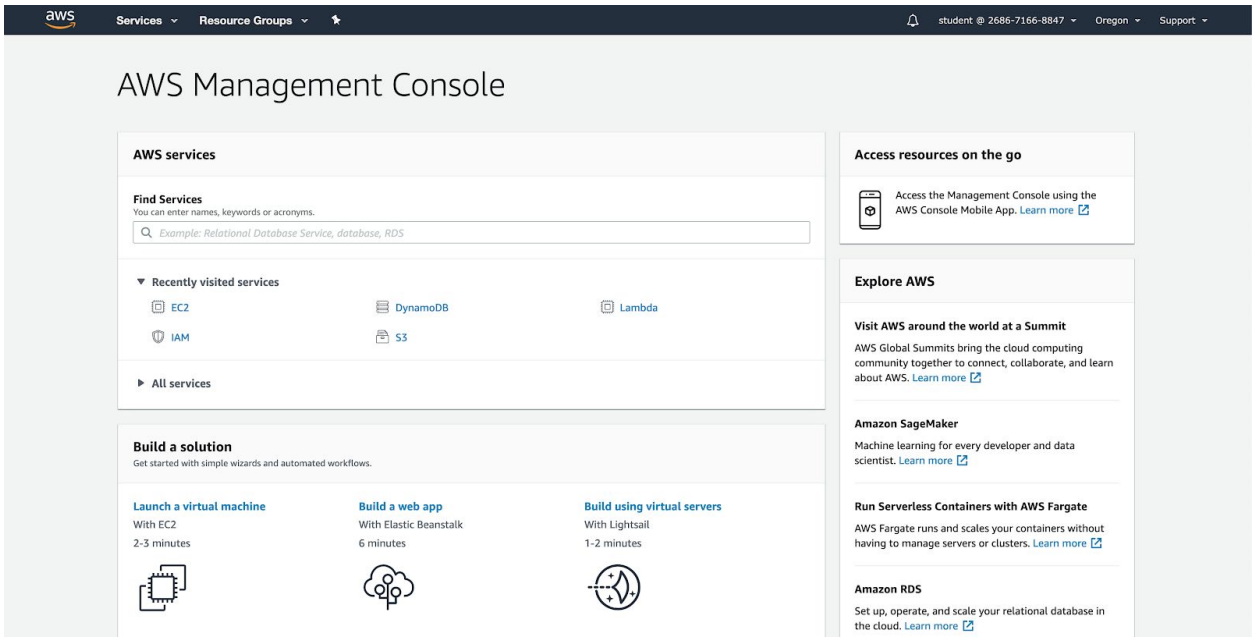
1 Logging in to the Amazon Web Services Console
2 Auto Scaling Overview
3 Creating a Network Load Balancer
4 Creating a Launch Configuration
5 Creating an Auto Scaling Group from a Launch Configuration
6 Testing the Auto Scaling Group from End-to-End
Validate Working with Amazon EC2 Auto Scaling Groups and Network Load Balancer

==========

Introduction

This Lab experience involves Amazon Web Services (AWS), 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, and even setting up new IAM Users.

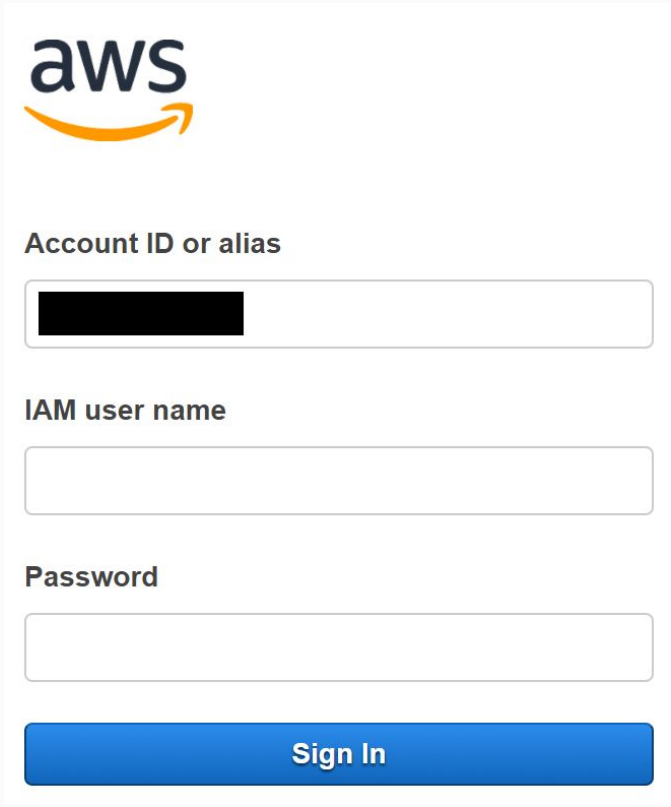
Instructions

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

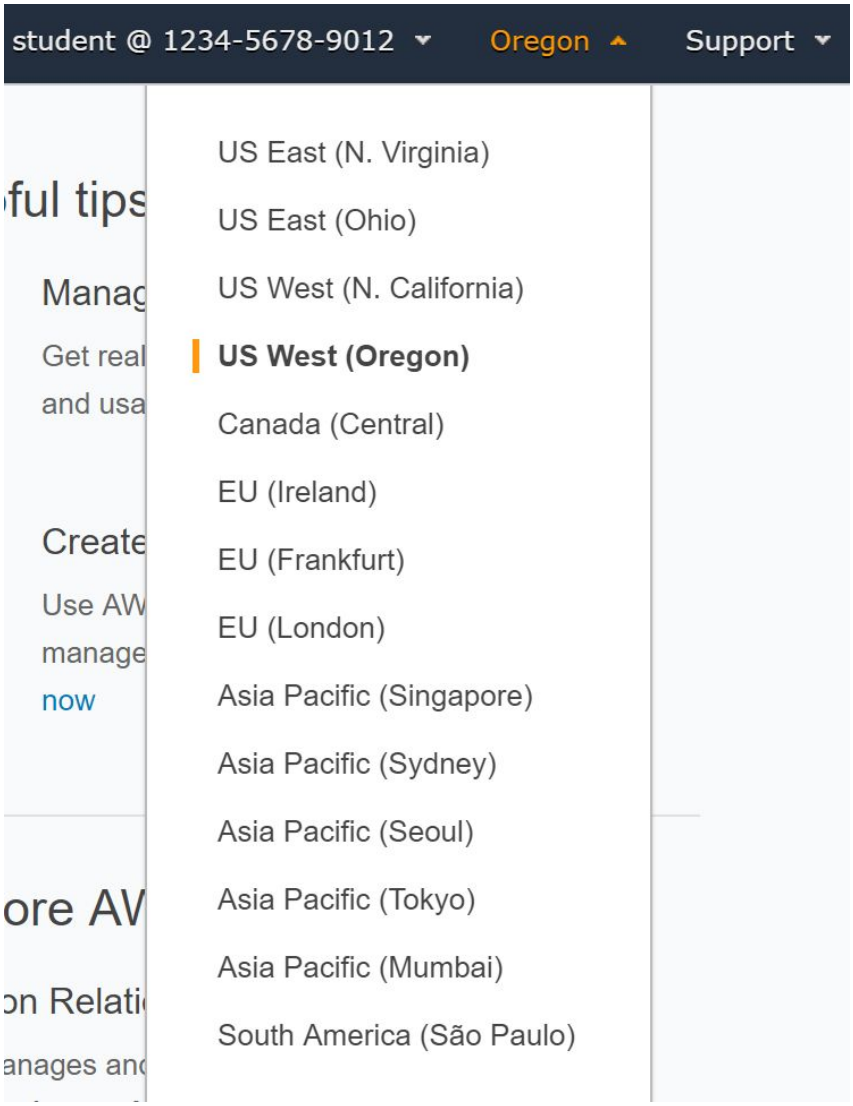
Open Console

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**: `Ca1_2XXAE2dC`



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



Amazon Web Services are 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 2`** for this Lab.

Introduction

Before going to the AWS console and creating an Auto Scaling Group, it's important to understand the key components of an Auto Scaling Group. AWS has done an excellent job defining them so the official definition is placed below for convenience sake:

Groups

> Your EC2 instances are organized into groups so that they can be treated as a logical unit for the purposes of scaling and management. When you create a group, you can specify its minimum, maximum, and desired number of EC2 instances.
>
> **Launch configurations**
>
> Your group uses a launch configuration as a template for its EC2 instances. When you create a launch configuration, you can specify information such as the AMI ID, instance type, key pair, security groups, and block device mapping for your instances.
>
> **Launch template**
>
> A launch template is similar to a launch configuration, in that it specifies instance configuration information. ... However, defining a launch template instead of a launch configuration allows you to have multiple versions of a template. With versioning, you can create a subset of the full set of parameters and then reuse it to create other templates or template versions.

After you complete this Lab you can read the full AWS documentation on Auto Scaling here.

Auto Scaling groups are commonly paired with load balancers that evenly distribute requests across all the instances in the group. In this Lab, you will create a load balancer before creating an Auto Scaling group so that the Auto Scaling group can be configured to work with the load balancer at creation time.

Introduction

Elastic Load Balancing (ELB) automatically distributes incoming application traffic across multiple Amazon EC2 instances. They enable 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.

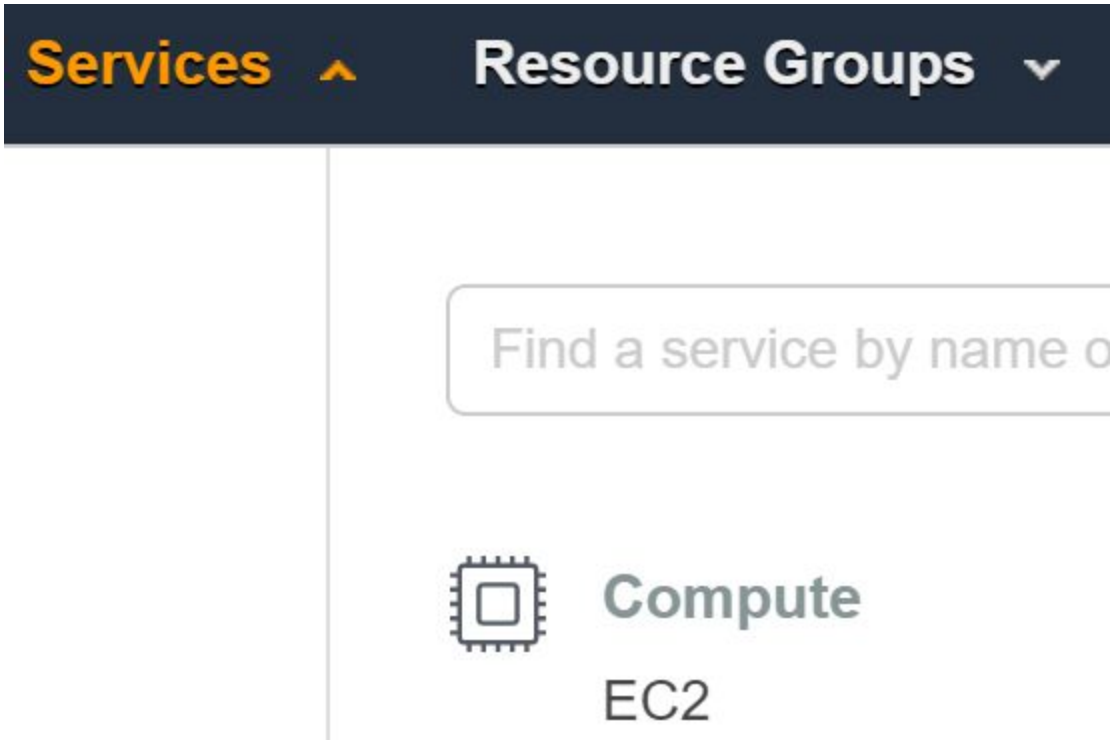
ELB detects unhealthy instances within a pool and automatically reroutes traffic to healthy instances until the unhealthy instances have been restored. Elastic Load Balancers can be enabled within a single Availability Zone or across multiple zones for greater consistent application performance.

There are several ELB load balancers to choose from. The network load balancer is a network layer (layer-4) load balancer operating on TCP connections and UDP. It can scale to millions of requests per second and is a more modern alternative to the classic load balancer (also a layer-4 load balancer). With a network load balancer, backend targets are organized into *target*

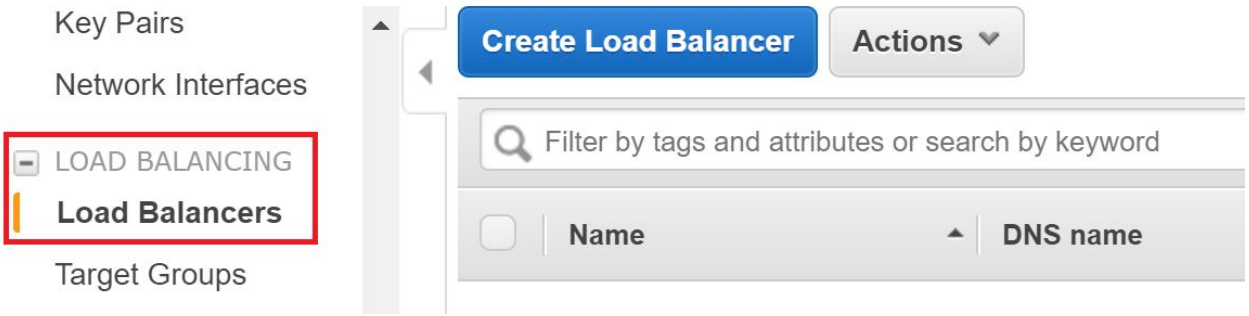
groups which the network load balancer distributes traffic across. You will create a network load balancer in this Lab Step.

Instructions

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

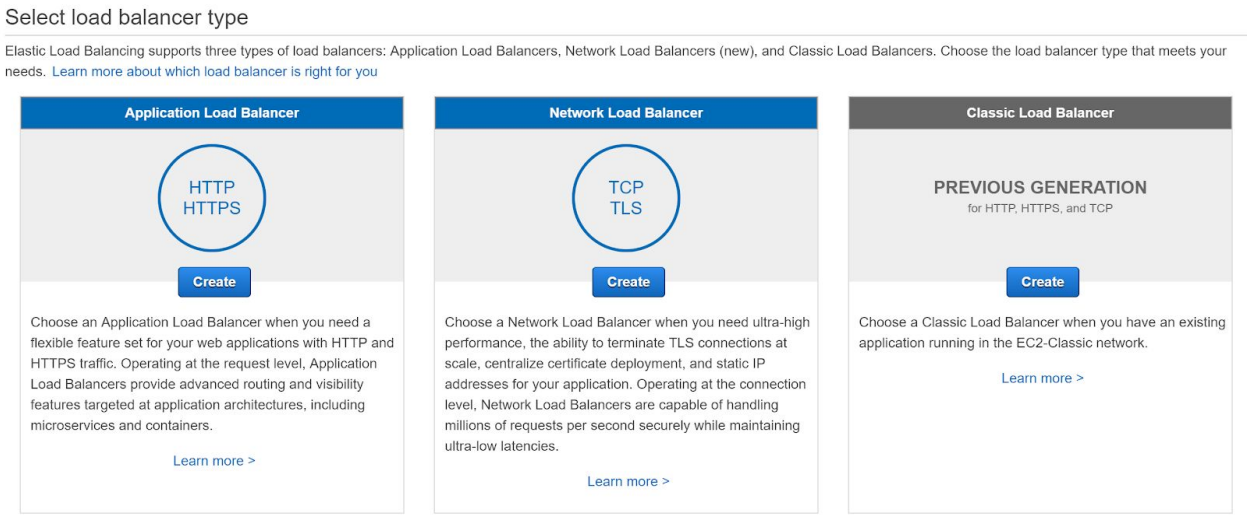


2. In the left pane, click **Load Balancers** in the **Load Balancing** section:



3. Click **Create Load Balancer**.

4. Take a moment to read the information for the load balancer types begore clicking **Create** in the **Network Load Balancer** tile:

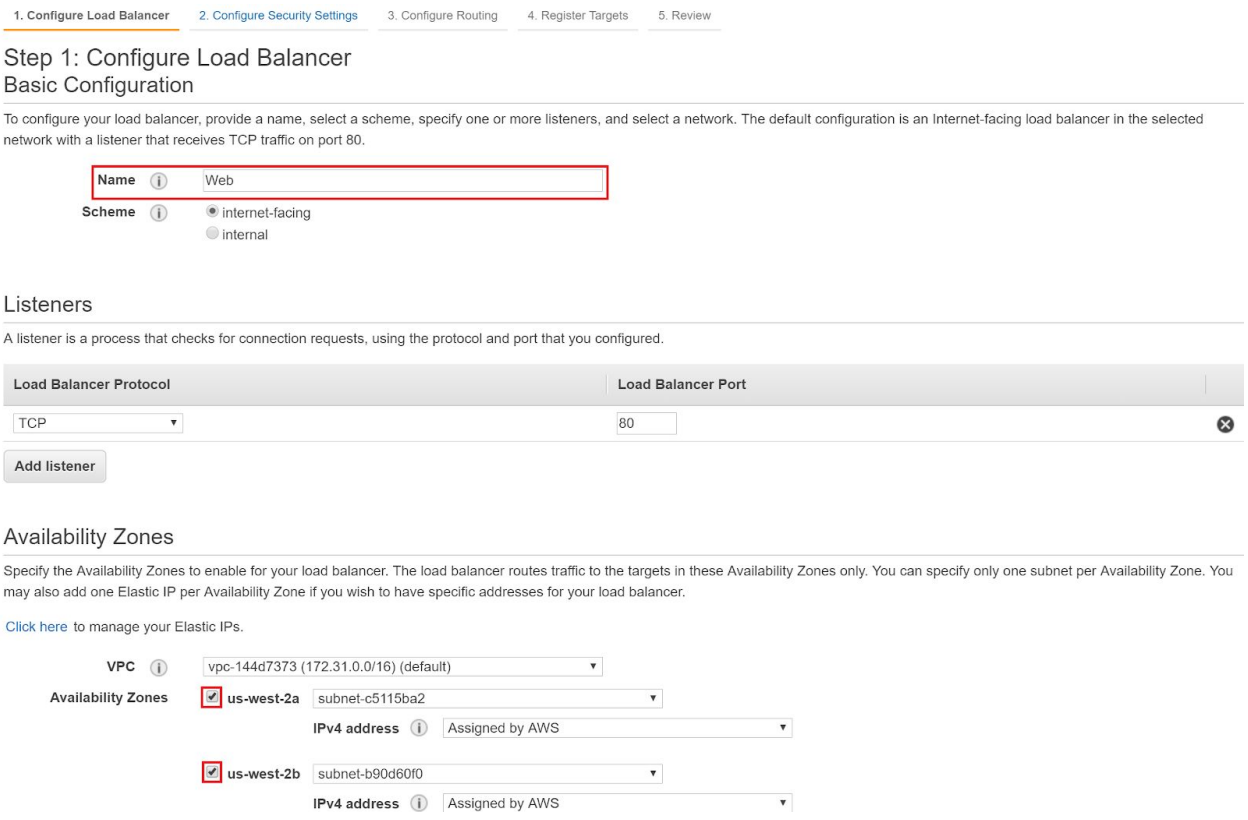


A multi-step wizard starts for creating a load balancer.

5. On the **Configure Load Balancer** step, set the following values leaving the others at their defaults:

- **Basic Configuration:**
    - **Name**: *Web*
- **Availability Zones**:

    - **Availability Zones**: Check **us-west-2a** and **us-west-2b**



Notice the default listener is TCP port 80 which is used for serving HTTP traffic.

6. Click **Next: Configure Security Settings** when ready.

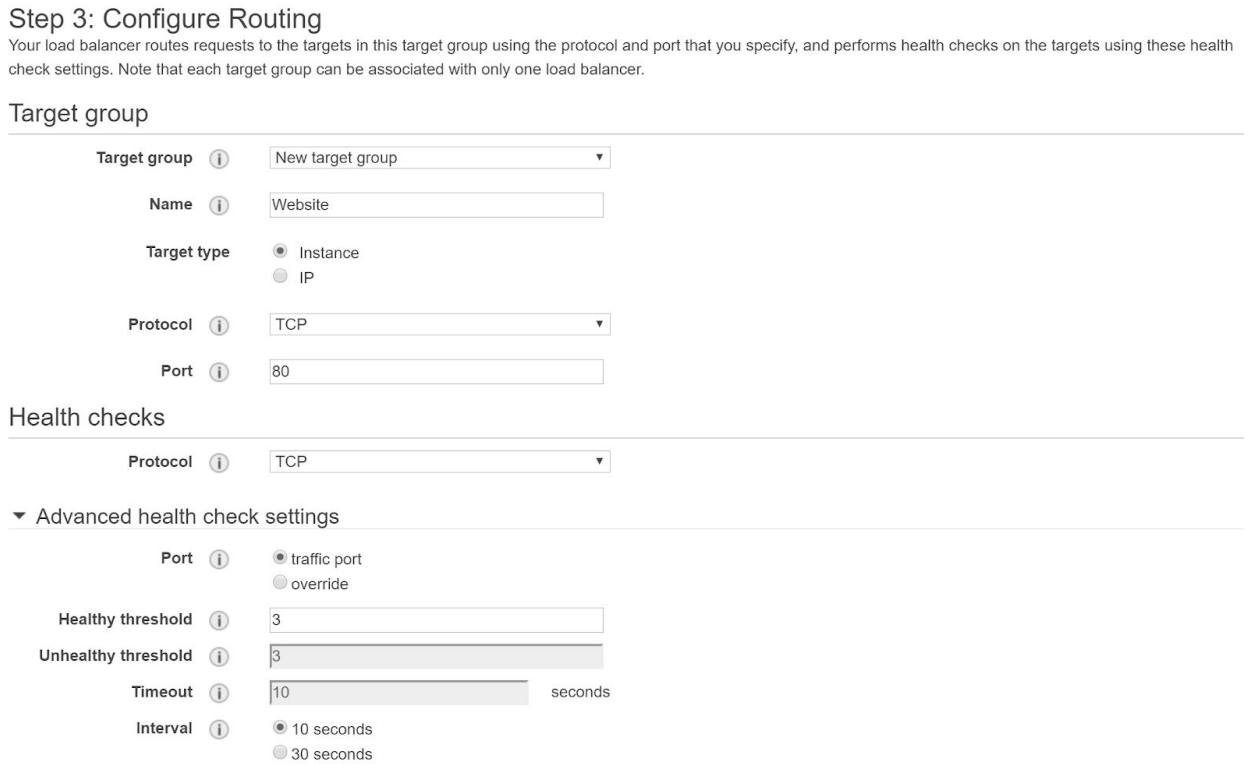
7. On the **Configure Security Settings** step, click **Next: Configure Routing**.

The warning message informs you that the listener isn't secure (not using TLS). You should carefully consider if you do not need TLS. For this Lab, it is not going to be an issue because there will be no sensitive information being transmitted.

8. On the **Configure Routing** step, set the following values leaving the others at their defaults:

- **Target group:**
  - **Name**: *Website*
- **Health checks**:
  - **Advanced health check settings**: (Click the triangle to expand the section)

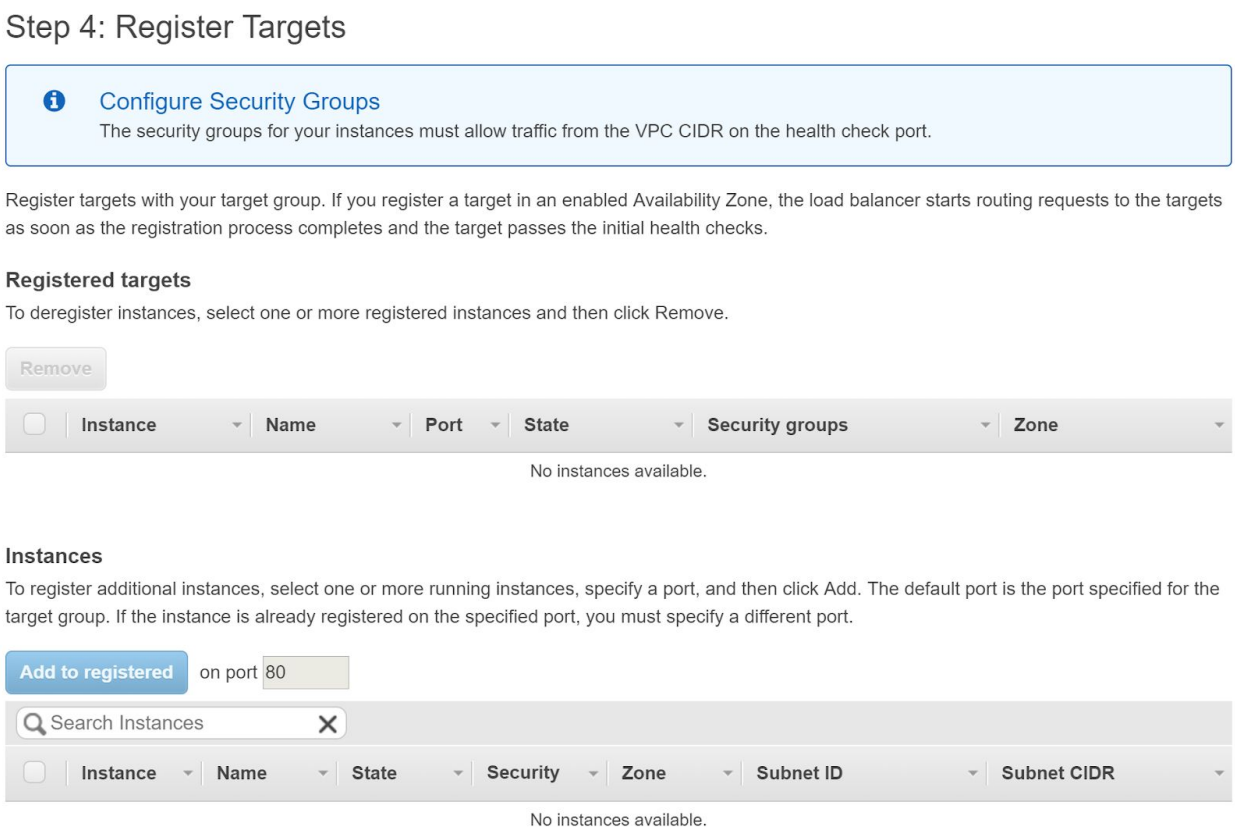
- **Interval**: **10 seconds** (This will cause instances to reach a healthy state faster for this Lab, but may be too fast for certain applications)



Target type allows you to target **IP** addresses rather than instances. This gives you the ability to use the Network Load Balancer with instances outside of AWS.

9. Click **Next: Register Targets**.

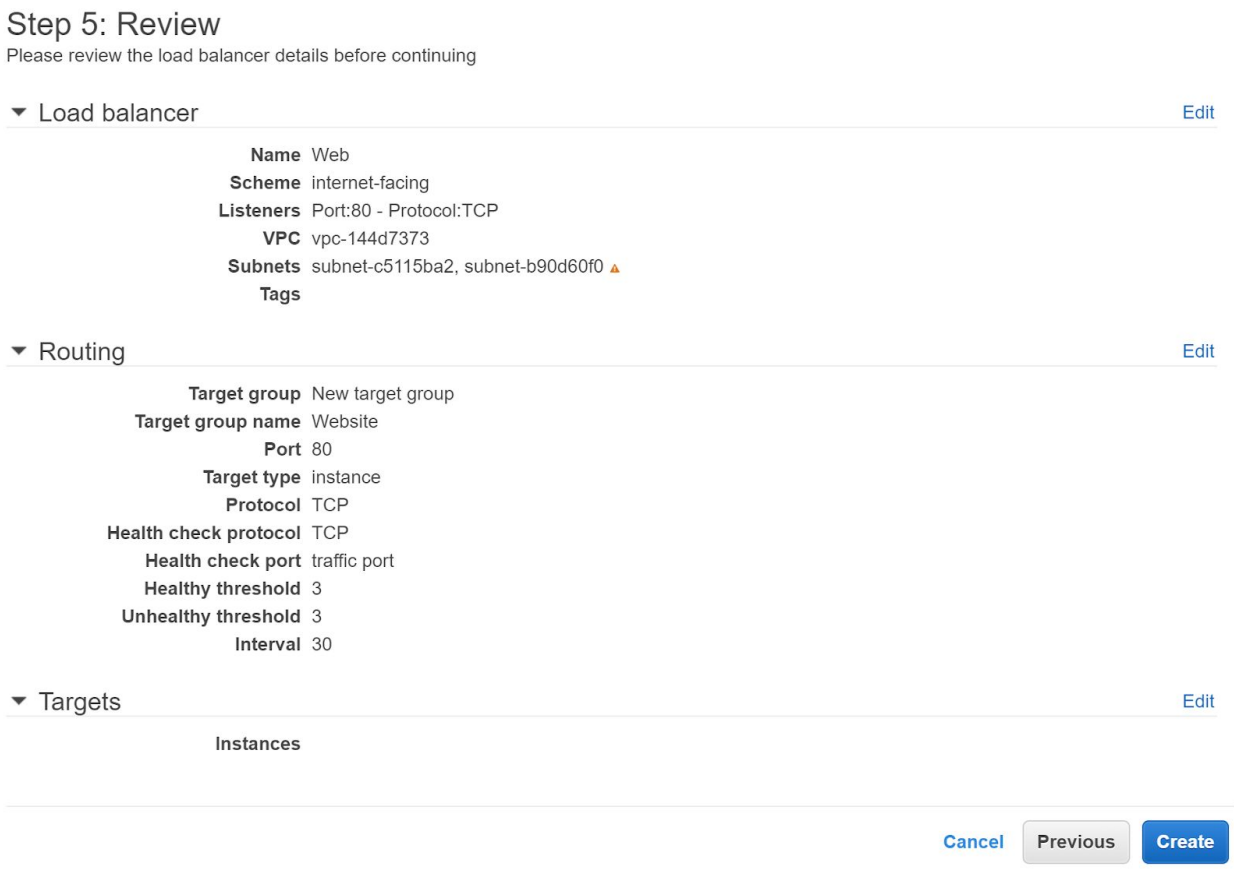
On the **Register Targets** step, notice there are **No instances available**:



The message is because you have not created an Auto Scaling Group or launched EC2 instances yet. That is not a problem. Your will configure your Auto Scaling group to register its EC2 instances in the Network Load Balancer's target group.

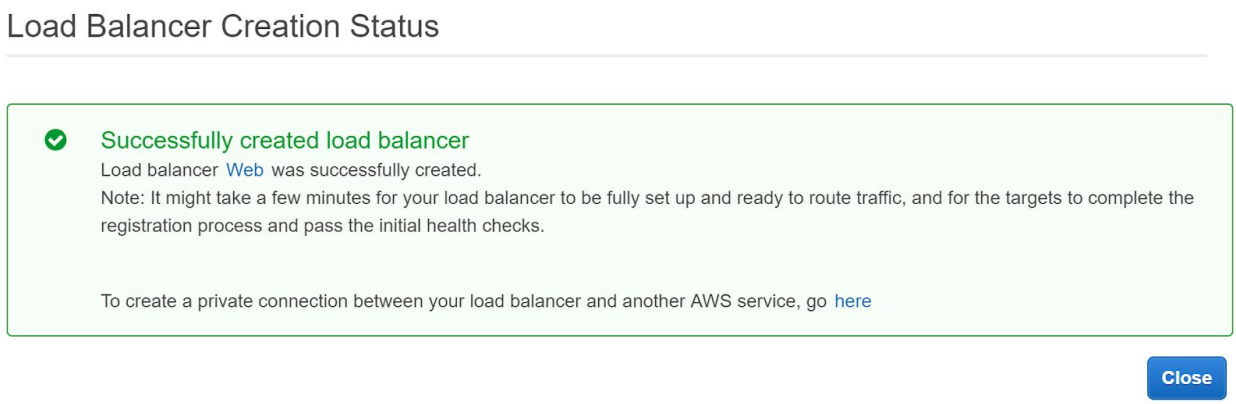
10. Click **Next: Review**.

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

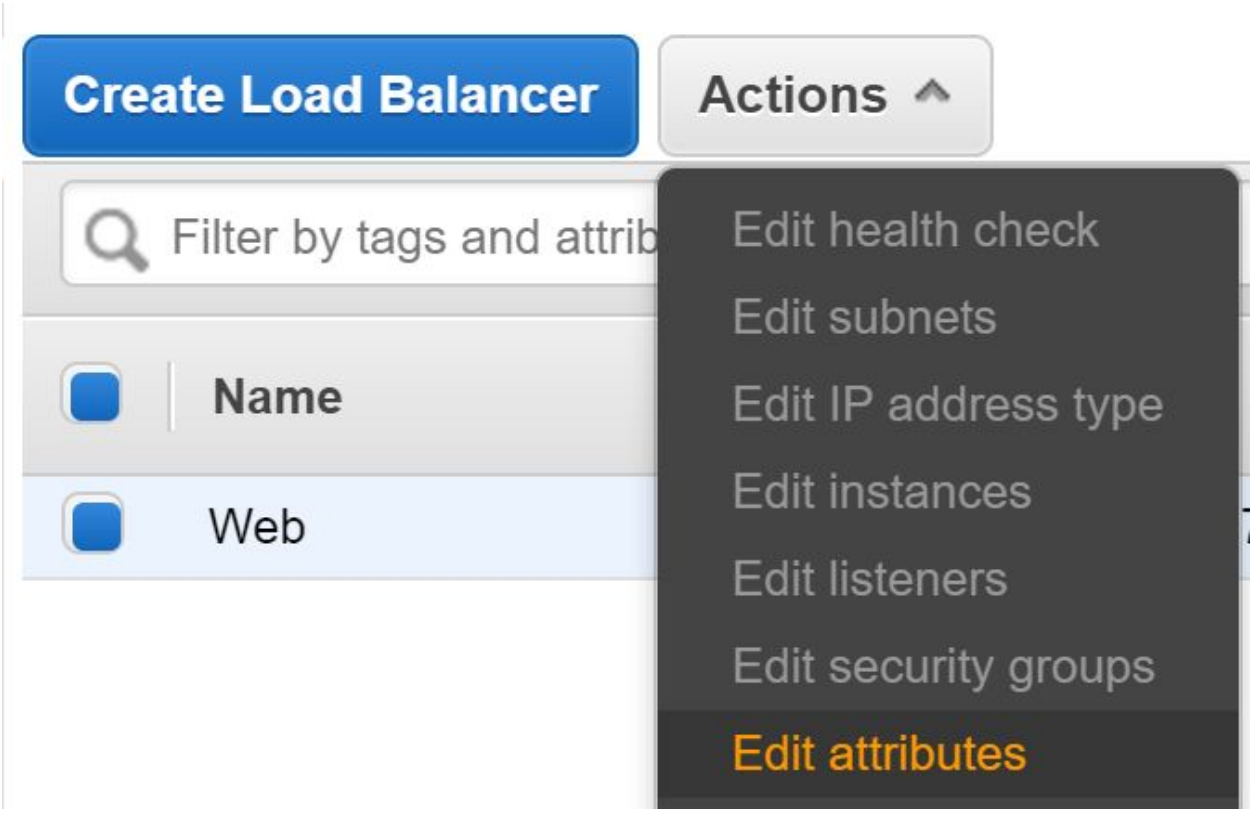


There is a warning icon by the **Subnets** value to inform you that there are no instances added from either Availabiliity Zone associated with the subnets. That is expected since no instances are registered as targets yet. When everything has been set up, it is a best practice to have instances in all availability zones for high availability.

12. Wait for the **Load Balancer Creation Status** success message to display before clicking **Close**:

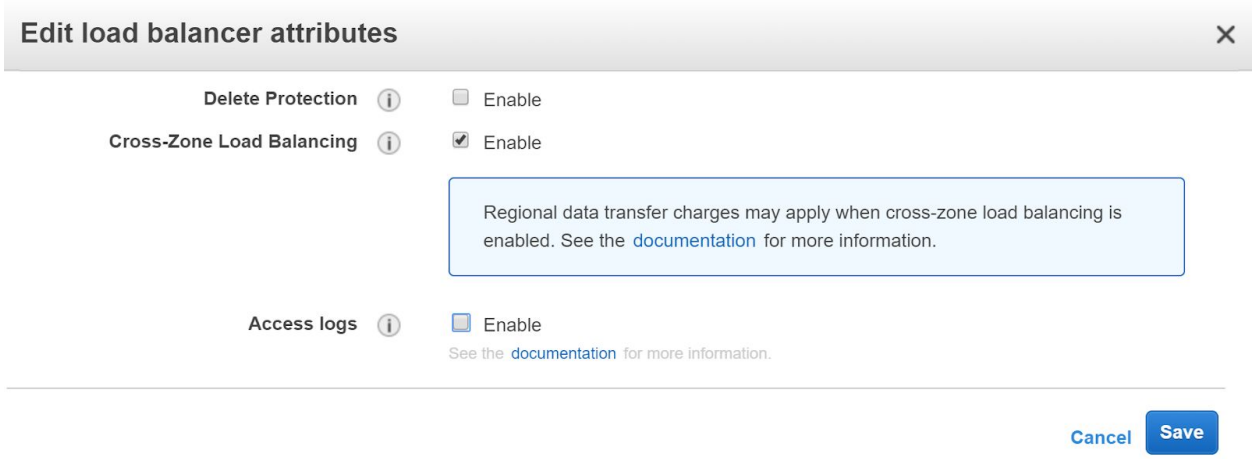


13. In the load balancers table, ensure the **Web** load balancer is selected and click **Actions** > **Edit attributes**:



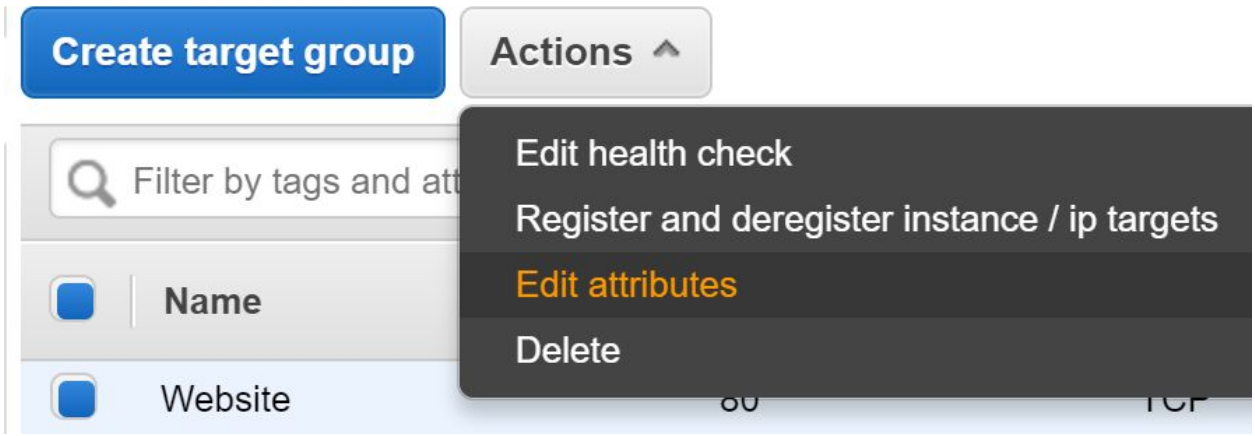
14. In the **Edit load balancer attributes** form, set the following value before clicking **Save**:

- **Cross-Zone Load Balancing**: **Enabled**

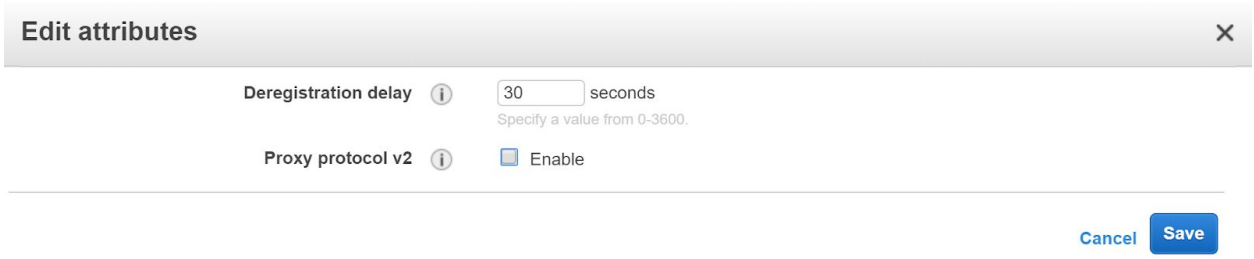


You must enable **Cross-Zone Load Balancing** to achieve the highest level of availability. Without enabling this feature, clients could cache the DNS address of the load balancer node in one availability zone and that node would only distribute requests to instances within the availability zone. Cross-Zone Load Balancing allows every load balancer node to distribute requests across all availability zones, although for the Network Load Balancer there are data transfer charges when this feature is enabled. (There are no data charges for other types of load balancers)

15. Navigate to the **LOAD BALANCING** > **Target Groups** section of the EC2 Console, then select the **Website** target group and click **Actions** > **Edit attributes**:



16. Change the **Deregistration delay** to *30* seconds and click **Save**:



The deregistration delay specifies how long the load balancer should wait before removing an instance from the target group. The default value of 300 seconds gives connections to the instance five minutes to drain before they are forcefully closed. Depending on your application, you may be able to reduce to delay to remove instances more quickly. Thirty seconds is enough for this Lab.

Summary

In this Lab Step, you created a Network Load Balancer with a target group ready to service HTTP requests on port 80. This load balancer will be used as the front-end to a website. The website will run on EC2 instances that are created via an Auto Scaling group. This is a very common use case.

Introduction

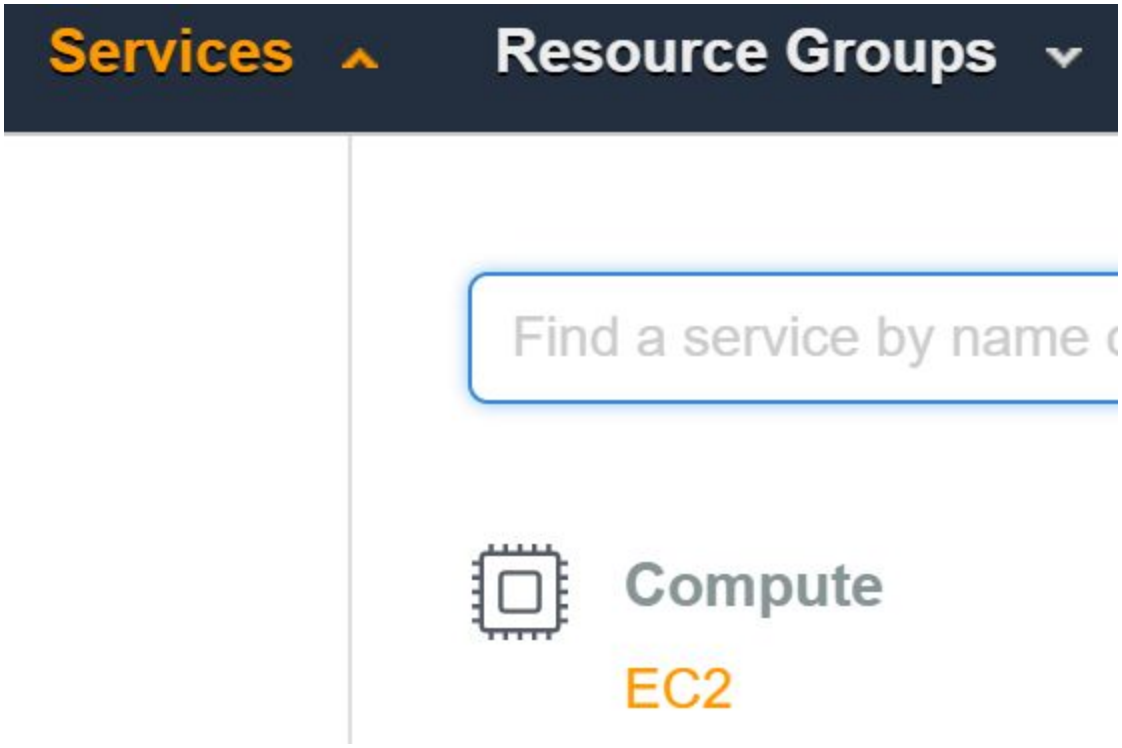
A Launch Configuration is a template that the Auto Scaling group uses to launch Amazon EC2 instances. If you've launched an individual EC2 instance before, you've already walked through the process of defining compute characteristics such as the instance type, security groups, and configuration scripts. 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.

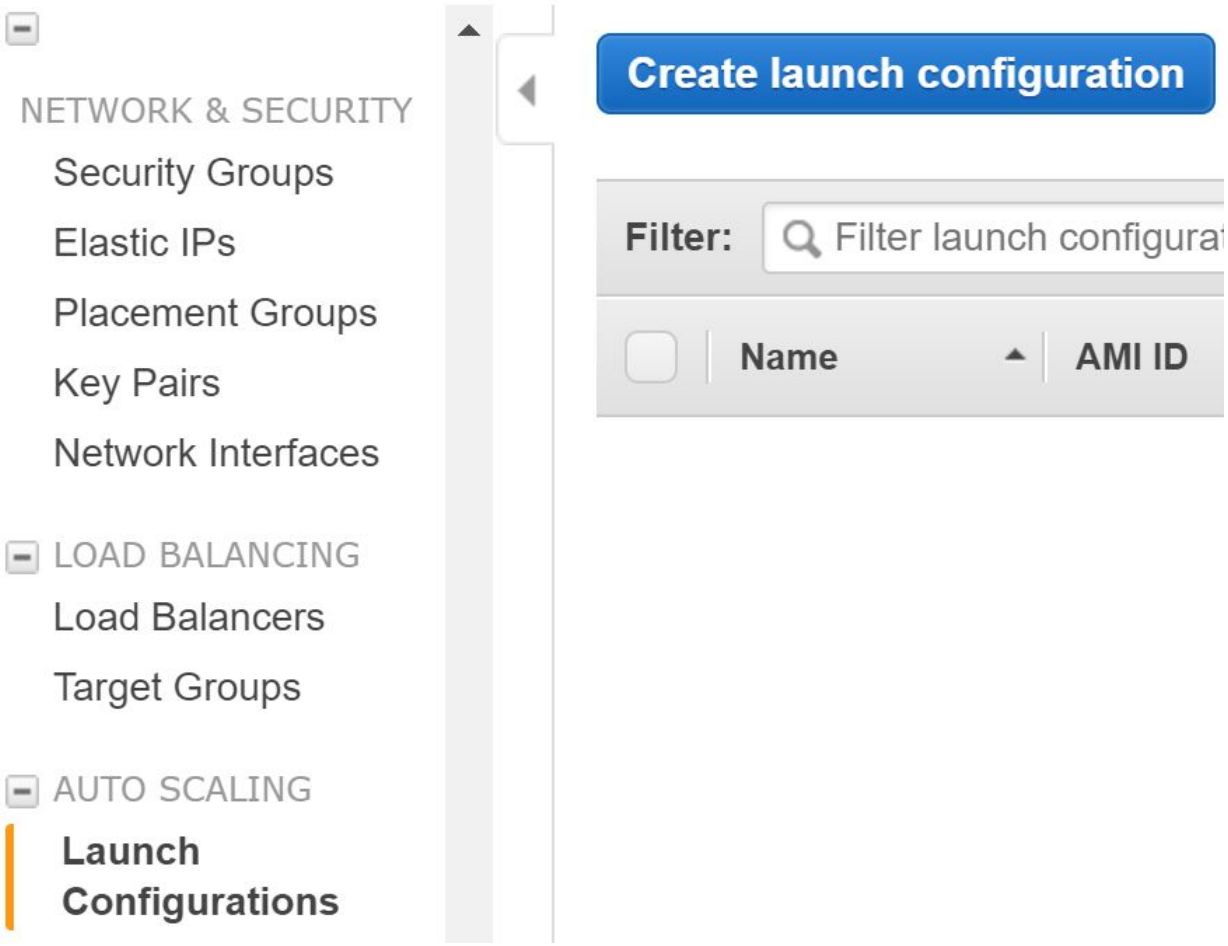
First you will create a Launch Configuration, then the Auto Scaling group.

Instructions

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



2. Open the **Launch Configurations** page and click the **Create launch configuration** button:

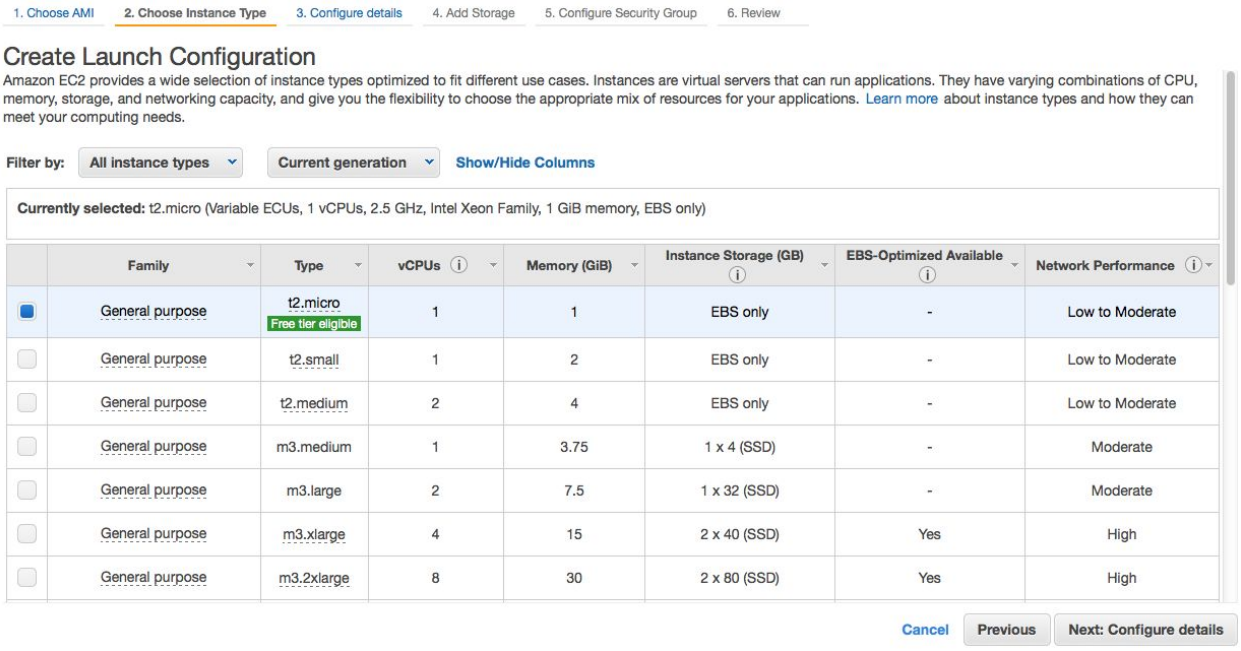


The Create launch configuration wizard starts.

3. 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 2 AMI:

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

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



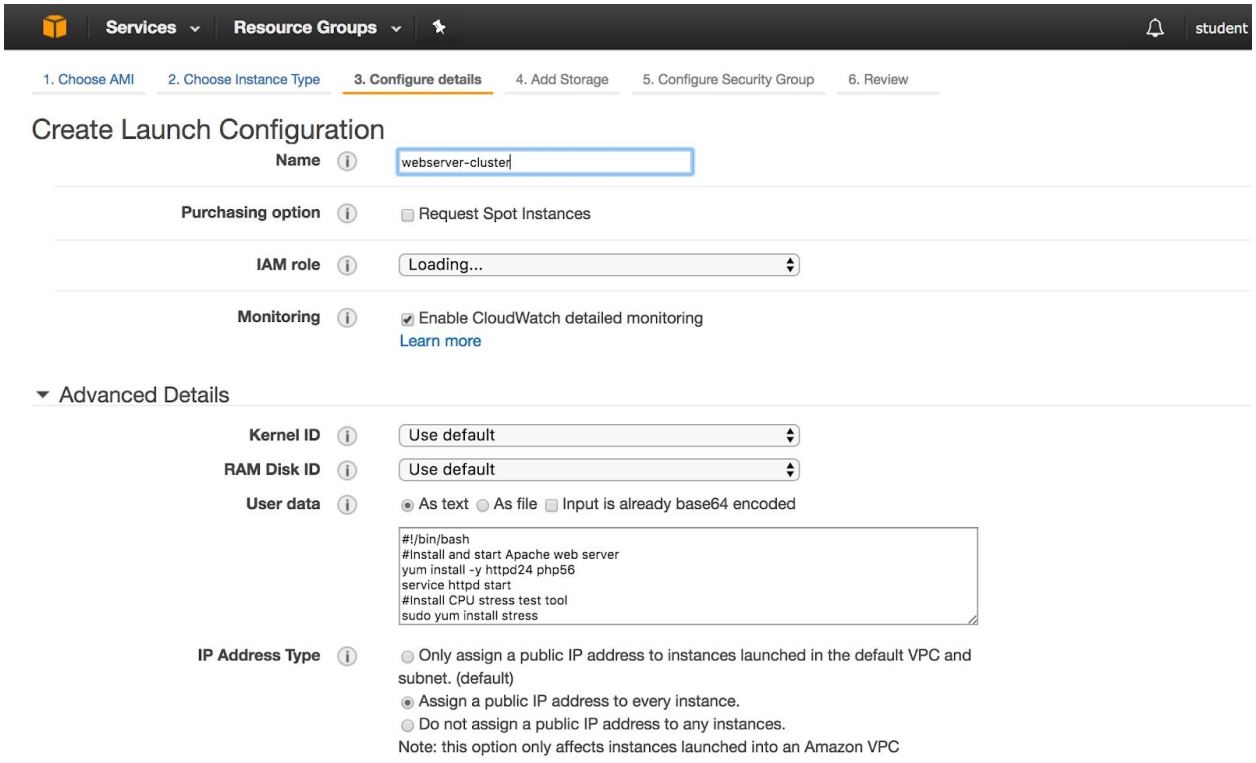
5. On the **Configure details** page:

- 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:

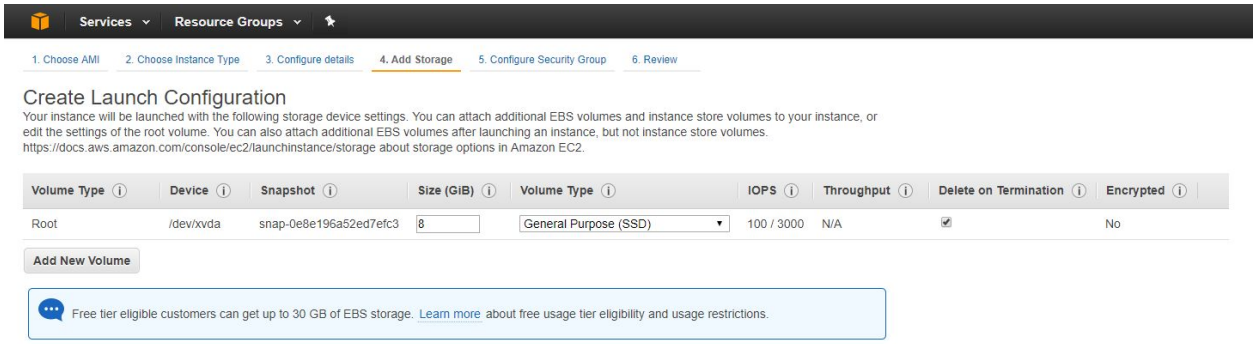
```
#!/bin/bash
#Enable the epel-release
amazon-linux-extras install epel
#Install and start Apache web server
yum install -y httpd php
service httpd start
#Install CPU stress test tool
sudo yum install -y stress
```

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.

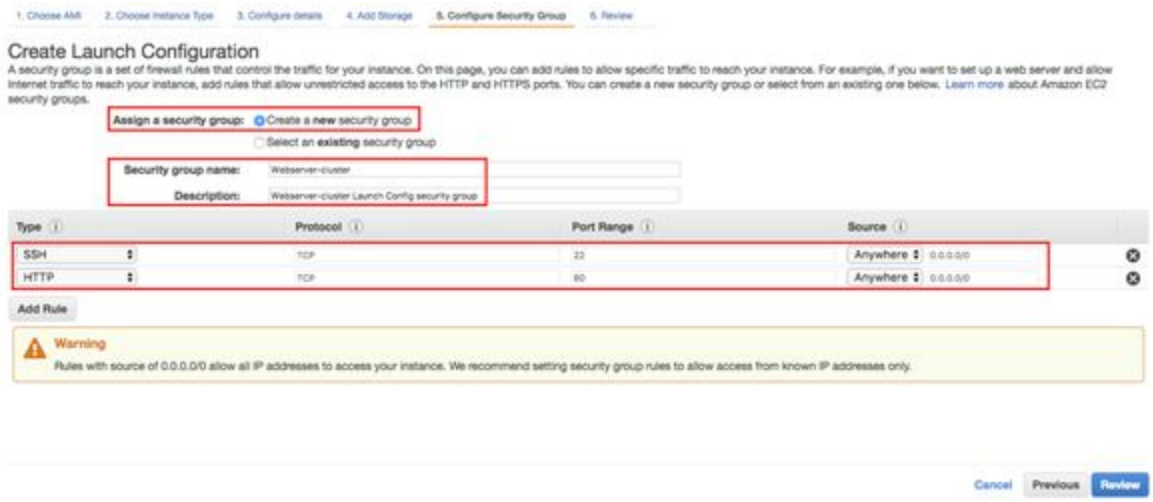
6. The **Add Storage** page of the wizard allows you to add or increment the size of any EBS volume attached to each EC2 instance started by the Auto Scaling group. Leave the defaults and do not add any EBS volumes.



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.

7. 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. The first rule is configured automatically:
    - **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.



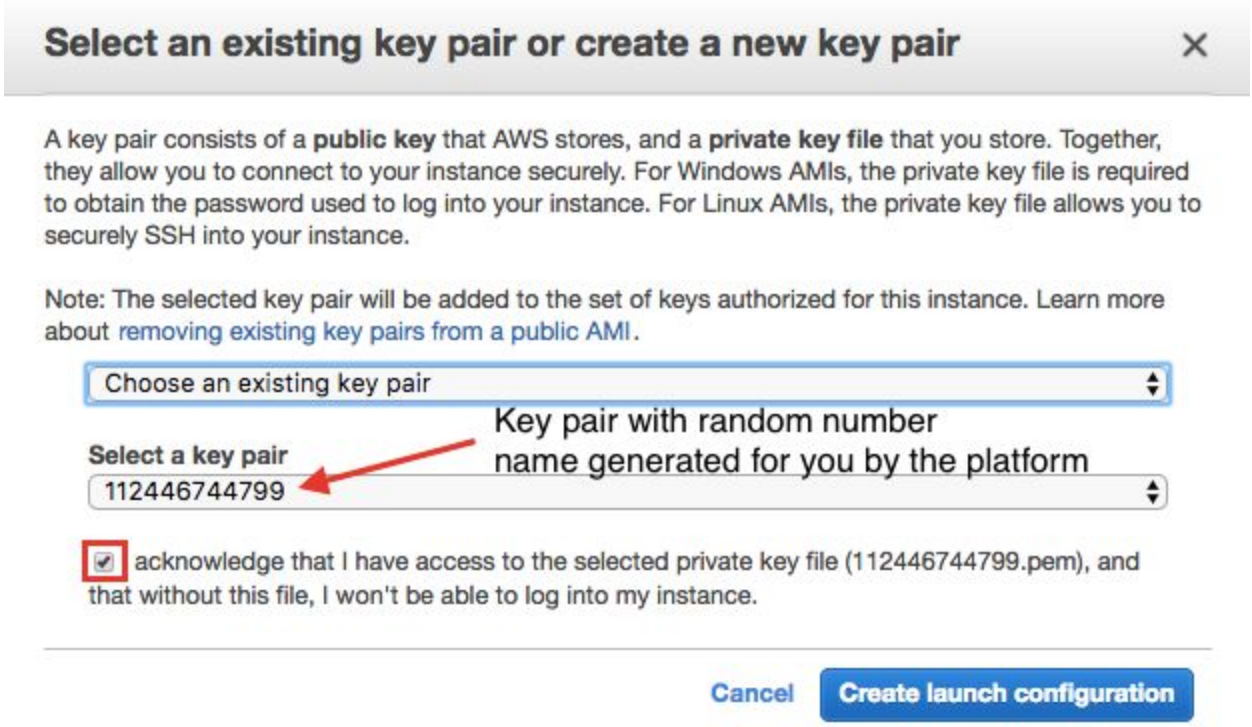
Click **Review** when ready.

8. 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.

9. 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.)

Introduction

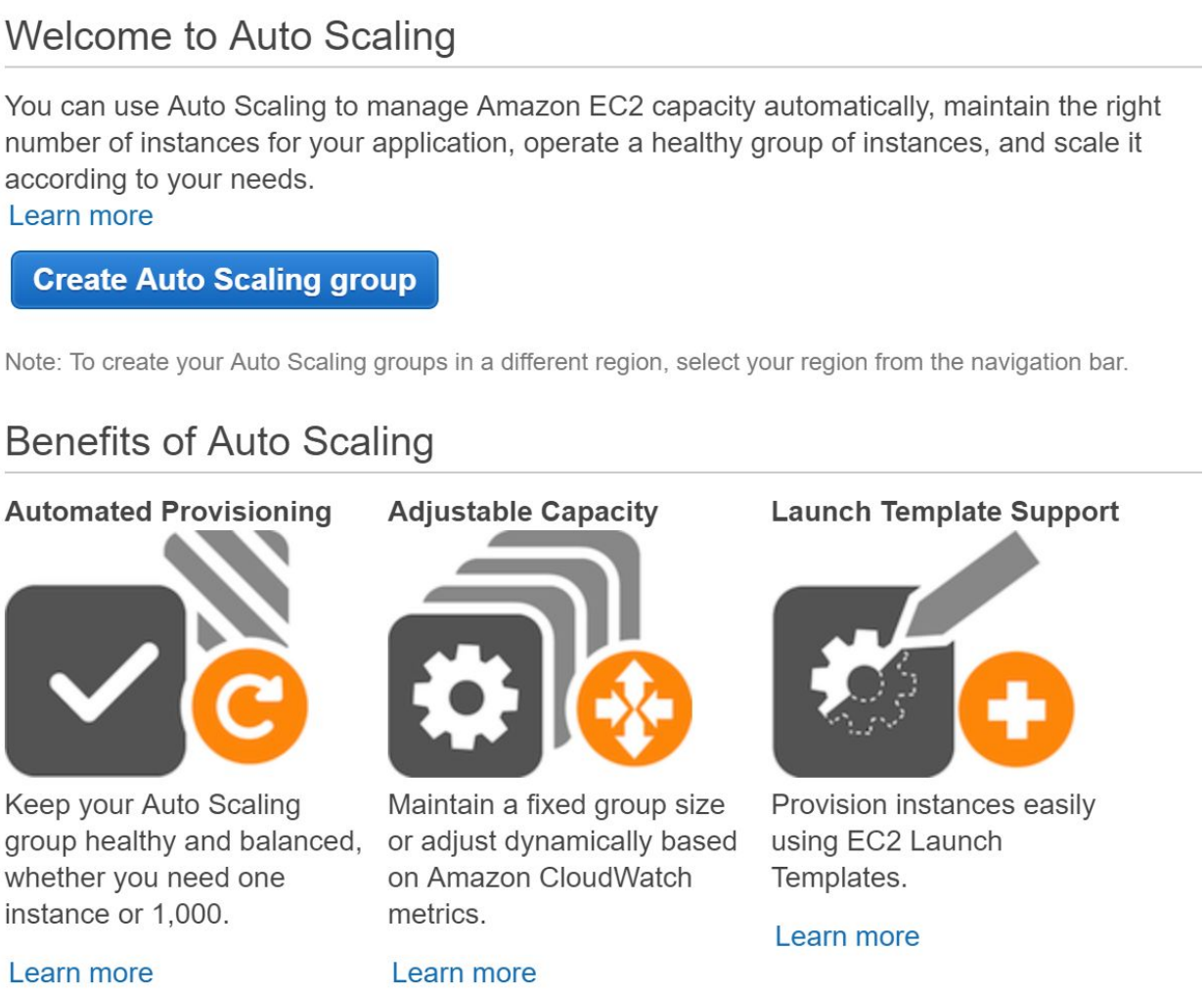
An Auto Scaling group is a representation of multiple Amazon EC2 instances that share similar characteristics and that are treated as a logical grouping for the purposes of instance scaling and management. For example, if a single application operates across multiple instances, you might want to increase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

When you create an Auto Scaling group, you can associate with a launch template or the older and less flexible launch configuration. You will use a launch configuration from the previous Lab Step in this Lab Step. Although launch configurations are older, they currently are the only option that allows the Lab to be completed without full EC2 privileges.

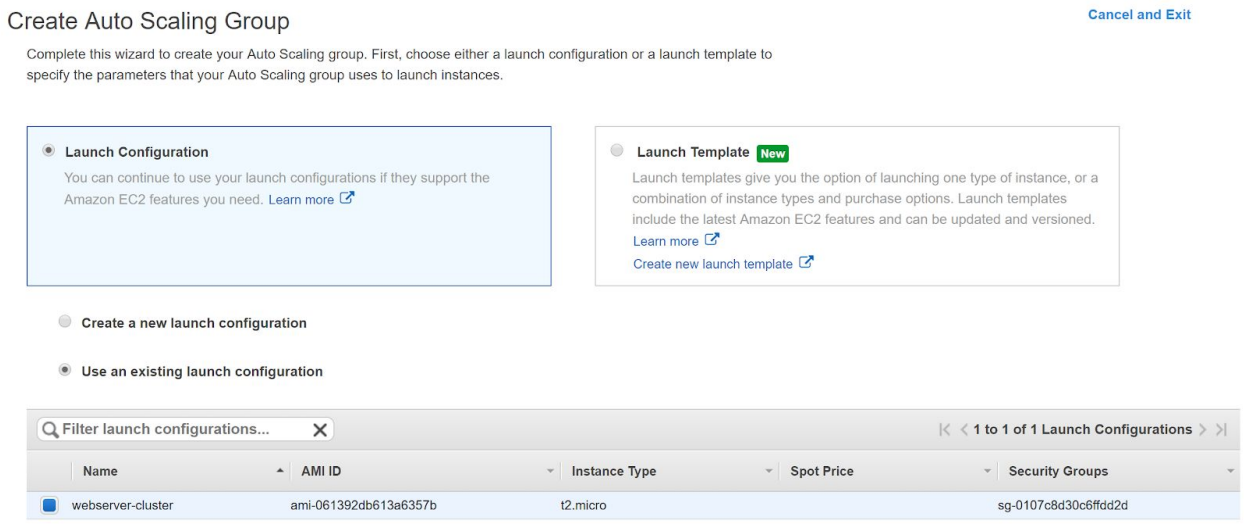
Instructions

1. Navigate to the **AUTO SCALING** > **Auto Scaling Groups** section of the EC2 console, then click **Create Auto Scaling group:**



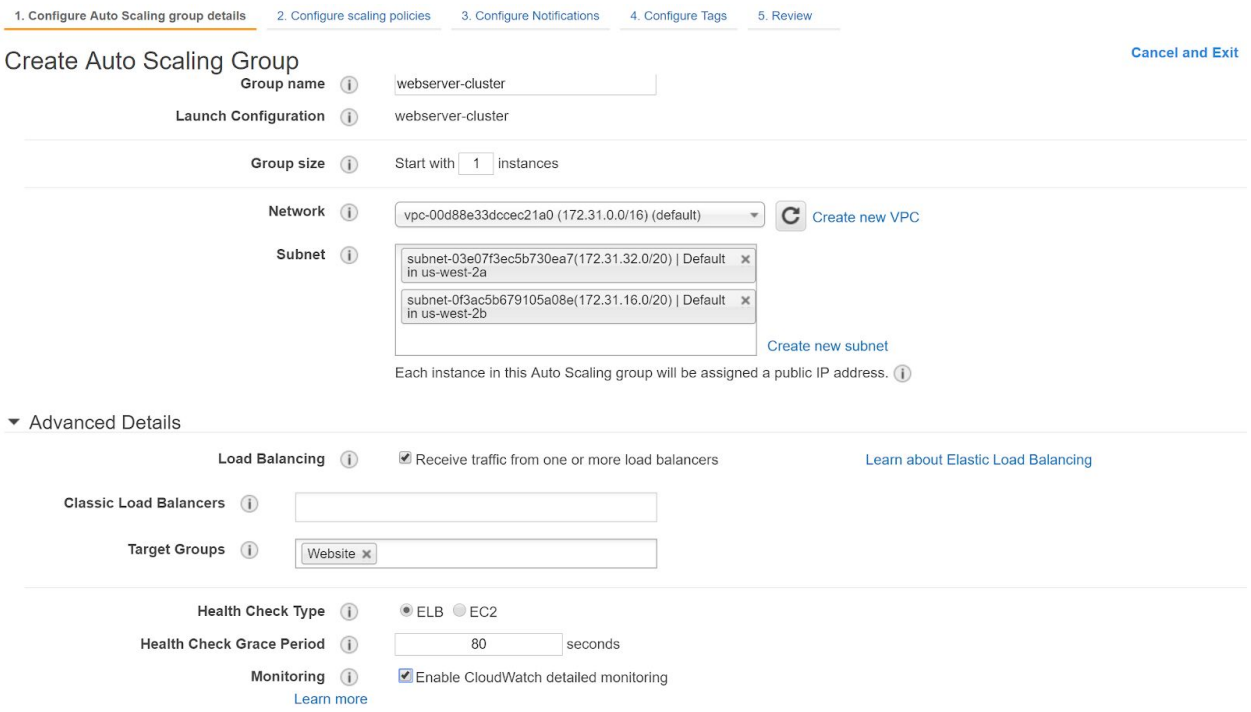
You will be able to associate the existing launch configuraiton with the Auto Scaling group you create here.

2. Select **Launch Configuration** and choose the **webserver-cluster** configuration before clicking **Next Step**:



3. On the **Configure Auto Scaling group details** step, set the following values leaving the defaults for the rest:

- **Group name**: *webserver-cluster*
- **Group size**: Start with *1* instance
- **Subnet**: Select both the **us-west-2a** and the **us-west-2b**
- **Advanced Details** (click the triangle to expand the section):
    - **Load Balancing**: Check **Receive traffic from one or more load balancers**
        - **Target Groups**: Select the **Website** target group made earlier when you created the Network Load Balancer
    - **Health Check Type**: ELB
    - **Health Check Grace Period**: *80* seconds
    - **Monitoring**: Check **Enable CloudWatch detailed monitoring**

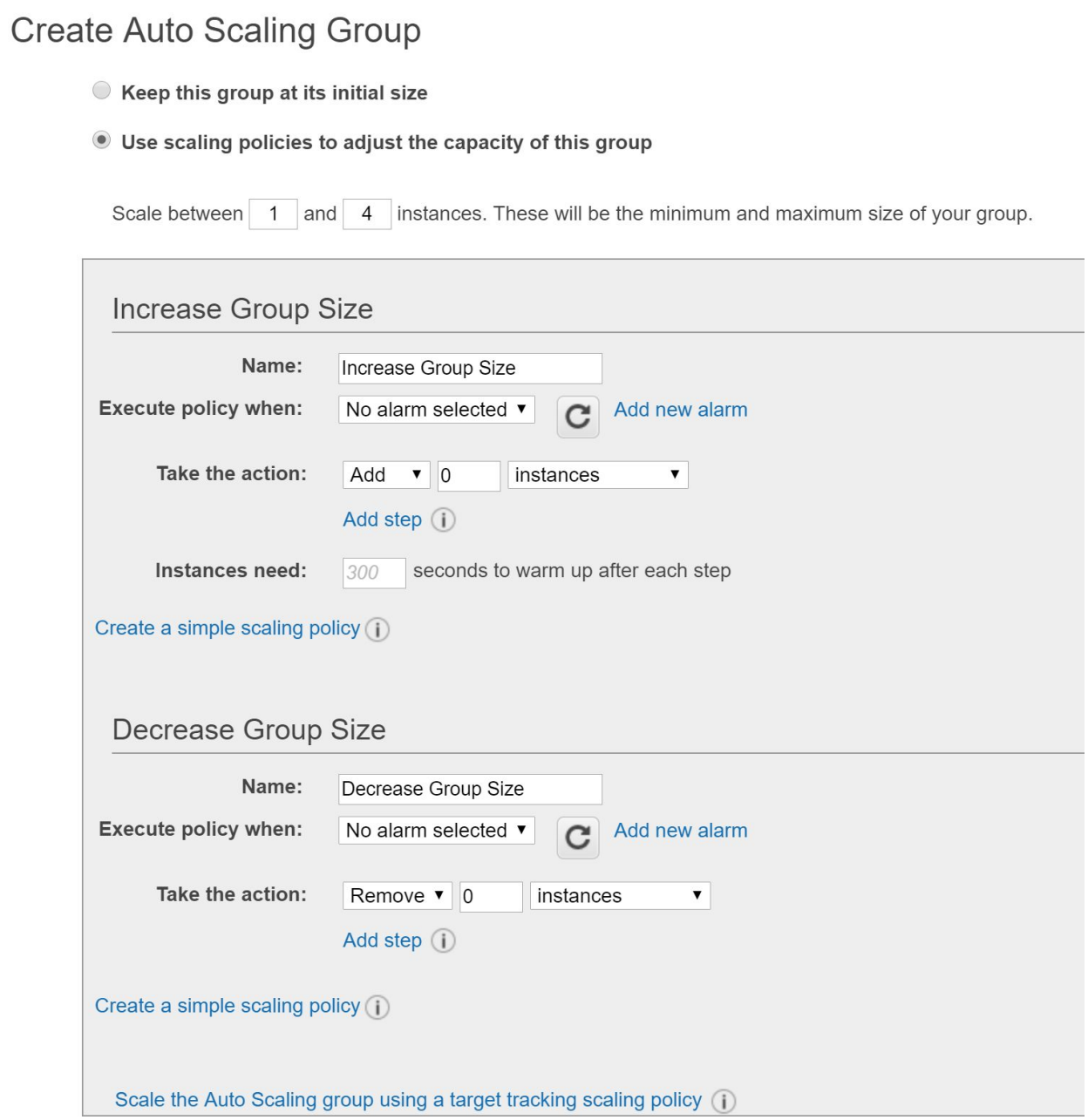


4. Click **Next: Configure scaling policies** when ready.

Scaling policies determine how and when your Auto Scaling group will scale up and scale down.

5. In the **Configure scaling policies** step, select **Use scaling policies to adjust the capacity of this group** and set the following values:

- Scale between *1* and *4* instances
- Click **Scale the Auto Scaling group using step or simple scaling policies** to display the **Increase Group Size** and **Decrease Group Size** sections:

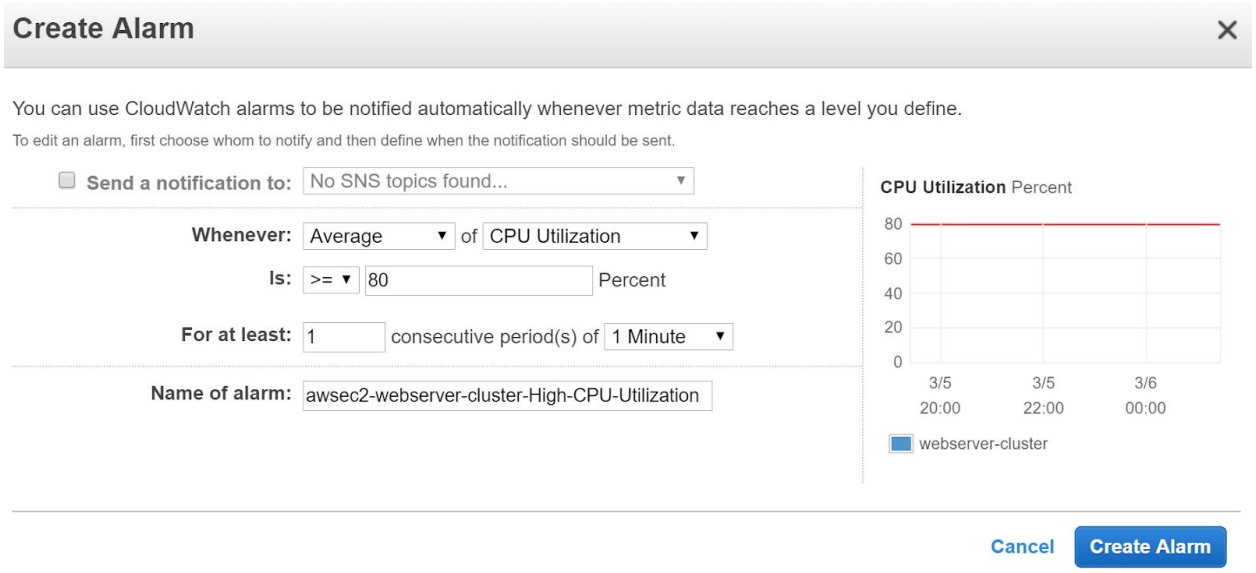


The Auto Scaling group policies allow you to automatically increase or decrease the group size based upon policies you define. In order to establish an **Increase Group Size** or **Decrease Group Size** policy, you must create a CloudWatch Alarm and then define which action should be taken if it is triggered.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

6. Click **Add new alarm** in the **Increase Group Size** section.

7. In the **Create Alarm** form, set the following values before clicking **Create Alarm**:

- **Send a notification to**: Uncheck (You can use SNS notifications to send emails and other notifications but you don't need to for this Lab)
- **is**: **>=** *80* Percent
- **For at least**: *1* consecutive period(s) of **1 Minute**
- **Name of alarm**: *awsec2-webserver-cluster-High-CPU-Utilization*

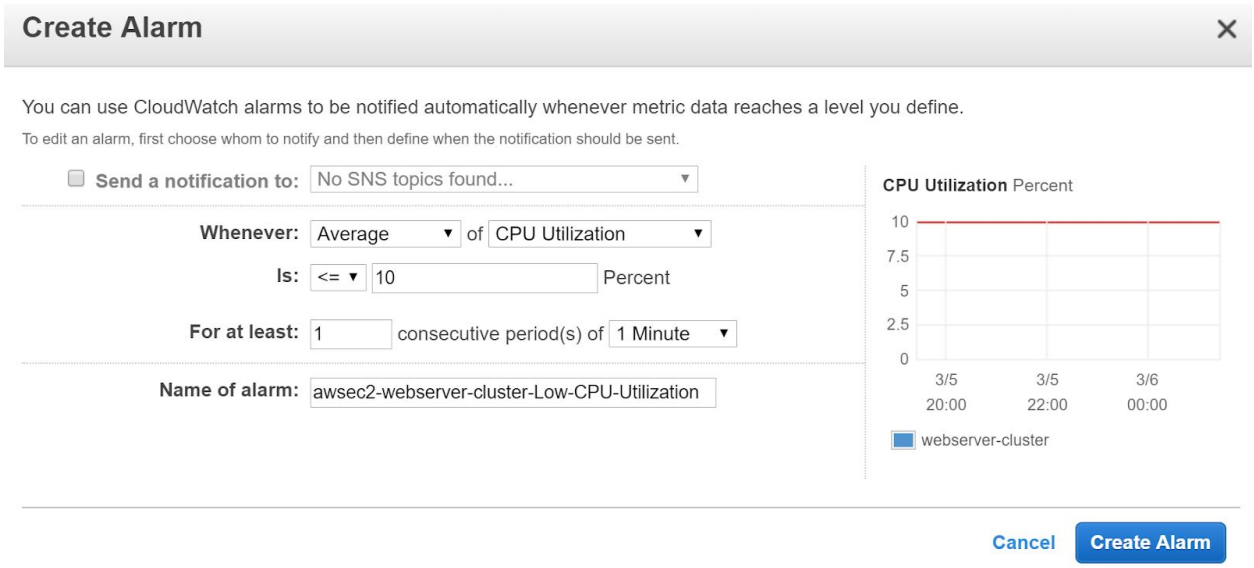


This will cause a scale up event to occur when the average CPU utilization of all instances in the Auto Scaling Group is over 80% for 1 minute. In practice, a 1 minute period may be too short causing scaling events to happen for intermittent traffic spikes. However, it will reduce the amount of waiting to produce scaling events for this Lab.

8. Click **Add new alarm** in the **Decrease Group Size** section.

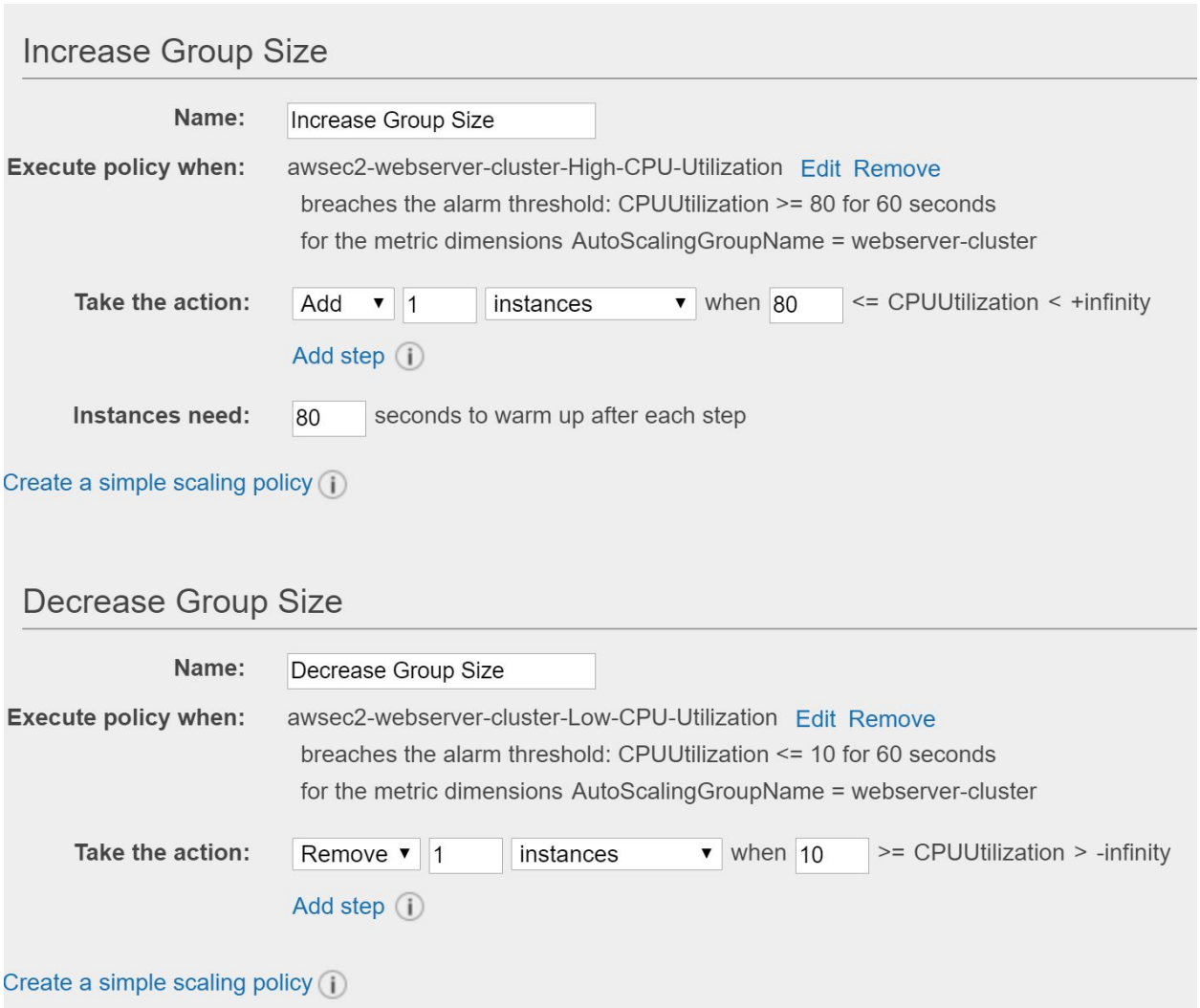
9. In the **Create Alarm** form, set the following values before clicking **Create Alarm**:

- **Send a notification to**: Uncheck
- **is**: **<=** 1*0* Percent
- **For at least**: *1* consecutive period(s) of **1 Minute**
- **Name of alarm**: *awsec2-webserver-cluster-Low-CPU-Utilization*



10. In the **Create Auto Scaling Group**, set the following remaining values:

- **Increase Group Size**:
  - **Take the action**: **Add** *1* **instances** when *80* <= CPUUtilization
  - **Instances need**: *80* seconds to warm up after each step
- **Decrease Group Size**:
  - **Take the action**: **Remove** *1* **instances** when *10* >= CPUUtilization

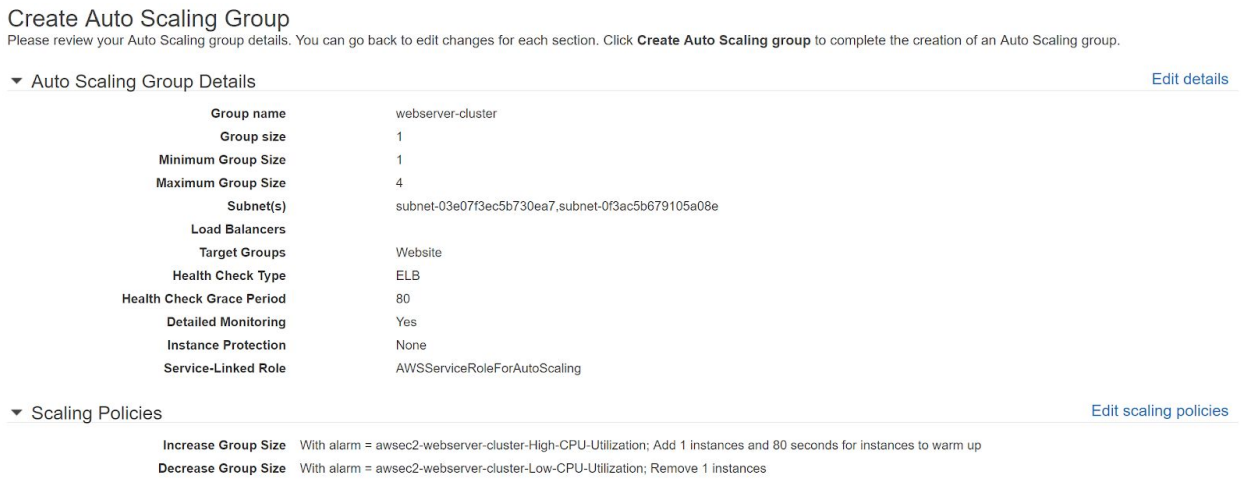


With this configuration, single instances will be added/removed with each scale event. The scaling can also add/remove a percentage of the instances in the Auto Scaling group rather than a fixed amount or set to a specific value.

11. Click **Review**.

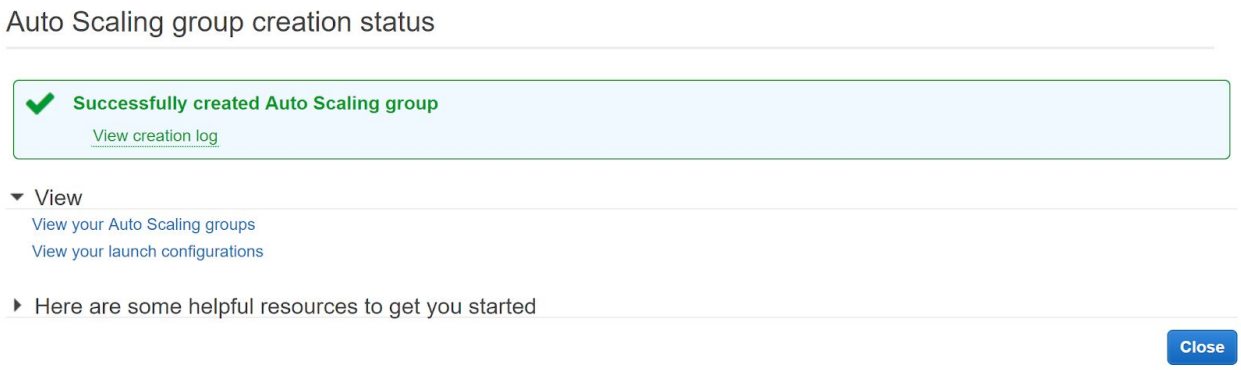
You will not make use of notifications in this Lab.

12. Review the configuration and then click **Create Auto Scaling group**:



You can always click **Previous** to return to previous steps if you notice a mistake.

13. Click **Close** on the success notification view:

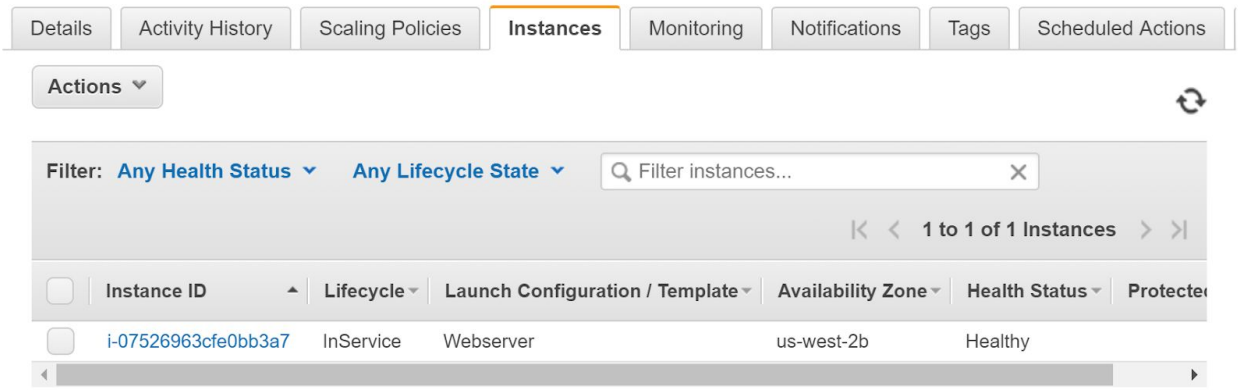


14. With the **webserver-cluster** Auto Scaling group selected, click **Actions** > **Edit**:

15. Change the **Default Cooldown** to *120* seconds and click **Save**:

The cooldown is the minimum delay between scale events.

16. Click the **Instances** tab for the Auto Scaling group and check that there is an instance displayed in the table:

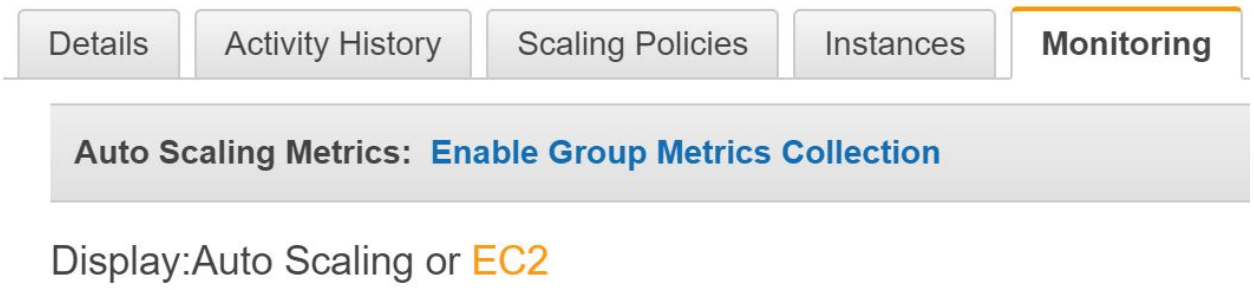


Note: You may need to click the refresh icon to update the table if the instance does not appear.

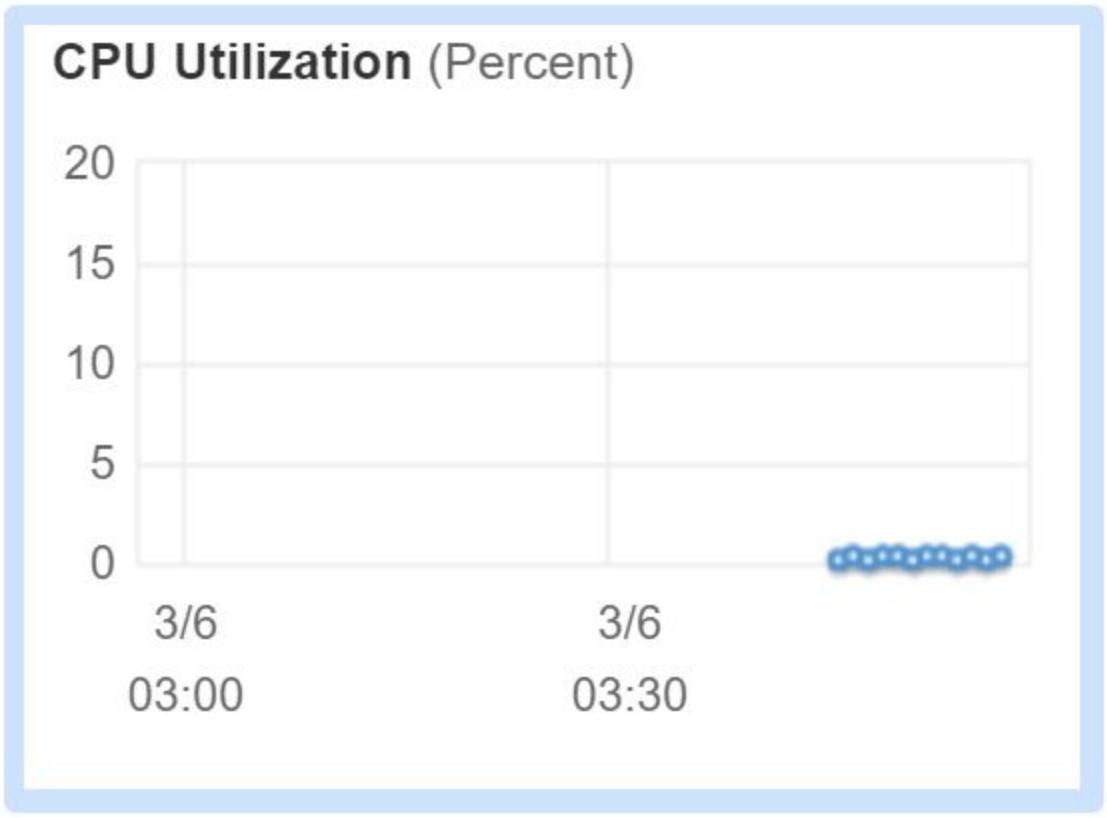
The Auto Scaling group has satisfied the desired instance count requirement you set of 1. Also, because the web server is not CPU-intensive and there is no load on the web server, the high

CPU alarm won't trigger. The minimum number of instances you set for the Auto Scaling group is also 1. That means the number of instances will stay at 1 unless something changes.

17. Click the **Monitoring** tab followed by **EC2**:



18. Observe the different metrics recorded, but focus on the **CPU Utilization** chart and observe it is near zero (**0**):

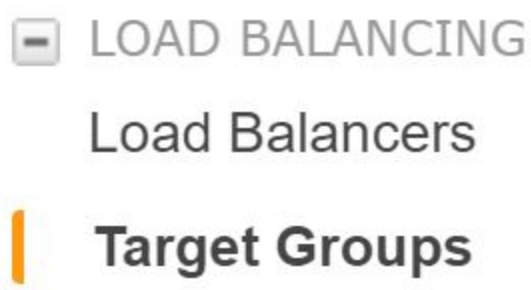


Note: If no data points display, click the refresh icon after a minute to update the chart

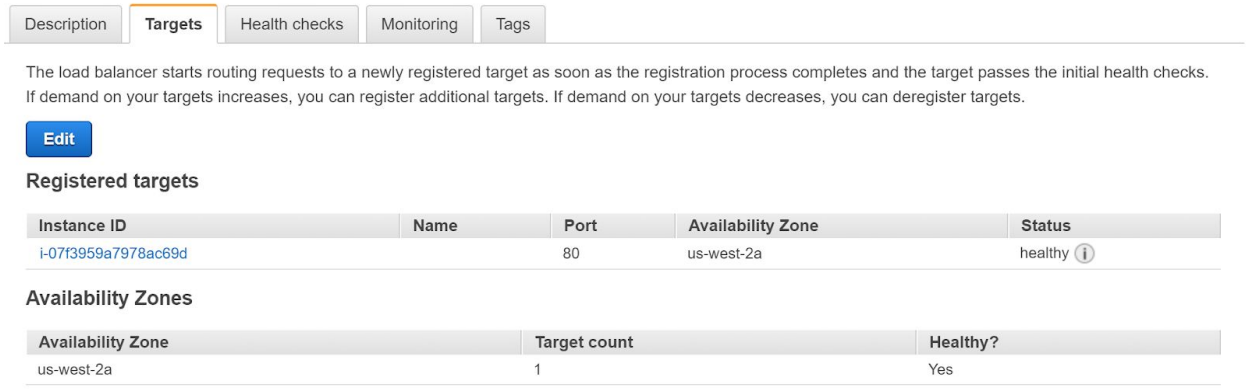
You can also double-click the chart for a zoomed in view.

The current CPU utilization would cause a scale in event if the Auto Scaling group was not already at the minimum number of instances, i.e. the scale in alarm is being triggered because CPU utilization is below 10%.

19. Navigate to the **LOAD BALANCING** > **Target Groups** section of the EC2 Console:



20. Select the **Website** target group, and then open the **Targets** tab:



Note: You may need to click the refresh icon to update the table if the instance is in the **initial** status while the load balancer waits for three successful health checks before assigning a healthy status.

Observe there is an instance added to the **Registered targets** and it is indeed the same instance created by the Auto Scaling group. Also, notice the **Status** is **healthy** meaning the instance is

reachable on TCP port 80 (HTTP). That implies the launch configuration's user data script successfully completed to start the Apache web server on the instance. Everything appears to be working. You will perform more thorough tests in the next Lab Step.

Summary

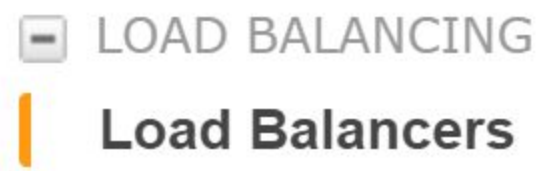
In this Lab Step, you created an Auto Scaling group using a launch configuration. You defined a scaling policy to scale up or down based on the average CPU utilization of all the instances in the Auto Scaling group. The scaling policy makes use of CloudWatch metrics to trigger an alarm to cause a scale in or scale out event. You also configured the Auto Scaling group to automatically register its instances to a target group of a Network Load Balancer.

Introduction

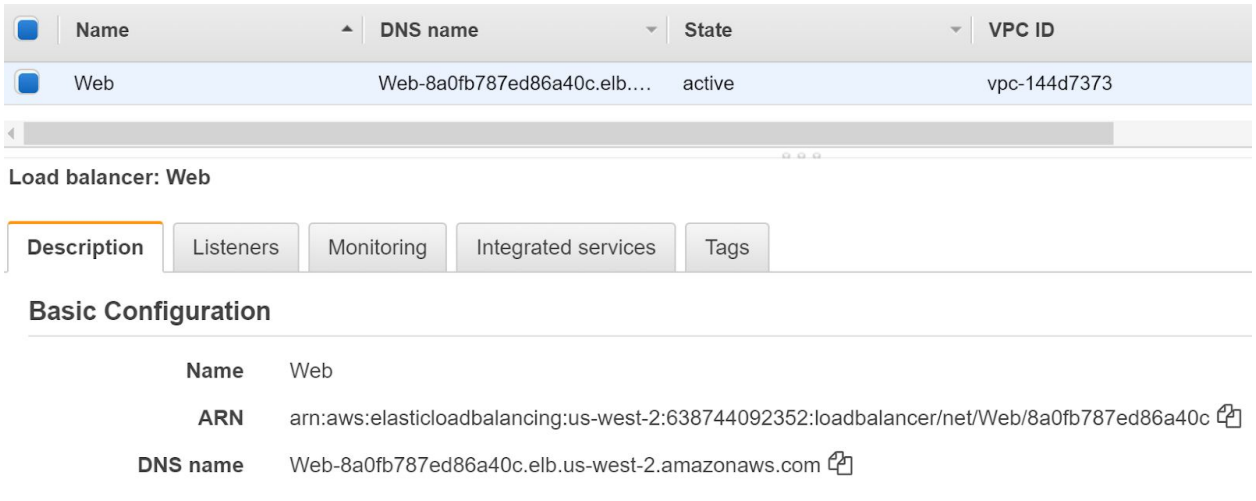
Performing end-to-end tests to make sure everything is working as you think it should is very important. Although this may be an automated procedure, often a quick sanity test by other individuals and/or groups directly from the AWS Console is also helpful. This Lab Step will point out a few ways to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

Instructions

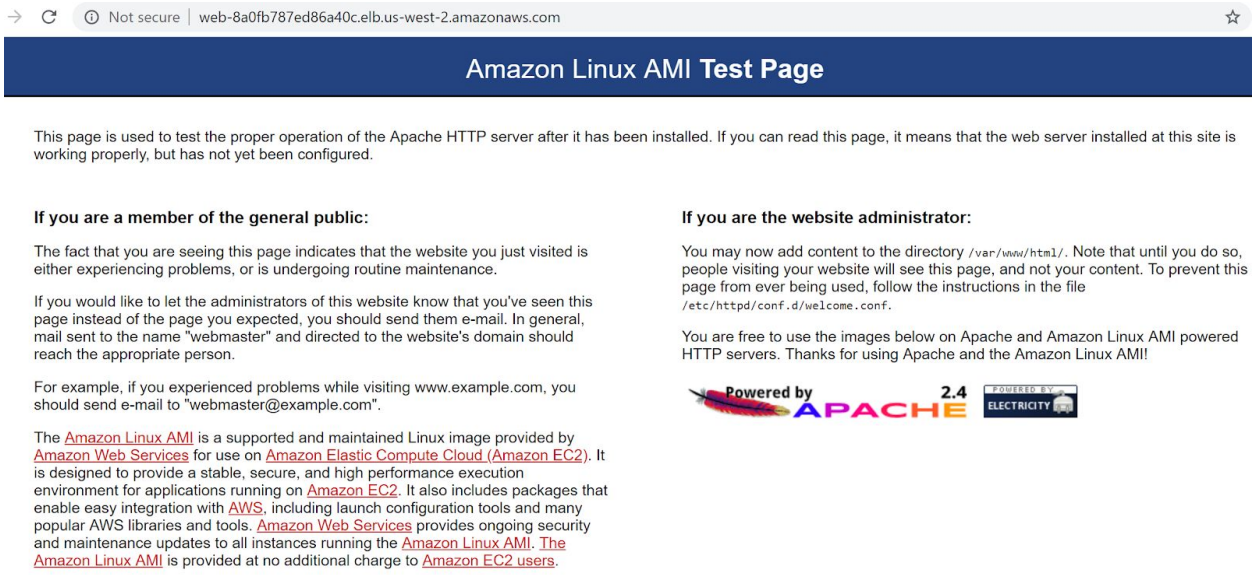
1. Navigate to the **LOAD BALANCING** > **Load Balancers** section of the EC2 Console:



2. Copy the **DNS name** of the **Web** load balancer and navigate to it in a new browser tab:



You should see the following default Apache web server page:



If the page were not displayed, there are several places you could check to troubleshoot the issue starting with the following:

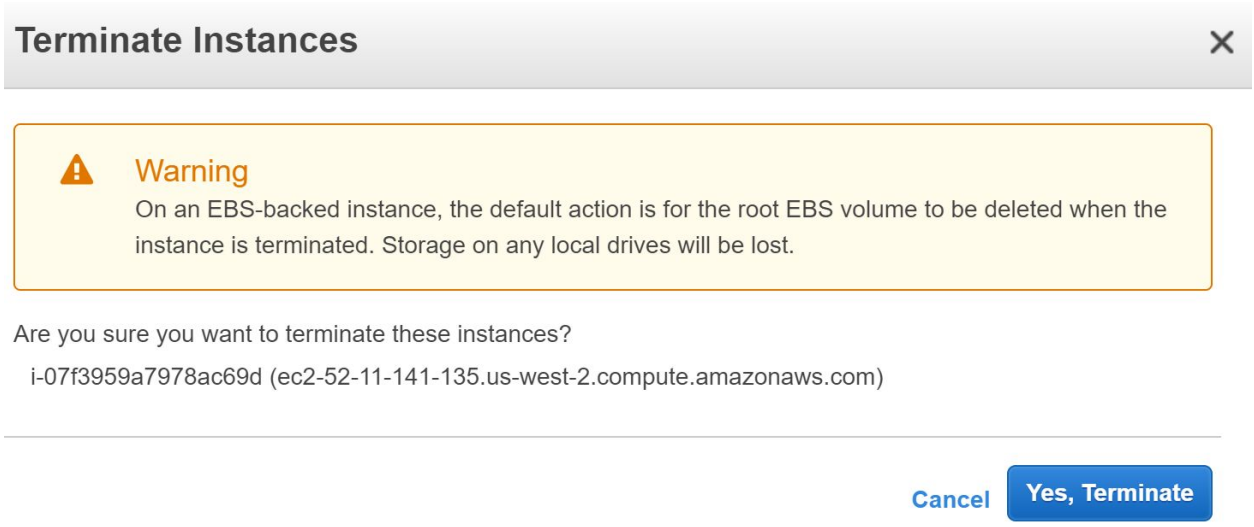
- Ensure the security groups of the load balancer and the instances allows HTTP ingress traffic
- Ensure the user data script in the launch template correctly installs and runs the Apache web server
- Ensure the Auto Scaling group is configured to add its instances to the load balancer's target group
- Ensure the health checks are configured for TCP port 80 otherwise the instances will never reach a healthy status and will be terminated and then replaced with a new

instance by the Auto Scaling group. The new instance will subsequently never reach a healthy status and be replaced, and the process repeats.

- To allow for you to debug the instances without having them be replaced, you can block instance termination by performing the following steps:

    - Navigate to **Auto Scaling Groups** > **Actions** > **Edit**
    - Set the **Suspended Processes** to **Terminate** (This will prevent instances in your group from getting terminated. Don't forget to remove the configuration once the issue is resolved.)

3. From the INSTANCES > **Instances** section of the EC2 Console, click **Actions** > **Instance State** > **Terminate** and click **Yes, Terminate** to the confirmation dialog:



The Auto Scaling group should detect the change and relaunch an instance automatically to meet the minimum desired capacity of 1.

4. Wait around 30 seconds and click the refresh icon to see the new instance launch and settle into a running state:

5. Connect to the running instance using the PEM (macOS/Linux) or PPK (Windows) key file in the **Your lab data** of this Lab.

Tip: You can click the **Connect** button for a refresher on how to SSH into the instance:

If you are using Window click the **connect using PuTTY** link in the dialog.

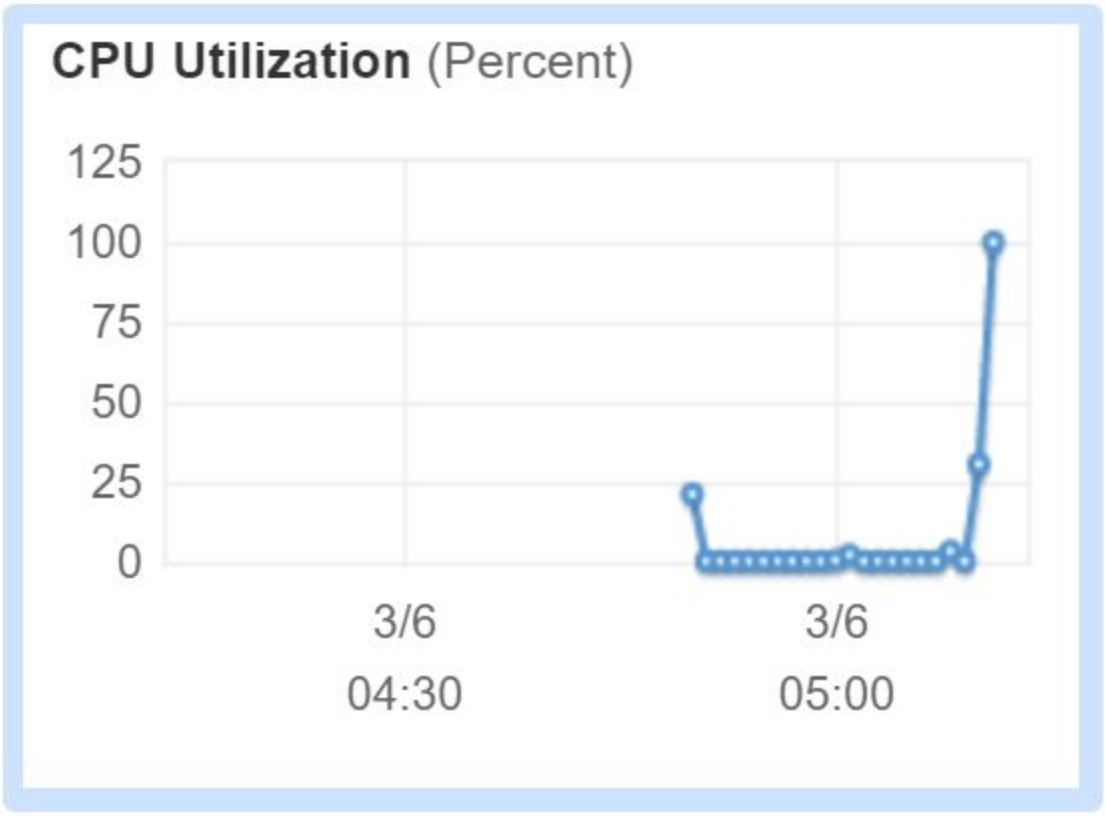
6. Enter the following command at the command line to run stress causing the CPU utilization to increase for five minutes:

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

```
stress: info: [3125] dispatching hogs: 2 cpu, 1 io, 1 vm, 0 hdd
```

You can enter man stress for more information about stress.

7. Navigate to Auto Scaling group's **Monitoring** tab and click **EC2** to view the **CPU Utilization**.



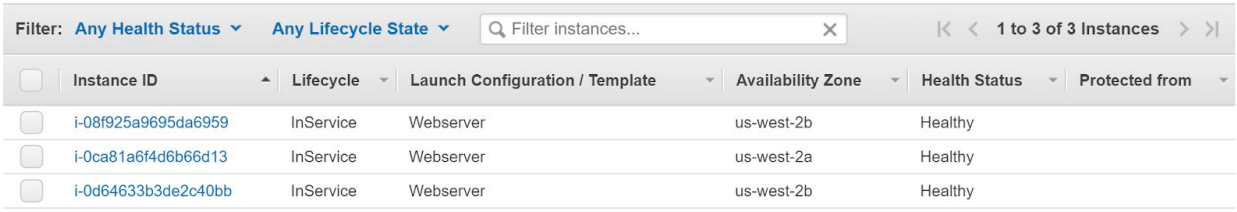
You can click the refresh button to update the chart after a minute. The chart should clearly show the CPU Utilization is at 100%.

8. Click the **Activity History** tab and observe that new instances have been created:

Status	Description
Waiting for instance warmup	Launching a new EC2 instance: i-08f925a9695da6959
Successful	Launching a new EC2 instance: i-0d64633b3de2c40bb
Successful	Launching a new EC2 instance: i-0ca81a6f4d6b66d13
Successful	Terminating EC2 instance: i-07f3959a7978ac69d
Successful	Launching a new EC2 instance: i-07f3959a7978ac69d

Depending on how long it has been since stress started running, you may see more or less rows.

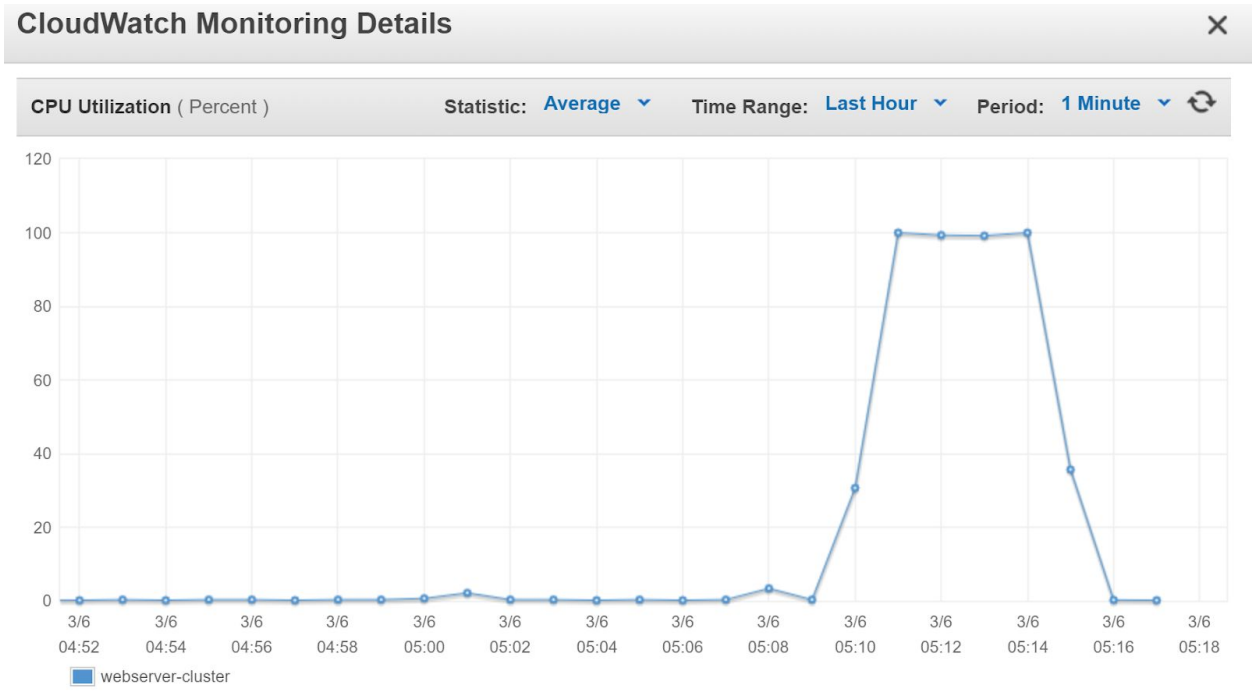
9. Click the **Instances** tab to confirm that a new instance has been created in response to the increased CPU utilization:



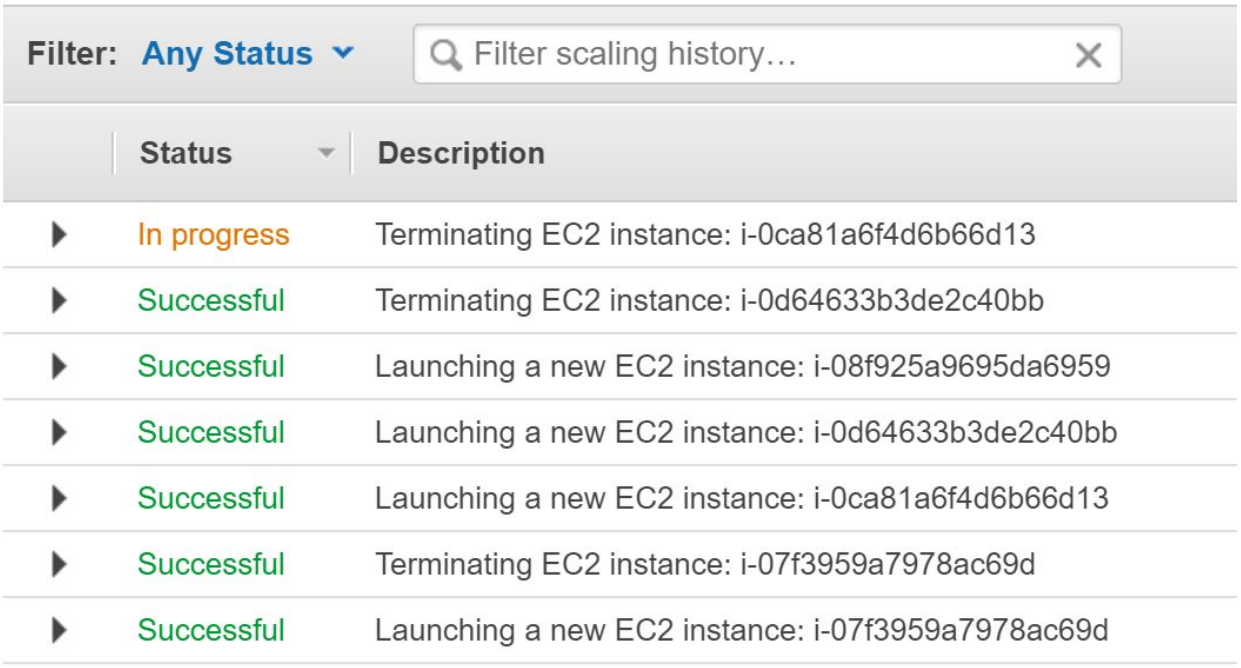
Because of the Auto Scaling timing configuration, two new instances get created rather than one. One is enough to drop the average CPU utilization to 50% but the high frequency of the scale up alarm allows for a second instance to be created before the average CPU utilization can drop below the 80% threshold.

You can also confirm the instances are added to the target group.

10. Wait until stress completes its five minute run and the average CPU utilization drops near zero:

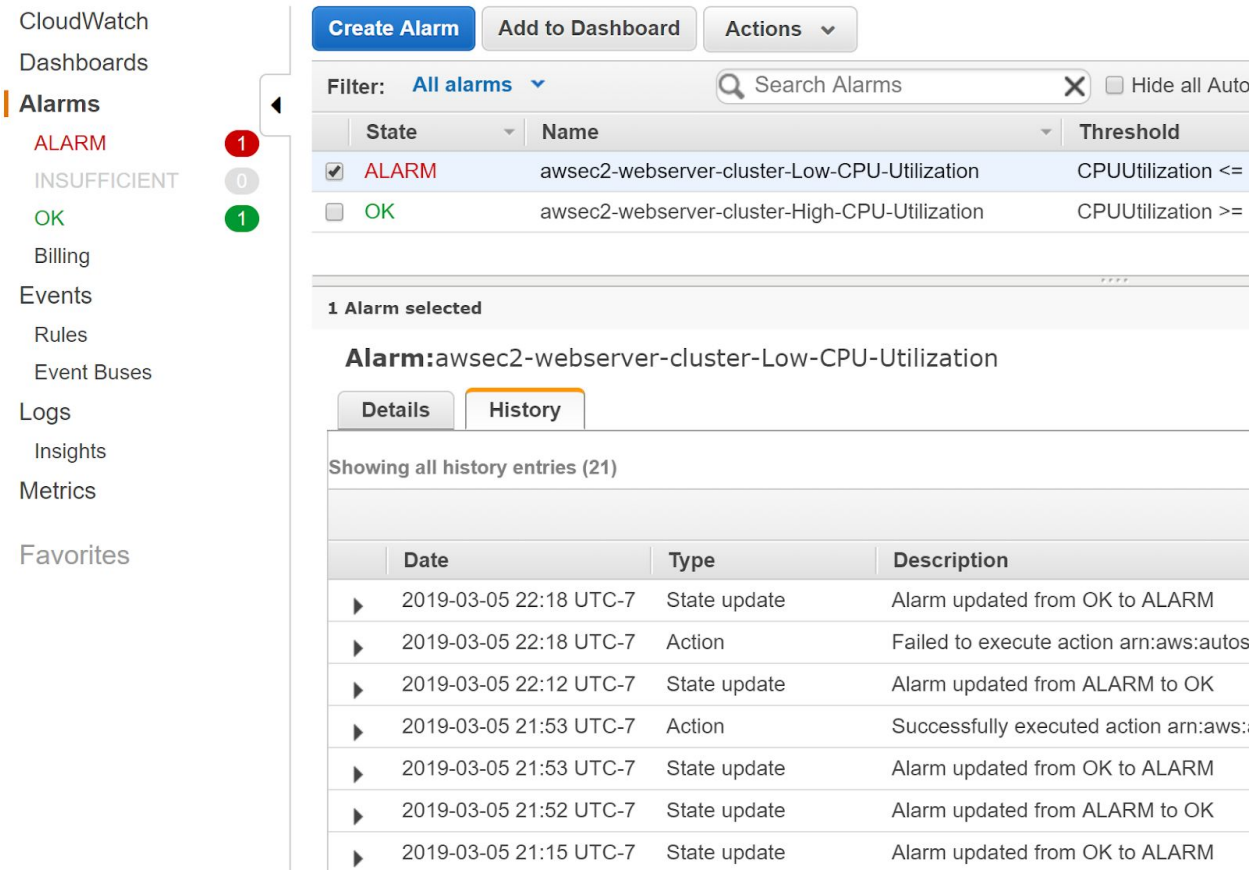


11. Return to the **Activity History** tab and observe instances beginning to terminate according to the scale in policy (CPU utilization <= 10%):



Eventually, the number of instances drops down to the minimum of 1.

12. Navigate to the CloudWatch Alarms Console and inspect the history of the alarms to see what triggered the Auto Scaling group scale events:



Auto Scaling groups integrate well with CloudWatch, but you can view even more details from the CloudWatch Console.

Summary

In this Lab Step, you performed several tests of the Auto Scaling group, launch template, and Network Load Balancer system. You also learned where to look when something doesn't work as expected with Auto Scaling groups.