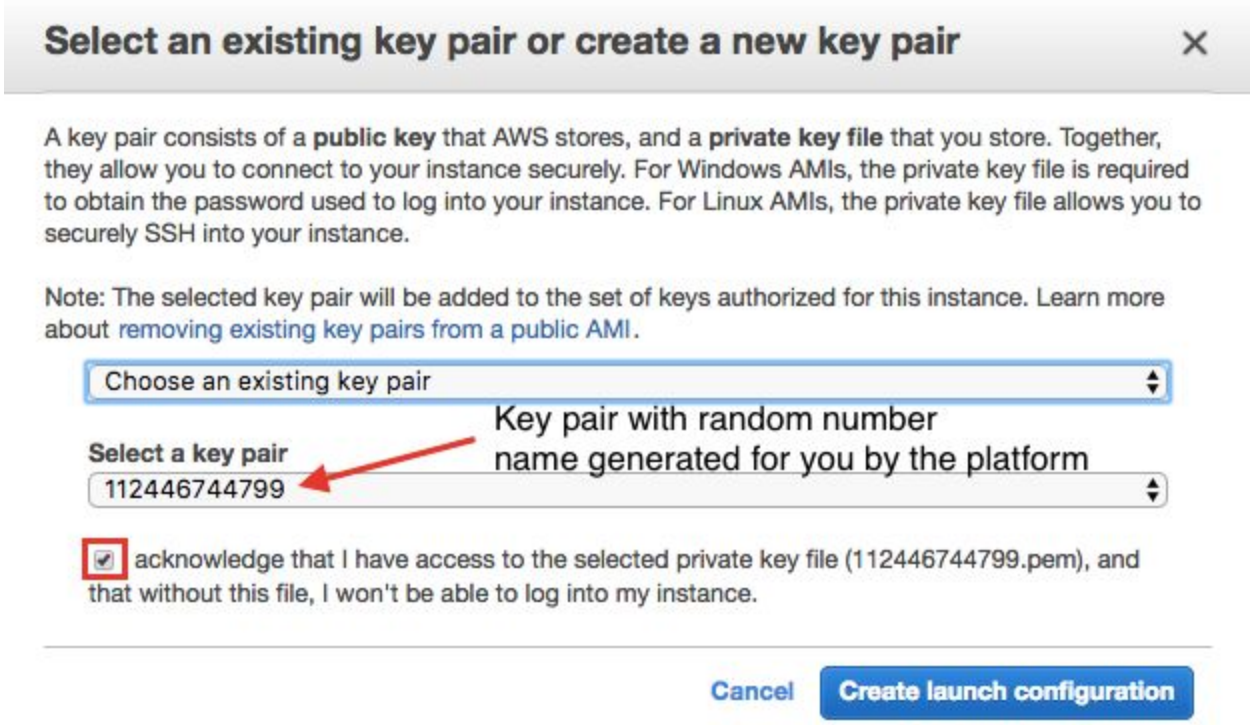
Click **Review** when ready.

9. Once you have reviewed the details for accuracy, click **Create launch configuration**:

You will be presented with the **Select an existing key pair or create a new key pair** dialogue. Notice that you will use this key pair to access all the instances that are going to be launched by the Auto Scaling service with this Launch Configuration. Always be sure to secure your key pair. Not doing so is a security risk.

10. In the **Select an existing key pair or create a new key pair** dialog:

- Select **Choose an existing key pair** from the first drop-down menu. If you did not need access to the instances, you could select **Proceed without a key pair** and acknowledge you will not be able to access the instance. (However, you will need SSH access later.)
- **Select a key pair**: Select the random number named key pair that is generated for you by the Cloud Academy platform
- Check the "I acknowledge that I have access to the selected private key file..." check box:



Click **Create launch configuration** when ready to proceed.

Because the normal flow after creating a Launch configuration is to create an Auto Scaling group that will use the configuration, the next screen will present you with the option to do so. However, click **Cancel** as you will navigate directly in the console menus and create it from scratch in the next Lab Step.

Summary

You have created a Launch Configuration that can be used by an Auto Scaling group to launch identical instances every time. Note that you cannot modify a Launch Configuration. Why? It would impact the effectiveness and very purpose of having a launch configuration. For example, if you have multiple instances starting and terminating in accord with your scaling policy, then change the launch configuration, future instances would be different than the current production instances. This can be a nightmare to maintain and troubleshoot. (You can however create new Launch Configurations and have the Auto Scaling group associate with a new Launch Configuration.)

Create an Auto Scaling Group

Introduction

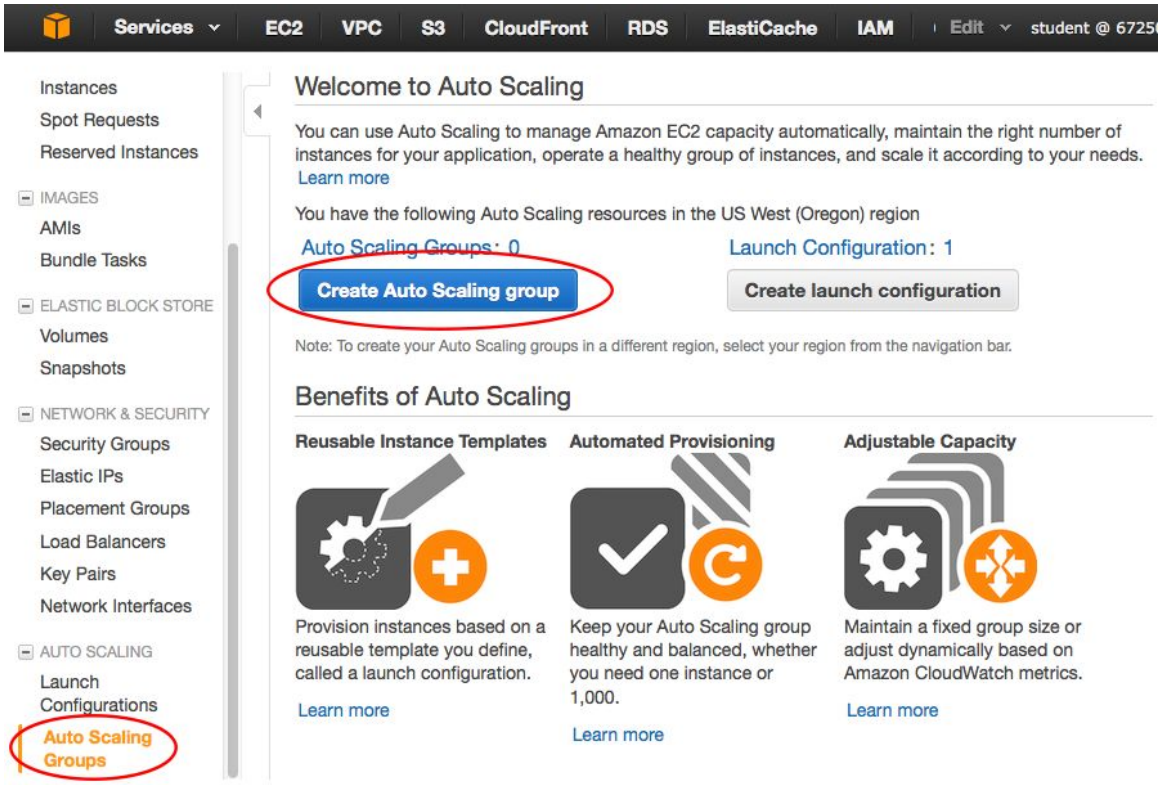
An Auto Scaling group is a representation of multiple Amazon EC2 instances that share similar characteristics and that are treated as a logical grouping for the purposes of instance scaling and management. For example, if a single application operates across multiple instances, you might want to increase or decrease the number of instances in that group to improve the performance of

the application. You can use the Auto Scaling group to automatically scale the number of instances or maintain a fixed number of instances. You create Auto Scaling groups by defining the minimum, maximum, and desired number of running EC2 instances the group must have at any given point of time.

An Auto Scaling group starts by launching the minimum number (or the desired number, if specified) of EC2 instances and then increases or decreases the number of running EC2 instances automatically according to the conditions that you define. Auto Scaling also maintains the current instance levels by conducting periodic health checks on all the instances within the Auto Scaling group. If an EC2 instance within the Auto Scaling group becomes unhealthy, Auto Scaling terminates the unhealthy instance and launches a new one to replace the unhealthy instance. This automatic scaling and maintenance of the instance in an Auto Scaling group is the core value of the Auto Scaling service. It's what puts the "elastic" in EC2.

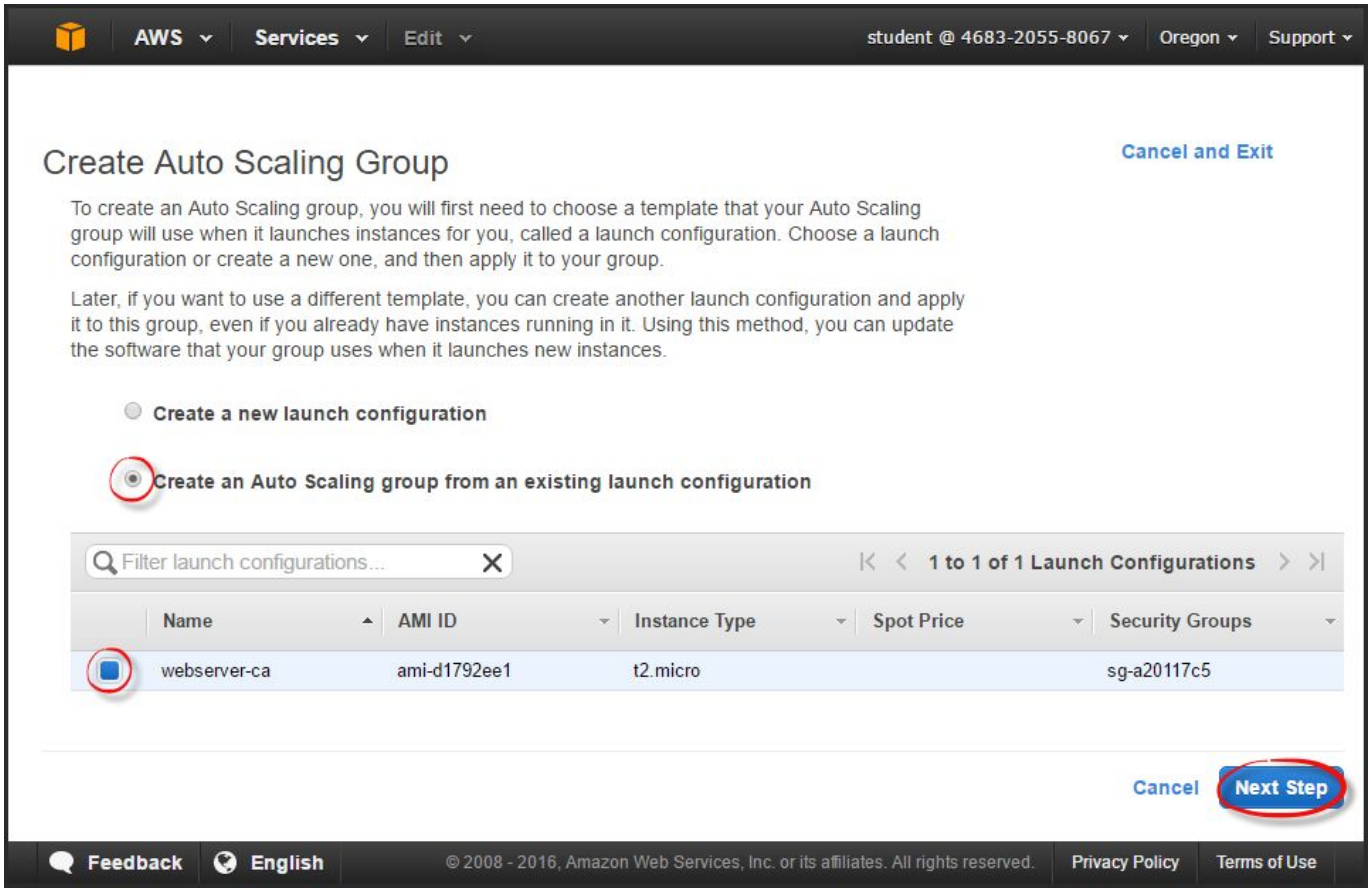
Instructions

1. Click **Auto Scaling Groups** in the left pane of the EC2 console, then click **Create Auto Scaling group:**



Since you created a Launch configuration earlier, you will be able to associate it with the Auto Scaling group you create here.

2. Select **Create an Auto Scaling group from an existing launch configuration.** Select the previously created launch configuration and then click **Next Step**:

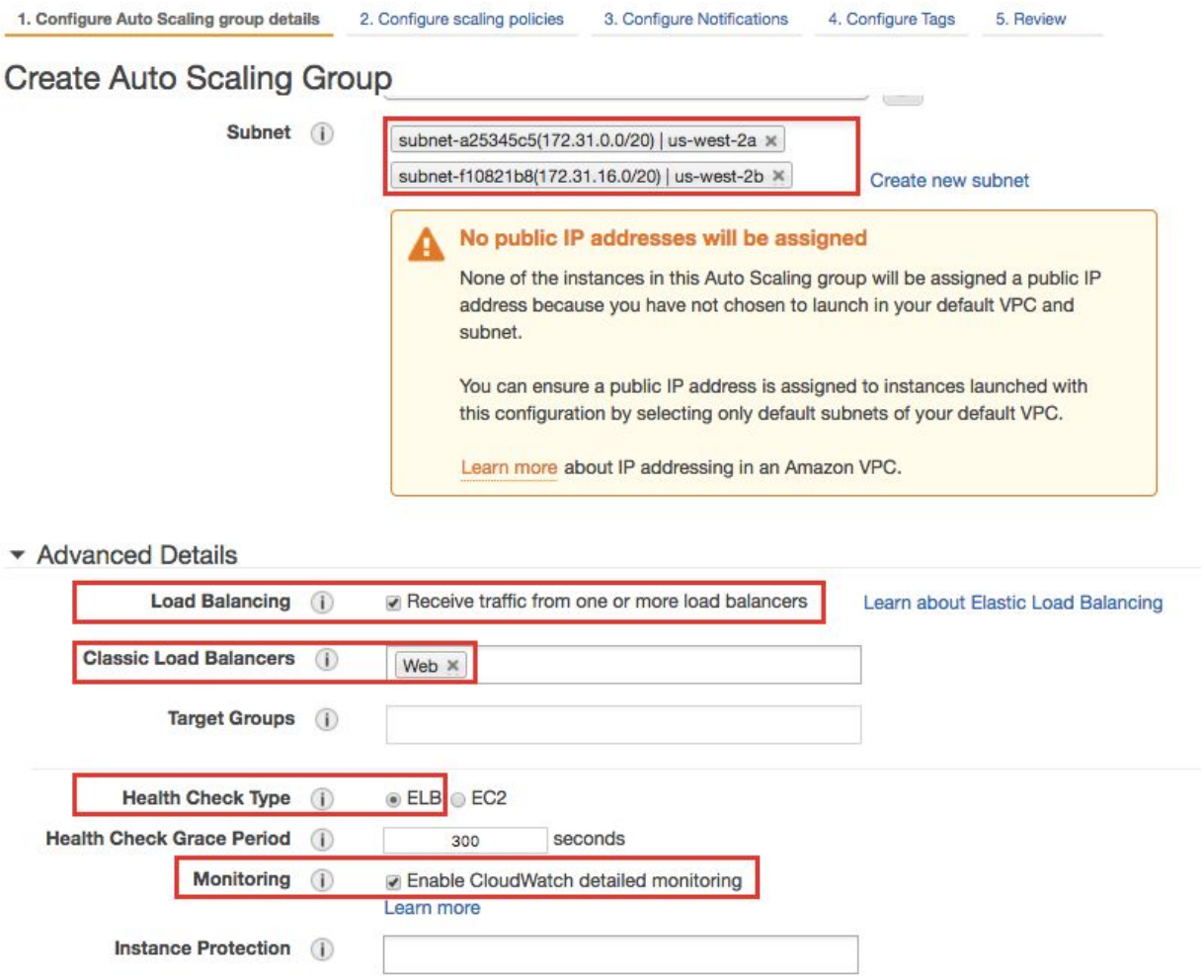


3. On the **Configure Auto Scaling group details** page, use the following settings:

- **Group name**:*webserver-cluster*
- **Group size**: Start with *1* instance
- **Network**: Select the **vpc-*<AlphaNumeric>*** from the drop-down menu
- **Subnet**: Select two subnets, onein *us-west-2a* and the other in *us-west-2b*.

While still on the first page of the wizard, expand **Advanced Detail**s, use the following settings:

- **Load Balancing**: Check **Receive traffic from one or more load balancers**
- Select **Web** for the **Classic Load Balancer** (the ELB you created earlier).
- **Health Check Type**: ELB
- **Monitoring**: Check **Enable CloudWatch detailed monitoring**

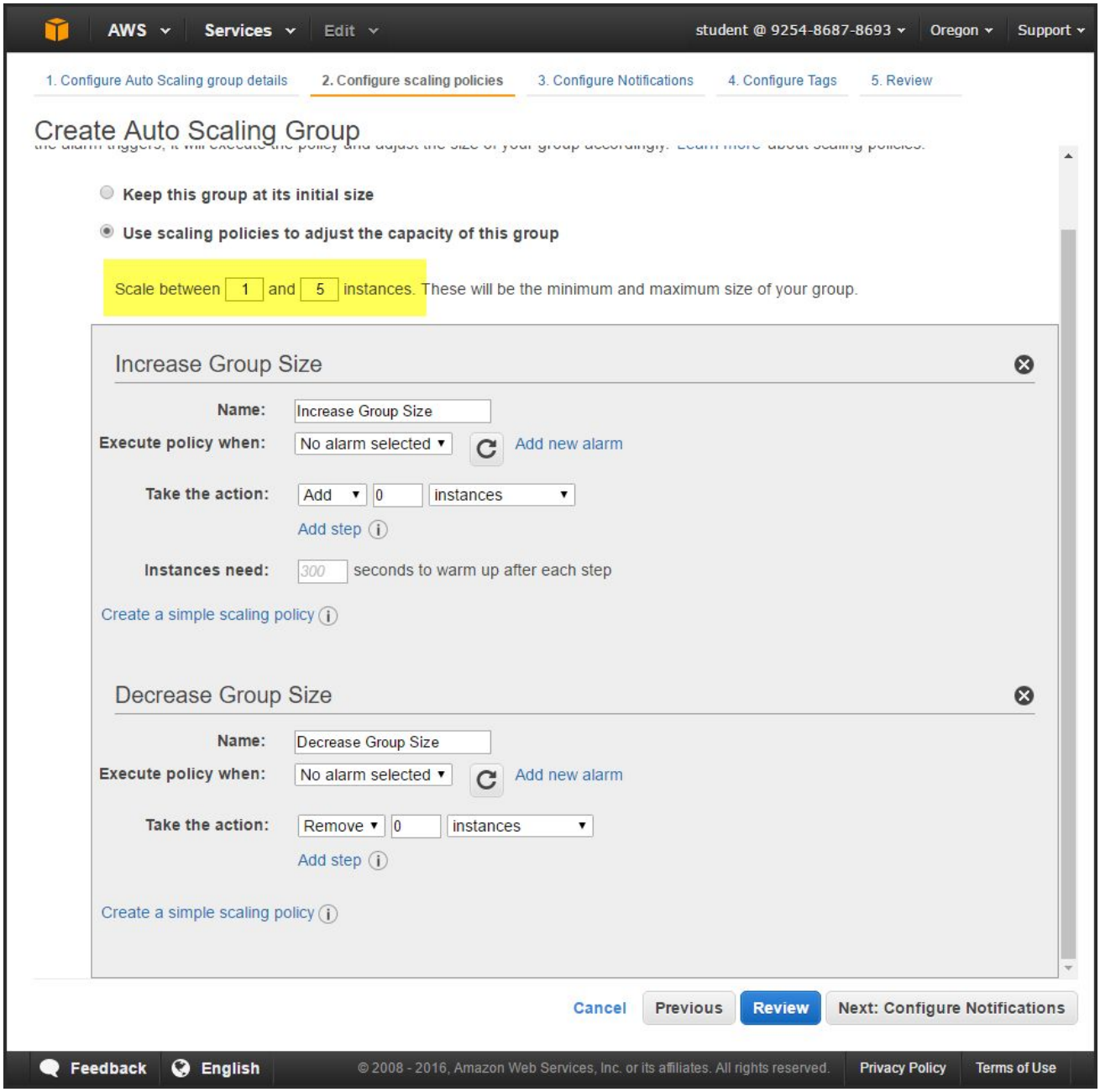


It is OK to ignore the "No Public IP addresses ..." warning message. Other settings can be left at the default values.

Click **Next: Configure scaling policies** when ready. Scaling policies determine how and when your infrastructure will scale up and scale back down.

4. In the **Configure scaling policies page**, select the **Use scaling policies to adjust the capacity**

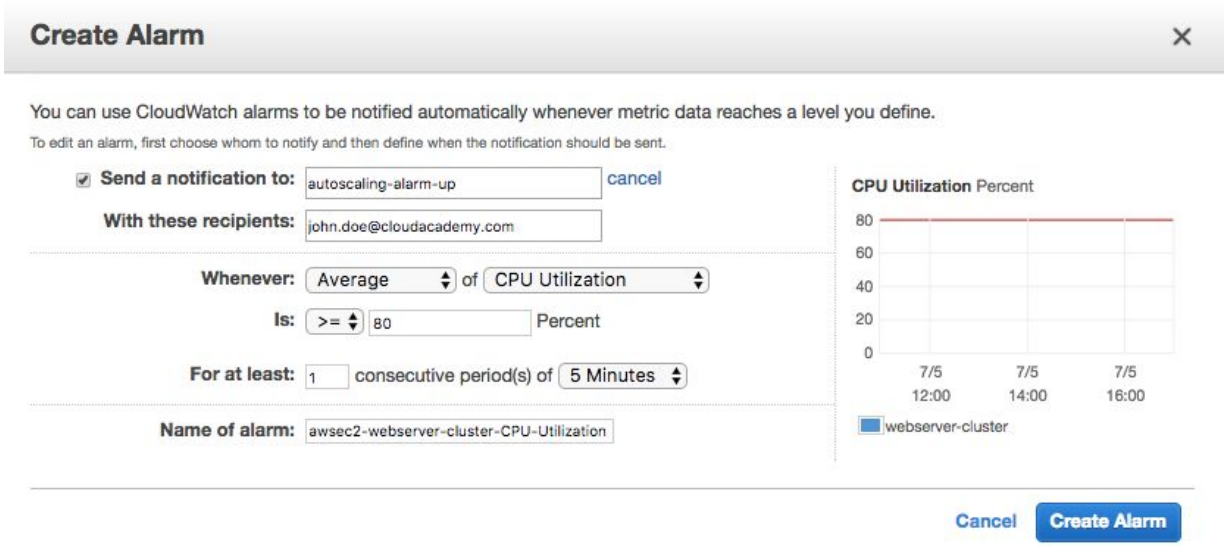
of this group radio button. Set it to scale between 1 and 5 instances:



The Auto Scaling group policies allow you to automatically increase or decrease the group size based upon policies you define. In order to establish an **Increase Group Size** or **Decrease Group Size** policy, you must create a CloudWatch Alarm and then define which action should be taken if it is triggered. *Don't* go to the next page of the wizard yet.

5. Click **Add new alarm** under the **Increase Group Size** section. A **Create Alarm** dialogue is

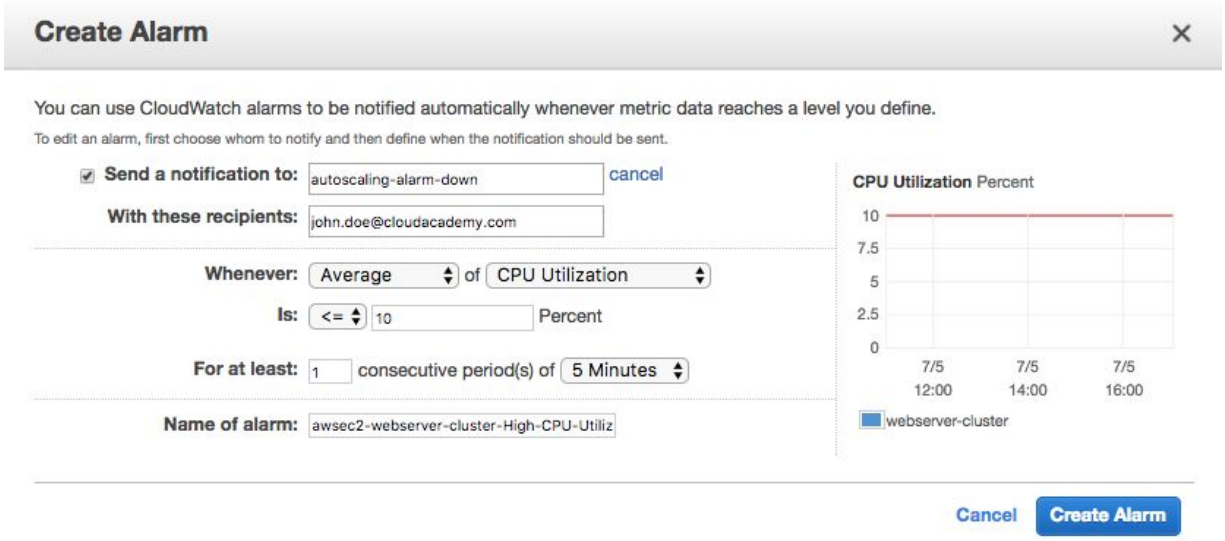
displayed. You will configure this for a alarm-up action:



If you want to receive a notification when the alarm is triggered, you need to set up an Simple Notification Service (SNS) topic.

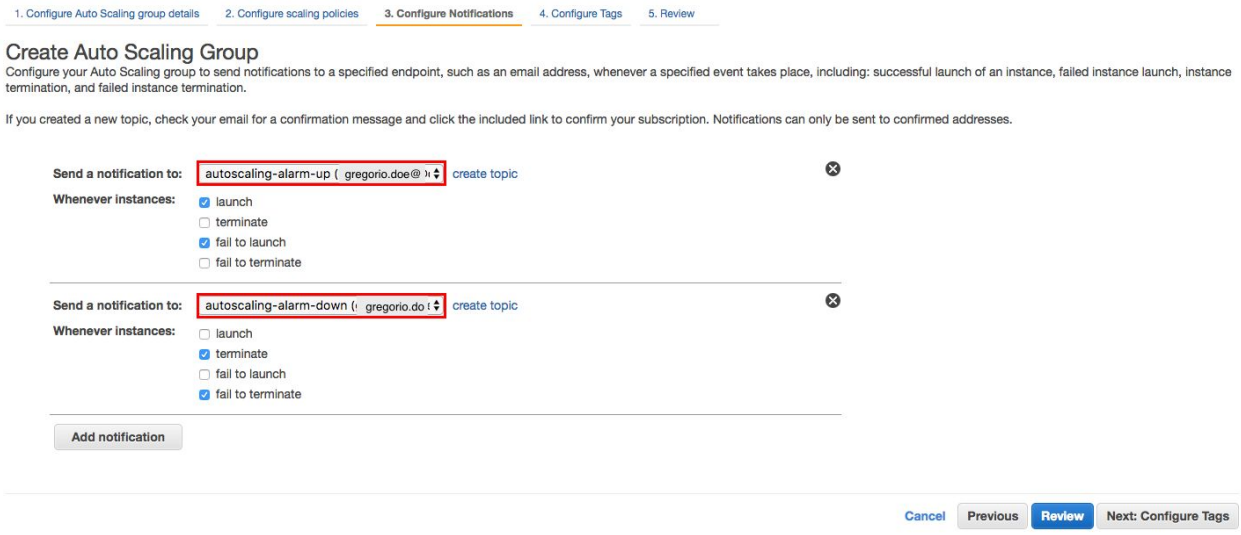
- Check the **Send a notification** checkbox
- Click **create topic.** Enter *autoscaling-alarm-up* for the SNS topic name and then enter your valid email address in the recipients field.
- Set **Whenever** to **Average** and **CPU Utilization** as the type of metric
- Set the relationship to **>=** and enter *80* as the **Percent** (Note this will be shown as a horizontal line at 80 in the adjoining graph soon.)
- Set **For at least** to *1* consecutive period of **5 Minutes**
- Enter a name for the alarm (or use the default name), and then click **Create Alarm**

6. Create another alarm for the **Decrease Group Size**. Name it appropriately (autoscaling-alarm-down). Configure similar to the last alarm, but this time set it to **<= 10 Percent of CPU Utilization**. Although not critical, the default **Name of alarm** is misleading in this case. Change the "High" within the default string to "Low". The **Increase/Decrease Group Size** sections should look similar to the following:



Click **Next: Configure Notifications** when ready. **Configure Notifications** will set you up so you are notified whenever an Auto Scaling Group instance is launched or terminated, with or without success.

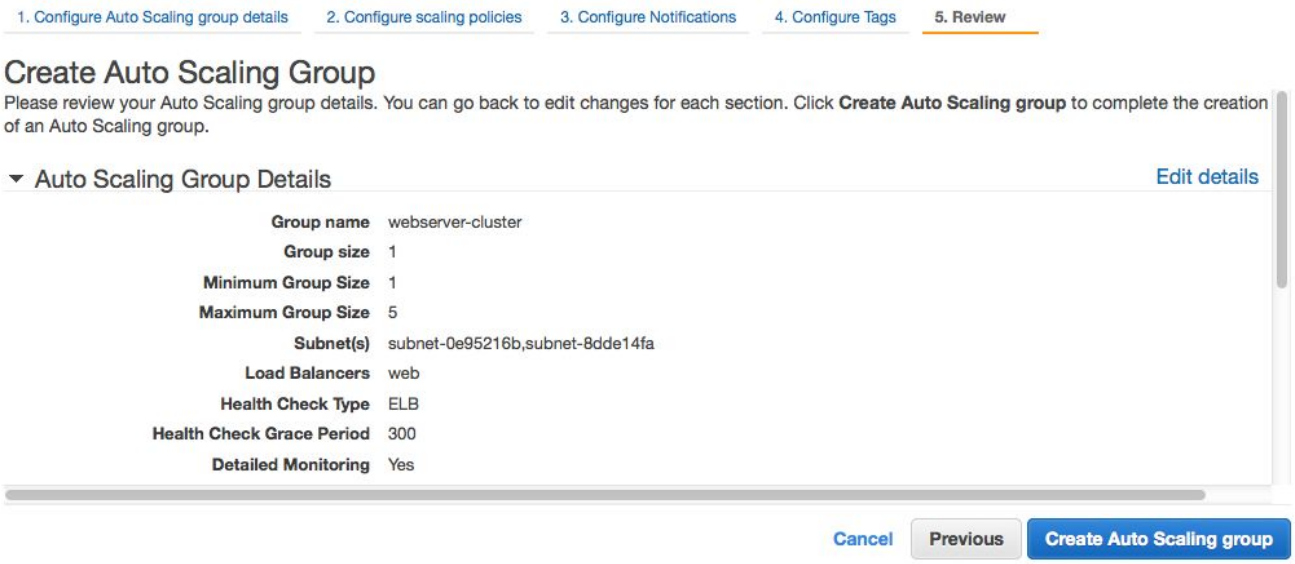
7. Click the **Add notification** button**.** You can use one of the same SNS topics previously created for the CloudWatch alarms. For the **autoscaling-alarm-up** select **launch** and **fail to launch.** Click **Add notification** to add a second notification. This time select an existing **autoscaling-alarm-down** along with **terminate** and **fail to terminate**. It should look similar to the following (but reflect your valid email address):



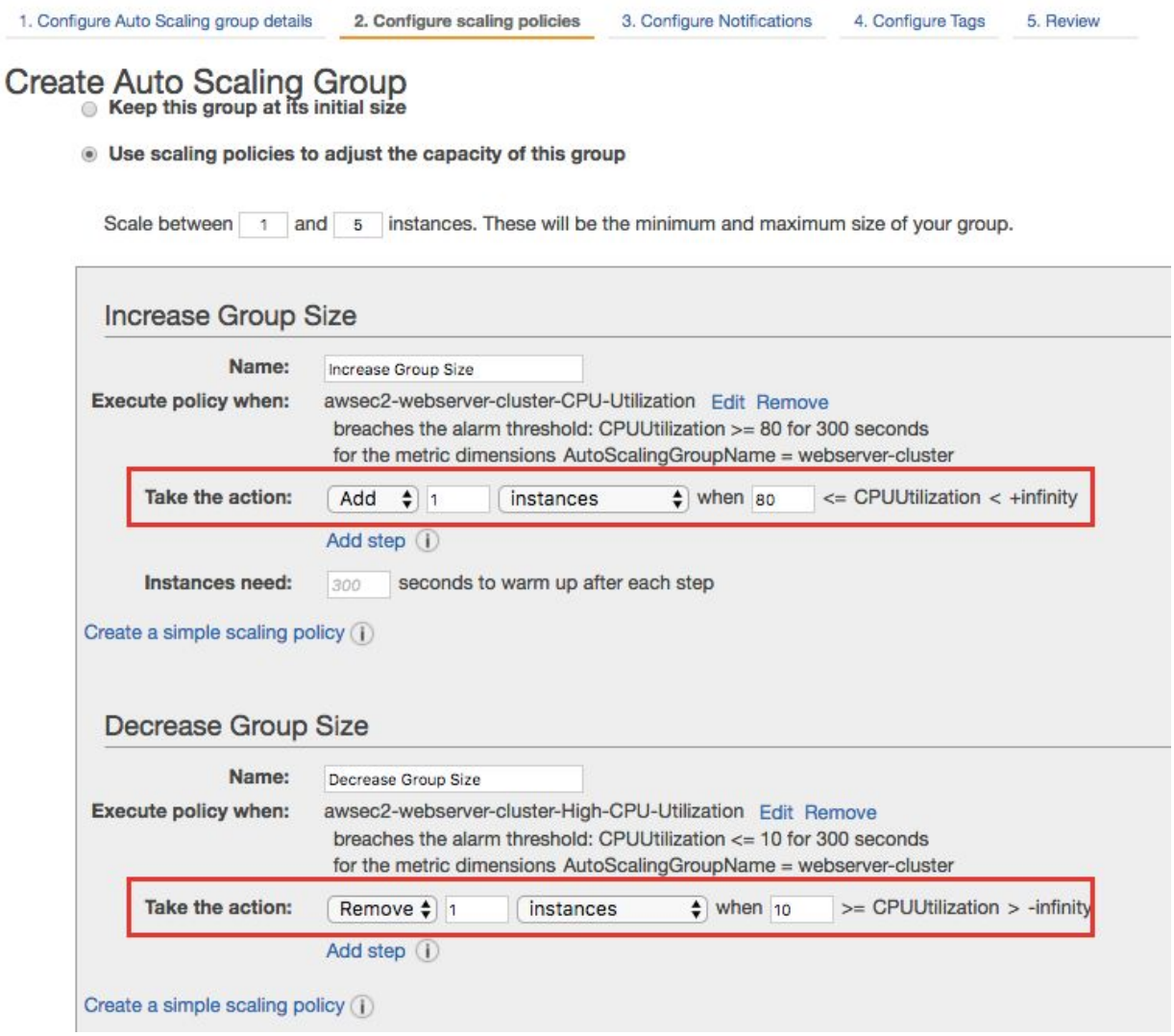
Alternatively, you could create a new topic, or use a single existing topic. For example, use the existing scale up topic and leave all **Whenever instance** options. The advantage of adding a notification for scaling up and another for scaling down (as shown above) is it's a bit easier to read the email alarm at glance, rather then parsing the details within the email body to determine

if there was actually a scale up or down event. You won't need to **Configure Tags** for this lab, so click **Review** when ready.

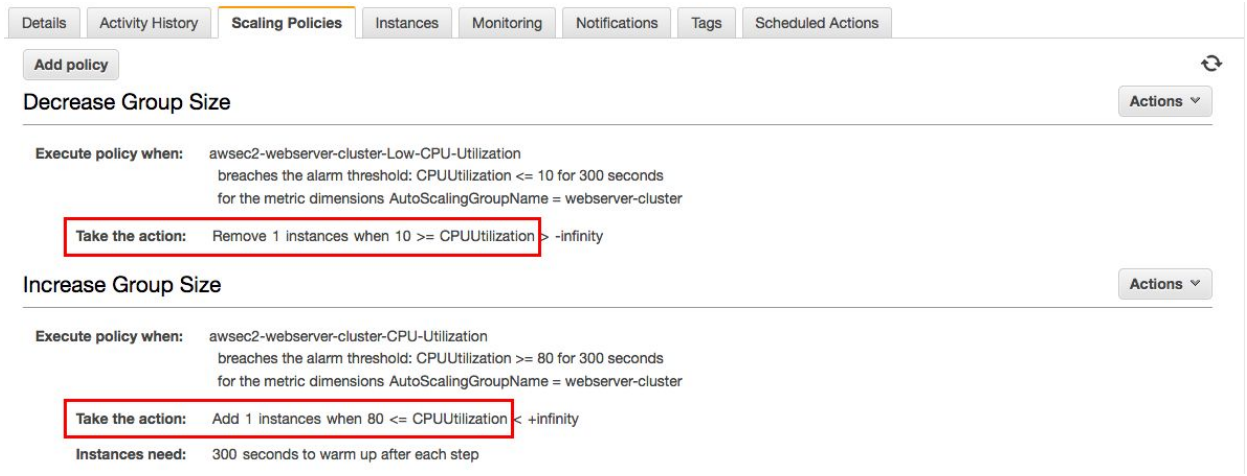
8. Review all the selected options:



Warning! It is very easy to make a mistake with the **Scaling Policies**. For example, mistakenly setting up the relationship incorrectly for increasing/decreasing the group size. To double-check your configuration, click **Previous** until at the **Configure Scaling Policies** section and verify it looks similar to the following:

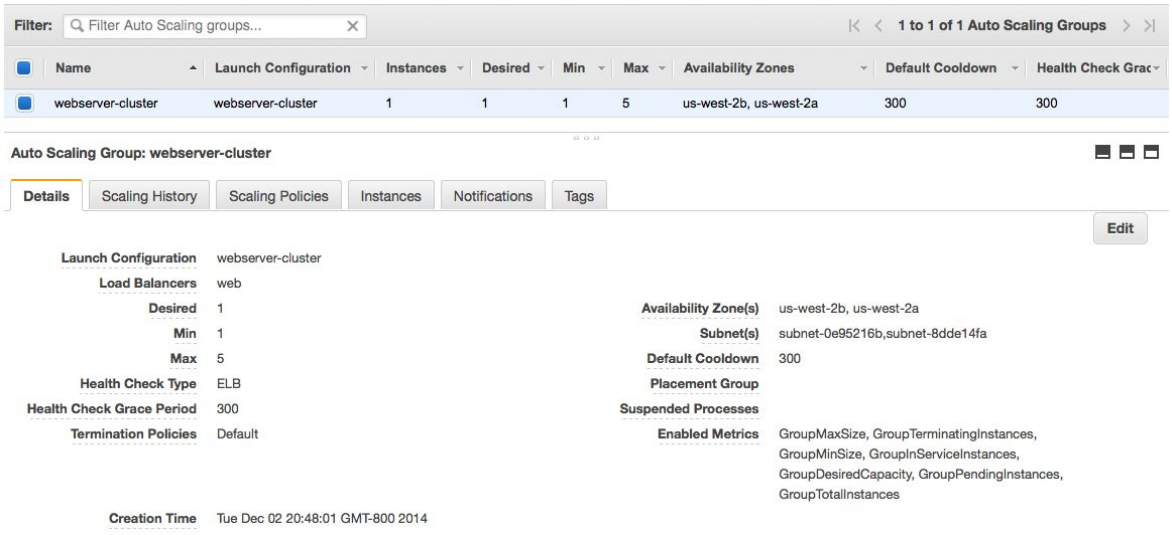


When you are satisfied, start the creation of your cluster by advancing the wizard and clicking **Create Auto Scaling group** on the **Review** screen of the wizard. Once created, viewing your **Scaling Policy** (**Auto Scaling Group > Scaling Policies** tab) should look similar to:



Important! Note that you can change values and parameters for your Scaling Policy with **Actions > Edit**. For example, to change the number of instances to scale up or down by was incorrect, you could fix it with **Actions > Edit**.

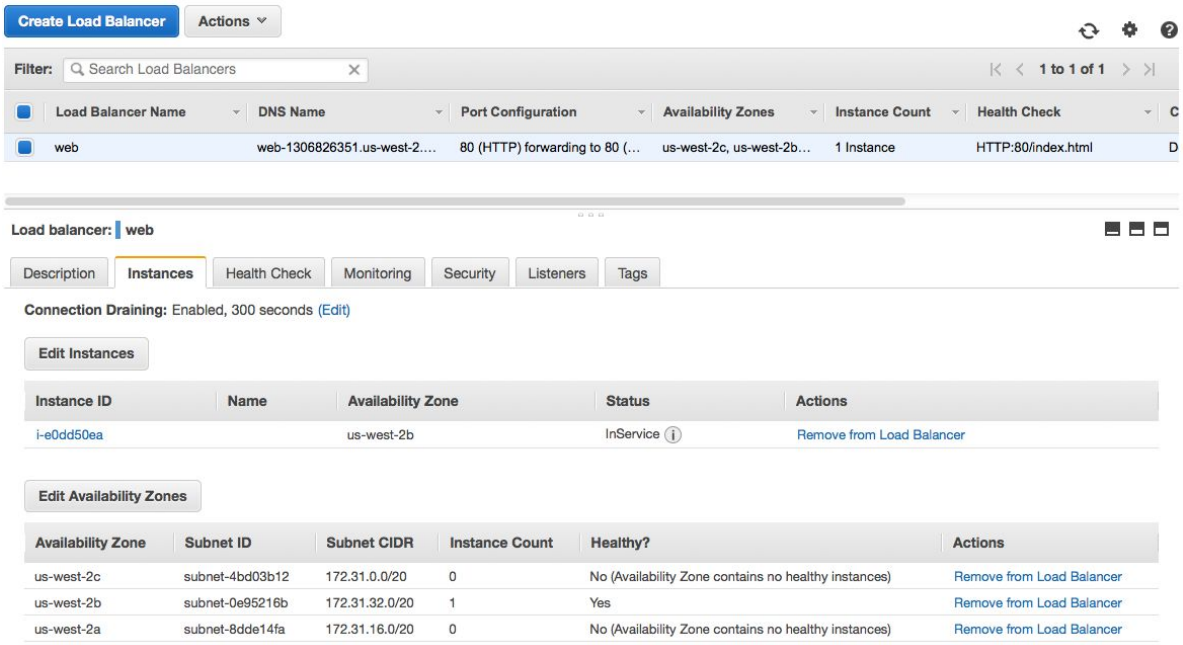
9. Navigate to the **EC2 Dashboard > Auto Scaling Groups**. In a minute or so your cluster will be deployed and your EC2 instance will be ready to use:



Note you can confirm this from the **EC2 Dashboard > Running Instances** too. Notice there should be one instance (singular) not multiple instances. Why? You set **Desired** and **Min** to *1* earlier.

10. Open the **Load Balancers** section, select your previously created ELB, and then open the **Instances** tab. You can see the new Auto Scaling instance(s) automatically added to the ELB

configuration:



End to end Testing (EC2 Auto Scaling Groups)

Introduction

Performing end-to-end tests to make sure everything is working as you think it should is very important. Although this is may be an automated procedure performed by a QA (Quality Assurance) department, often a quick sanity test by other individuals and/or groups directly from the AWS console is also helpful. This Lab Step will point out a few ways to test that your Launch Configuration is working in conjunction with the Auto Scaling group and CloudWatch Alarm (which uses AWS Simple Notification Service (SNS)).

Note: This Lab Step assumes you are fairly comfortable at this point with the AWS console and can navigate to the various parts of the console already used within this Lab. The *Instructions* below are fairly terse based on this assumption.

Instructions

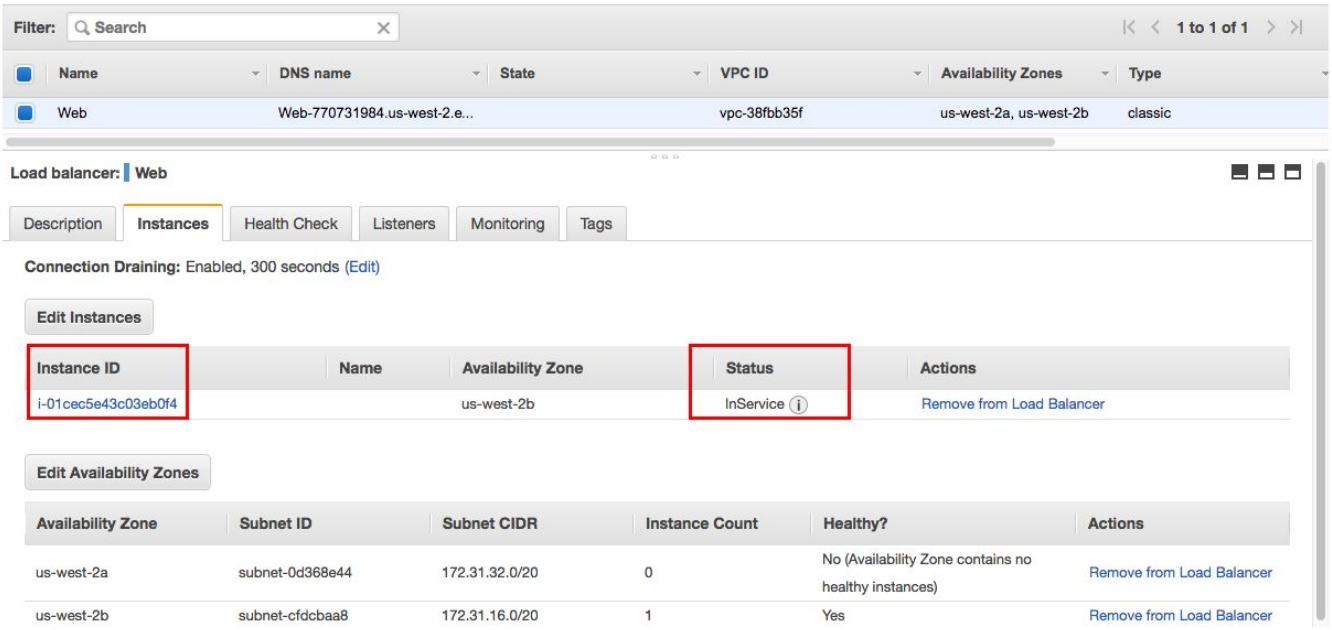
Quick Tests

- Confirm an instance is running and settles at the desired number of one. *Reminder*: You can do this from the **EC2 Dashboard > Running Instances** or **Auto Scaling Group > Instances** tab:

- Check the email for the address you entered for the SNS notifications earlier. You should have received a **AWS Notification - Subscription Confirmation** email from **AWS Notifications** for both

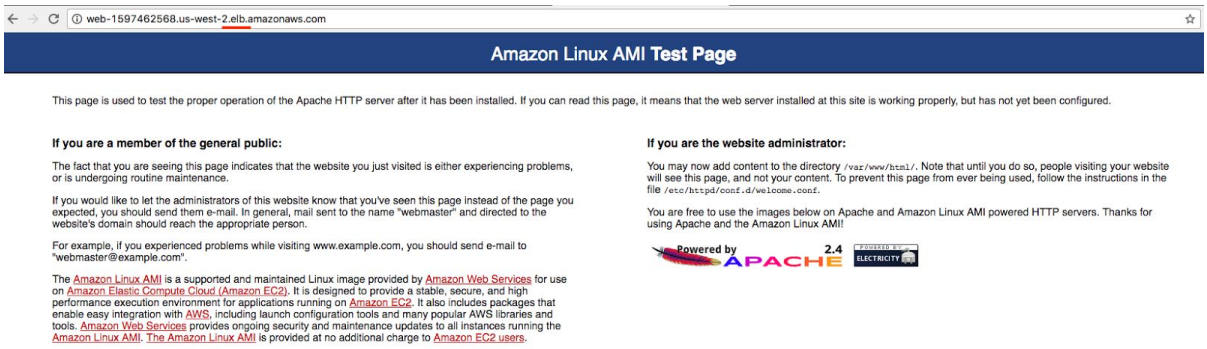
a **autoscaling-alarm-up** and **autoscaling-alarm-down** SNS topics. Click the **confirm subscription** link in the email body if so.

- Open up a new browser window or tab. Enter either the **Public DNS** or **Public IP** address of the running instance. You should receive the **Amazon Linux AMI test page** from the Apache web server you installed and started via **User data** in the Launch Configuration. Now you know the instance is up and running and the Apache web server installed OK when the instance started.
- Navigate to **EC2 > Load Balancers**, then select the **Instances** tab for your Load Balancer. Confirm the running instance is registered with the Load Balancer *and* the **Status** is **InService** (meaning the health checks are running correctly):



- *Tip:* If trouble shooting efforts are required, here is a very handy trick. If, for example, health checks are failing, then chances are the running instance will get terminated automatically. A new one will start, but five minutes later the health check will also fail and that new instance will get terminated. Rinse/repeat! To "break" the cycle while troubleshooting:
    - Navigate to **Auto Scaling Groups > Details** tab
    - Click **Edit** then click the **Suspended Processes** field
    - Select **Terminate** (This will prevent instances in your group from getting terminated. ***Important!*** This trick is for troubleshooting only! Don't forget to remove the configuration once the issue is resolved.)
    - *Note*: **Launch** is also an option here. Temporarily setting this to **Launch** has the same effect as setting **Desired** and **Min** to *0.* After either of these settings and manually terminating an instance, a new instance will not be started. (A very helpful trick if you want to put a hold on everything until the next day, saving on potential instance usage charges.)
- From the **EC2 > Load Balancing > Load Balancers** page, copy the **DNS name.** (Example: Web-770731984.us-west-2.elb.amazonaws.com) Paste the **DNS name** into a new browser window or tab. You should see the default **Amazon Linux AMI Test Page** again. Note

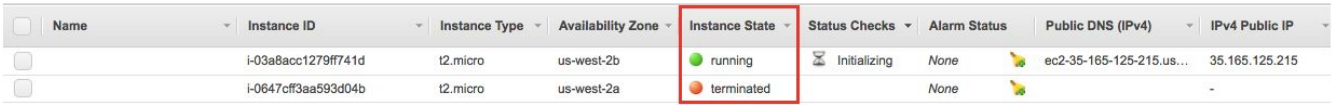
that earlier you saw this page, but you had bypassed the load balancer, hence it only confirmed the instance was up and running Apache. This tests that the ELB will route requests to an instance in its instance pool. Notice the URL includes the ELB you created earlier. The request went through the ELB to the instance:



Auto Scaling Tests

Note: This section assumes you know how to SSH into the Linux instance. Either from the EC2 console (on a Mac OS/Linux local host) or using Putty (for a local Windows host). The details are not included in this Lab Step.

- From the **EC2 > Running Instances,** select **Actions > Instance State > Terminate** on the sole running instance. It should relaunch automatically. Let the new instance launch and settle into a running state. This could take a minute or two:



- SSH into the running instance and stress test it.
    - The simplest way to do this is to download and then use the correct key pair file supplied for you by the Cloud Academy platform under the **YOUR LAB DATA** section (at the top). Two key pairs are available for use: PEM (Privacy-Enhanced Mail) and PPK (public-private key pair). PEM files can be used directly by a local Linux SSH client. Connecting from a local Windows host to a running Linux host requires a PPK file. (Although you can use PuTTYgen to convert a PEM file to a PPK, Cloud Academy has done that for you for convenience sake.)
    - *Hint for Linux users*: From the **Running Instances** page click **Connect**. The SSH command for the selected instance using the local private key file you previously downloaded earlier can be copy/pasted and used from a terminal window. You may need to change the permissions using the `chmod` command as instructed to make sure the key file has read-only permissions.)
    - *Hint for Windows users*: Download and use the PPK file supplied by Cloud Academy. As previously noted, it's simpler than creating your own key, converting it using PuTTYgen and then connecting via PuTTY.

- Install stress by using the yum package manager. *Note*: Although stress should already be installed from the User data in your Launch Configuration, this will guarantee the latest/greatest version, and that the install directory is in your environment. ($PATH)

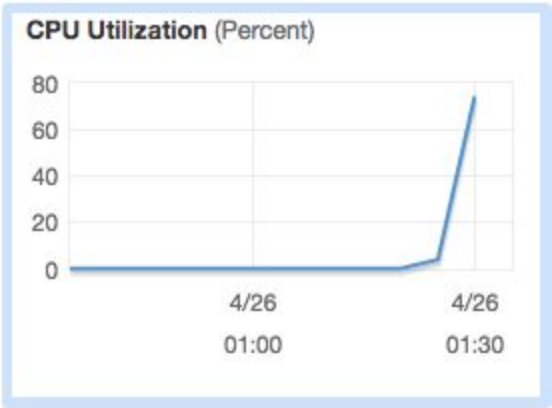
```
$ sudo yum install stress
Loaded plugins: priorities, update-motd, upgrade-helper
amzn-main                                                | 2.1 kB     00:00

amzn-updates                                             | 2.3 kB     00:00

Resolving Dependencies
--> Running transaction check
---> Package stress.x86_64 0:1.0.4-4.2.amzn1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
=
 Package        Arch           Version                  Repository         Size
================================================================================
=
Installing:
 stress         x86_64         1.0.4-4.2.amzn1          amzn-main          38 k

Transaction Summary
================================================================================
=
Install  1 Package

Total download size: 38 k
Installed size: 89 k
Is this ok [y/d/N]: y
Downloading packages:
stress-1.0.4-4.2.amzn1.x86_64.rpm                          |  38 kB   00:00

Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : stress-1.0.4-4.2.amzn1.x86_64
1/1
  Verifying  : stress-1.0.4-4.2.amzn1.x86_64
1/1

Installed:
  stress.x86_64 0:1.0.4-4.2.amzn1
```

- Enter `which stress` to confirm the stress executable is in your path. (/usr/bin/stress)
- Finally, run stress to eat up CPU cycles for five minutes (-t 5m option):

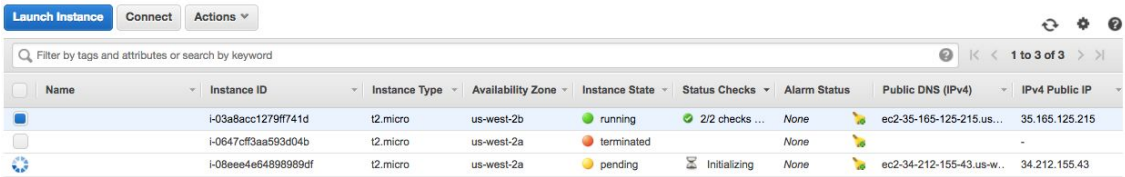
```
$ stress -c 2 -i 1 -m 1 --vm-bytes 128M -t 5m
```

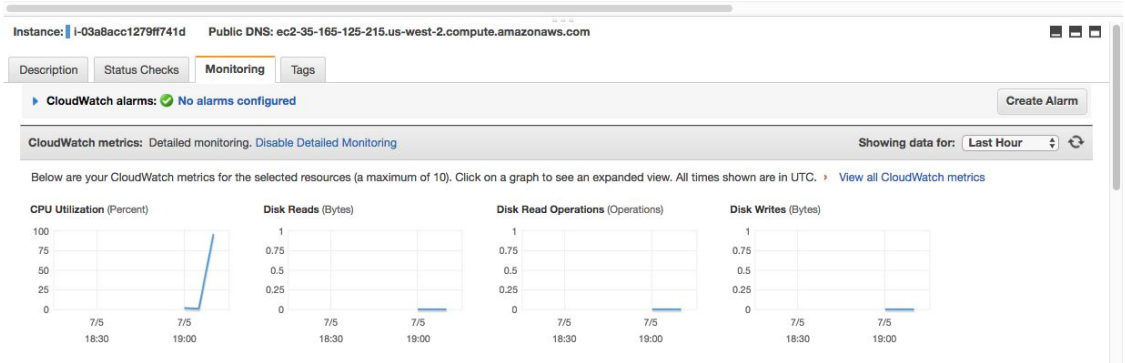
```
stress: info: [23710] dispatching hogs: 2 cpu, 1 io, 1 vm, 0 hdd
stress: info: [23710] successful run completed in 300s
```

- Navigate to **EC2 Dashboard > Running Instances > Monitoring** tab. Look at the **CPU Utilization**. Hint: You may want to click the refresh icon every 30 seconds or so on the bottom pane. It should quickly ramp up from nearly 0% to 100% within a few minutes:

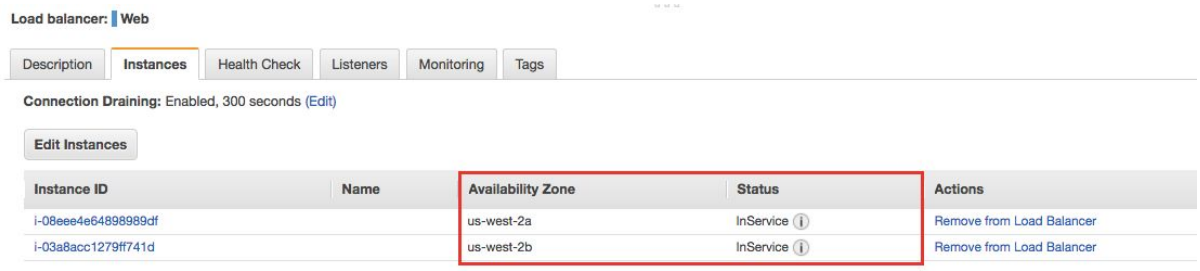


- Recall the way that you configured the scaling policy in your Auto Scaling group. What do you think should happen as the CPU climbs on the running instance over the next 5+ minutes?
    - A new instance should automatically start:





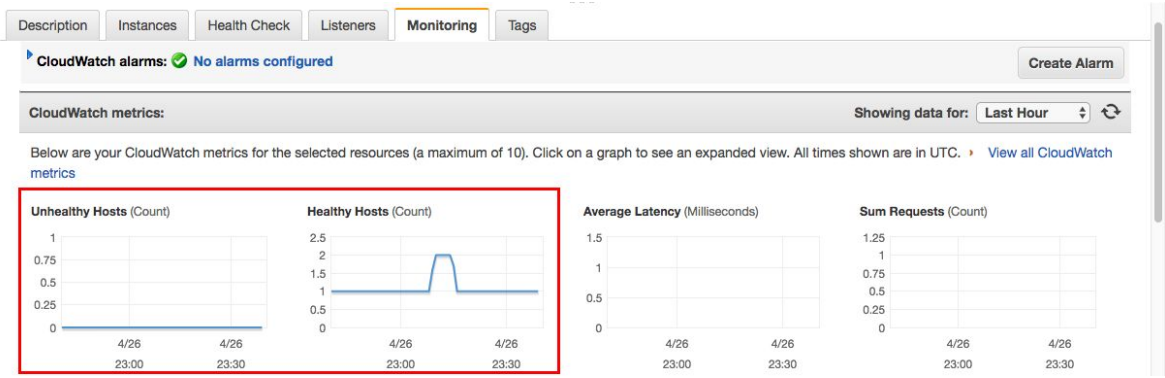
- Further, it should register with the ELB and pass health checks:



- Keep waiting for another ~5 minutes. Should anything else happen?
    - Yes! Since the stress test runs for only 5 minutes, the CPU utilization spread across both instances in your load balancer pool will quickly go back down to near zero. Since the scale-down policy relationship is <= 10, one of the two instances should be automatically shut down and then terminated:

The **Instance State** transitions from **shutting-down** to **terminated** automatically. Congratulations... it's working! If you check your email, you should receive another notification as a CloudWatch Alarm is raised. For example, the email from **AWS Notifications** titled: **ALARM: "awsec2-webserver-cluster-Low-CPU-Utilization" in US West - Oregon.** Eventually, you should see one running instance, and two terminated instances (one you terminated manually, and another terminated based on the alarm triggered when the CPU dropped below 10% for 5 minutes):

- Navigate to **Load Balancers > Monitoring** tab. This is another helpful page, where you can confirm no **Unhealthy Hosts**, and see the **Healthy Hosts** scale up from 1 to 2 and back down to 1:



When you are done looking at various screens in the AWS Management Console to check status, feel free to terminate any running instances. (Otherwise, we'll do it for you when the lab is completed.)

Summary

Although there is always more testing that could be done, or even automated testing implemented, the manual checks performed in this Lab Step are pretty thorough. Chances are very good that if all of the tests in this Lab Step work then your Auto Scaling Group, Launch Configuration, Elastic Load Balancer, CloudWatch alarm and SNS notification are all properly configured and working together. Congratulations!!