Name: _____ Username: _____

Lab 8

# Elastic Load Balancing, Auto Scaling and CloudWatch

## Submission

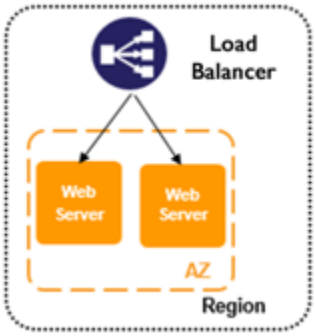
Five submission requirements (screenshots and data) are spread throughout this document. Submit via **private Piazza Post by 10:00 pm tonight** (Tuesday February 7th). Late submissions will not be accepted and a timestamp of 10:01 pm is a late submission.

## Overview

This lab introduces the concept of Elastic Load Balancing. In this lab you will use Elastic Load Balancing to load balance traffic across multiple Amazon Elastic Compute Cloud (EC2) instances in a single Availability Zone. You will deploy a simple application on multiple Amazon EC2 instances and observe load balancing by viewing the application in your browser.

First, you will launch a pair of instances, bootstrap them to install web servers and content, and then access the instances independently using Amazon EC2 DNS records. Next, you will set up Elastic Load Balancing, add your instances to the load balancer, and then access the DNS record again to watch your requests load balance between servers. Finally, you will view Elastic Load Balancing metrics in Amazon CloudWatch.

The following diagram provides a high-level overview of the architecture you will implement in this exercise.




This lab also shows you how to set up Auto Scaling to automatically launch compute instances of your choosing in response to conditions that you specify. In this lab, you will use Auto Scaling via the AWS console to set up the basic infrastructure of a Launch Configuration and an Auto Scaling group. You will test the configuration by terminating a running instance and viewing the results as Auto Scaling starts another instance.

## Topics covered

This lab will take you through:

Name: _____ Username: _____

- Launching a multiple server web farm on Amazon EC2.
- Using bootstrapping techniques to configure Linux instances with Apache, PHP, and a simple PHP application downloaded from Amazon Simple Storage Service (S3).
- Creating and configuring a load balancer that will sit in front of your Amazon EC2 web server instances.
- Viewing Amazon CloudWatch metrics for the load balancer.
- Create an Auto Scaling Launch Configuration
- Create an Auto Scaling group
- Test the Auto Scaling Infrastructure
- View the results of the Auto Scaling launch

## What is Elastic Load Balancing?

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

Achieve higher levels of fault tolerance for your applications by using Elastic Load Balancing to automatically route traffic across multiple instances and multiple Availability Zones. Elastic Load Balancing ensures that only healthy Amazon EC2 instances receive traffic by detecting unhealthy instances and rerouting traffic across the remaining healthy instances. If all of your Amazon EC2 instances in one Availability Zone are unhealthy, and you have set up Amazon EC2 instances in multiple Availability Zones, Elastic Load Balancing will route traffic to your healthy Amazon EC2 instances in those other zones.

Elastic Load Balancing automatically scales its request-handling capacity to meet the demands of application traffic. Additionally, Elastic Load Balancing offers integration with Auto Scaling to ensure that you have back-end capacity to meet varying levels of traffic levels without requiring manual intervention.

Elastic Load Balancing works with Amazon Virtual Private Cloud (VPC) to provide robust networking and security features. You can create an internal (non-Internet facing) load balancer to route traffic using private IP addresses within your virtual network. You can implement a multi-tiered architecture using internal and Internet-facing load balancers to route traffic between application tiers. With this multi-tier architecture, your application infrastructure can use private IP addresses and security groups, allowing you to expose only the Internet-facing tier with public IP addresses.

Elastic Load Balancing provides integrated certificate management and SSL decryption, allowing you to centrally manage the SSL settings of the load balancer and offload CPU-intensive work from your instances.

This lab guide explains basic concepts of Elastic Load Balancing in a step-by-step fashion. However, it can only give a brief overview of Elastic Load Balancing concepts. For further information, see http://aws.amazon.com/elasticloadbalancing/.

Name: _____ Username: _____

## Launch Web Servers

In this section, you will launch two Amazon Linux EC2 instances, with an Apache PHP web server and basic application installed on initialization. You will also demonstrate a simple example of bootstrapping instances using the Amazon EC2 metadata service.

## Amazon Machine Images (AMIs) and instances

Amazon EC2 provides templates known as *Amazon Machine Images (AMIs)* that contain a software configuration (for example, an operating system, an application server, and applications). You use these templates to launch an *instance*, which is a copy of the AMI running as a virtual server in the cloud.

You can launch different types of instances from a single AMI. An *instance type* essentially determines the hardware capabilities of the virtual host computer for your instance. Each instance type offers different compute and memory capabilities. Select an instance type based on the amount of memory and computing power that you need for the application or software that you plan to run on the instance. You can launch multiple instances from an AMI.

Your instance keeps running until you stop or terminate it, or until it fails. If an instance fails, you can launch a new one from the AMI.

When you create an instance, you will be asked to select an instance type. The instance type you choose determines how much throughput and processing cycles are available to your instance.

- On the **Services** menu, click **EC2**.
- Click **Launch Instance**.
- In the row for the **Amazon Linux AMI**, which will normally be the first option on the list, click **Select**.
- The **t2.micro** instance type, which is the lowest-cost option, should be automatically selected. Click **Next: Configure Instance Details**.
- In the **Number of instances** box, type **2**.
- Ensure **Auto-assign Public IP** is set to **Enable**.
- Expand the **Advanced Details** section.
  In this section, you will use the **User data** field to bootstrap your instance, running a custom script to install the software packages (Apache and PHP) and sample code (PHP scripts) needed for this lab. User data provides a mechanism to pass data or a script to the Amazon metadata service, which instances can access at launch time.
- Copy the following initialization script:

```
#!/bin/sh
curl -L https://us-west-2-aws-training.s3.amazonaws.com/awsu-spl/spl03-working-elb/static/bootstrap-elb.sh | sh
```

- Copy the above script and paste it into the **User data** box with the **As text** option selected. This will automatically install and start the Apache Web server and other

components when the instance is created and launched.
**Tip** If you type this text instead of copying it, press SHIFT+ENTER to create new lines in the text box.

- Click **Next: Add Storage**.
- Click **Next: Tag Instance** to accept the default storage device configuration.
- In the **Value** box, type a name for your instance **<user-id>_lab8**.
- Click **Next: Configure Security Group**.

Now you will create a new security group. A *security group* acts as a firewall that controls the traffic allowed into a group of instances. When you launch an Amazon EC2 instance, you can assign it to one or more security groups. For each security group, you add rules that govern the allowed inbound traffic to instances in the group. All other inbound traffic is discarded. You can modify rules for a security group at any time. The new rules are automatically enforced for all existing and future instances in the group.

- Leave **Create a new security group** selected
- In the **Security group name** box, type the name **<user-id>_lab8_SG**.
- (Optional) Add a description for your security group.
  **Note** By default, AWS creates a rule that allows Secure Shell (SSH) access from any IP address. It is *highly recommended* that you restrict terminal access to the ranges of IP addresses (e.g., IPs assigned to machines within your company) that have a legitimate business need to administer your Amazon EC2 instance. Because the lab image you are using will be recycled within two hours, you can bypass this step for the lab.
- Click **Add Rule** to open a new port.
- In the **Type** drop-down list, click **HTTP**.
  This will add a default handler for HTTP that will allow requests from anywhere on the Internet. Since you want this web server to be accessible to the general public, you can leave this rule as is without any further configuration.
- Click **Review and Launch**.
- Review your choices, and then click **Launch**.
  **Note** You may see a warning on this screen that "Your security group … is open to the world." This is a result of not restricting SSH access to your machine, as described earlier. For the purposes of this lab only, you may ignore this warning.
- In the key pair dialog box, create a new key pair and name it <user-id>_lab8. Download key pair and then click **Launch Instances**.
  A status page notifies you that your instances are launching.
- Click **View Instances**.
- Before proceeding to the next step, check that the instances you started have finished their creation cycle. When they have finished their creation cycle, you'll notice that the instances transition to a *running* state with *2/2 checks passed*. This indicates that you instance is now fully available for us.

Name: _____ Username: _____

## Independently Connect to Each Web Server

When the servers are up and running, you will retrieve the public DNS entry allocated to each server in order to access them from your web browser.

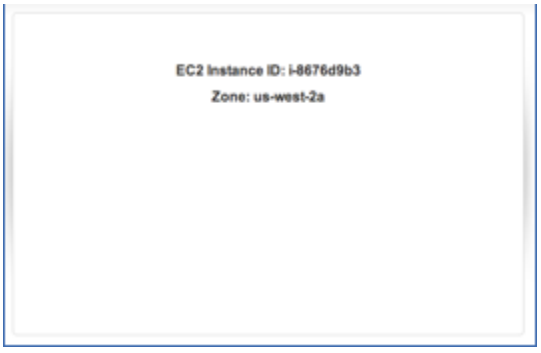
All Amazon EC2 instances are assigned two IP addresses at launch: a *private IP address* (RFC 1918) and a *public IP address* that are directly mapped to each other through Network Address Translation (NAT). Private IP addresses are only reachable from within the Amazon EC2 network. Public addresses are reachable from the Internet.

Amazon EC2 also provides an *internal DNS name* and a *public DNS name* that map to the private and public IP addresses, respectively. The internal DNS name can only be resolved within Amazon EC2. The public DNS name resolves to the public IP address outside the Amazon EC2 network and to the private IP address within the Amazon EC2 network.

### Retrieve your host's public DNS address

- Select your first Amazon EC2 instance to display a list of details and a status update for your instance in the bottom pane of the console.
- Copy the **Public DNS** value to your Clipboard. It will look something like *ec2-54-84-236-205.compute-1.amazonaws.com.*

Open a new browser window, paste the Public DNS value into the address bar, and press ENTER.
Your browser will display a screen like this:




This is the web page returned by the PHP script that was installed when the instance was started. It is a simple script that interrogates the metadata service on each machine and returns the instance ID and the name of the Availability Zone in which the instance is running.

- Repeat the previous two steps for your second instance.

Notice that each machine displays a different instance ID. This will help you identify which instance is processing your request when you put a load balancer in front of them.

Name: _____ Username: _____

**Note**: If you see an error instead of the instance ID and Availability Zone when you access the instances from the browser, try again after a couple of minutes. It's possible that the bootstrapping script is still running and has not yet completed installing and starting the web server and PHP application. If errors persist, verify that you entered the bootstrap script correctly when you launched your instances and that the security group has port 80 open.

## Create a Load Balancer

You now have two web servers. Now you need a load balancer in front of these servers to give your users a single location for accessing both and to balance user requests across them. For this lab, you will be creating a simple HTTP load balancer.

- Return to the AWS Management Console.
- In the Console's left navigation pane, click **Load Balancers**.
  You may need to scroll down to see the link.
- Click **Create Load Balancer**. The **Create Load Balancer** wizard opens.
- Click **Classic Load Balancer**, then click **Continue**.
- In the **Load Balancer name** box, type a new name like **<user-id>-lab8-ELB**.
- Leave all other default settings and click **Next: Assign Security Groups**.
- For **Assign a security group**, make sure that **Select an existing security group** is selected.
- Select the security group you created when you created your web server instances (i.e., **<user-id>_lab8_SG**), and make sure that no other security groups are selected.
- Click **Next: Configure Security Settings**.
- Click **Next: Configure Health Check.**
- Change the **Ping Path** value to **/** (delete the text **index.html**).
- Change the **Healthy Threshold** value to **3**.

The ELB will periodically test the ping path on each of your web service instances to determine health: a 200 HTTP response code indicates a healthy status, and any other response code indicates an unhealthy status. If an instance is unhealthy and continues in that state for a successive number of checks (*unhealthy threshold*), the load balancer will remove it from service until it recovers.

In this configuration, making the ping path just a slash (/) will return the default page—the PHP-generated page seen earlier.

The *healthy threshold* is the number of successful checks the load balancer expects to see in a row before bringing an instance into service behind the load balancer. The lower value will speed things up for this exercise.

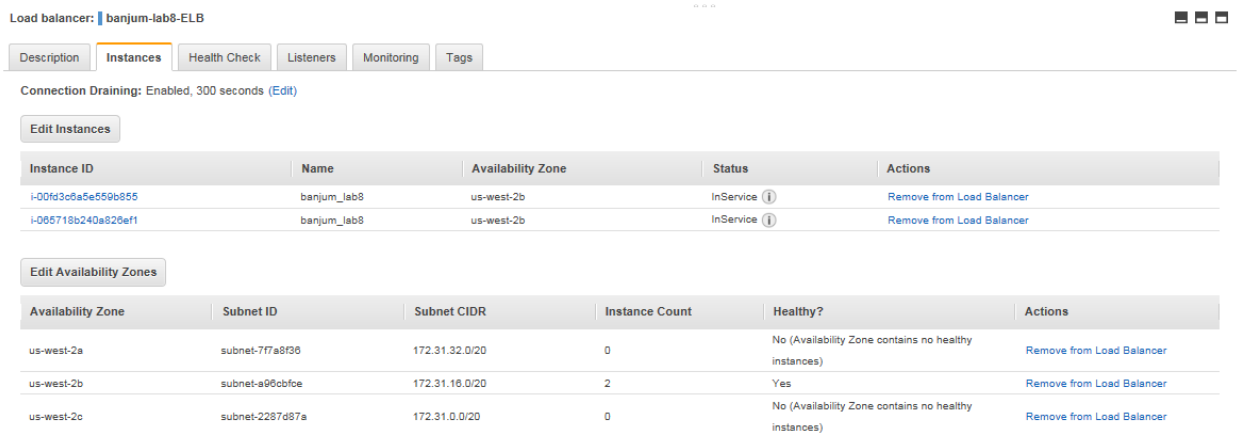
- Click **Next: Add EC2 Instances.**
- Select both of your web server instances to add them to your load balancer, and then click **Next: Add Tags**.
- Here is where you could add tags and data to your tags. For this lab, tags are not necessary. Leave these fields empty and click **Review and Create.**

Name: _____ Username: _____

- Review your settings, and then click **Create.**
- AWS is now creating your load balancer. It will take a couple of minutes to start up the load balancer, attach your web servers, and pass the health checks. Click on **<user-id>-lab8-ELB** to monitor its progress.
- Select the load balancer that you just created, click the **Instances** tab, and wait for the status of both Instances to change to *InService*. To refresh the status, click the circular arrow icon in the upper-right.

**Submission: When the status of both Instances is *InService*, take a screenshot of the dashboard, with Instances tab selected and both InstanceID and AvailabilityZones visible underneath it. Make sure the name of the Load Balancer is also visible.**

My screenshot looks like this:




- When the status of both Instances is *InService*, click the **Description** tab.
- Copy the **DNS Name** value to your Clipboard. It will look something like *<user-id>-lab8-ELB-756456754.us-west-2.elb.amazonaws.com*. Do not copy the "(A Record)" text.

**Note** Load balancers can span Availability Zones, and they also scale elastically as needed to handle demand. Therefore, you should always access a load balancer by DNS hostname, and not by IP address. A load balancer may have multiple IP addresses associated with its DNS hostname.

- Open a new browser window, paste the DNS Name value into the address bar, and press ENTER.
- Refresh your browser a few times, and you should see the Amazon EC2 Instance IDs changing. This means that the repeated responses are coming back through your two different web servers.

Name: _____ Username: _____


**Submission: What is the Public DNS Name of your load balancer? Your professor will be testing the above step while marking.**

## View Elastic Load Balancing CloudWatch Metrics

Elastic Load Balancing automatically reports load balancer metrics to CloudWatch.

Amazon CloudWatch provides monitoring for AWS cloud resources and the applications customers run on AWS. Developers and system administrators can use it to collect and track metrics, gain insight, and react immediately to keep their applications and businesses running smoothly. CloudWatch monitors AWS resources such as Amazon EC2 and Amazon Relational Database Service (RDS) DB instances and can also monitor custom metrics generated by your applications and services. With CloudWatch, you gain system-wide visibility into resource use, application performance, and operational health.

Amazon CloudWatch provides a reliable, scalable, and flexible monitoring solution that you can start using within minutes. You no longer need to set up, manage, or scale your own monitoring systems and infrastructure. Using CloudWatch, you can easily monitor as much or as little metric data as you need. CloudWatch lets you programmatically retrieve your monitoring data, view graphs, and set alarms to help you troubleshoot, spot trends, and take automated action based on the state of your cloud environment.

- Return to the AWS Management Console and, on the **Services** menu, click **CloudWatch**.
- In the navigation pane, click **Metrics** and in the **All Metrics** tab, click **ELB**.

  **Tip:** You could also use the search field to search for any metric you would like to view.

- Scroll up and down to select the metric or metrics you would like to view.

Load balancing metrics include latency, request count, and healthy and unhealthy host counts. Metrics are reported as they are encountered and can take several minutes to show up in CloudWatch.

Name: _____ Username: _____

- Go to Metrics -> All Metrics -> (All -> ELB -> Per-LB Metrics)
- For your load balancer, select Latency, HealthyHostCount, and UnHealthyRequestCount.
- Open a new browser window, paste the DNS Name value into the address bar, and press ENTER.
- Refresh your browser a few times to get some numbers for Latency.
- Wait 5 minutes and refresh the graph. You should see 3 metrics in the graph

Keep these metrics running, we will come back to them at the end of the lab.

## Amazon EC2 Auto Scaling Introduction

Auto Scaling helps you ensure that you have the correct number of EC2 instances available to handle the load for your application. You create collections of EC2 instances, called *Auto Scaling groups*.

- You can specify the minimum number of instances in each Auto Scaling group, and Auto Scaling ensures that your group never goes *below* this size.
- You can specify the maximum number of instances in each Auto Scaling group, and Auto Scaling ensures that your group never goes *above* this size.

If you specify the *desired* capacity, either when you create the group or at any time thereafter, Auto Scaling ensures that your group has this many instances.

If you specify scaling policies, then Auto Scaling can launch or terminate instances as demand on your application increases or decreases.

## Amazon EC2 Auto Scaling Components and Terminology

The following table describes the key components of Auto Scaling.

| Images | contents |
|---|---|
|  | **Groups**<br><br>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. For more information, see Auto Scaling Groups. |
|  | **Launch configurations**<br><br>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, |

Name: _____ Username: _____

Images contents

instance type, key pair, security groups, and block device mapping for your instances. For more information, see Launch Configurations.

**Scaling plans**

A *scaling plan* tells Auto Scaling when and how to scale. For example, you can base a scaling plan on the occurrence of specified conditions (dynamic scaling) or on a schedule. For more information, see Scaling Plans.

## Pricing for Auto Scaling

There are no additional fees with Auto Scaling, so it's easy to try it out and see how it can benefit your AWS architecture.

## Specify the AMI and instance type to be created on your behalf

A launch configuration specifies the type of EC2 instance that Auto Scaling creates for you. You create the launch configuration by including information such as the Amazon Machine Image (AMI) ID to use for launching the EC2 instance, the instance type, key pairs, security groups, and block device mappings, among other configuration settings.

- In the AWS Management Console, click **EC2**.
- In the left-hand navigation pane, under **Auto Scaling,** click **Launch Configurations**.
- Click **Create a new launch configuration**.
- The **Choose AMI** page displays a list of basic configurations, called Amazon Machine Images (AMIs), that serve as templates for your instance. Select the **64-bit Amazon Linux AMI** (usually the first in the list).
- On the **Choose Instance Type** page, you will select a hardware configuration for your instance. Leave the t2.micro instance type selected by default.
- Click **Next: Configure details**.

## Specify additional details about your instance

- On the **Configure Details** page, do the following:
    - In the **Name** field, enter a name of your launch configuration (for example, **<user-id>-lab8-config**).
    - Click to expand **Advanced Details**, select an IP address type to be assigned to the instance. If you want to be able to connect to an instance in a VPC, you must select an option that assigns a public IP address. If you want to connect to your

instance but aren't sure whether you have a default VPC, select Assign a public IP address to every instance. For now, **leave the default selection**.
    - Under **Advanced Details**, under **User data**, copy the script that we used to launch our EC2 instances earlier in the lab
    - Click **Next: Add Storage**. Keep defaults.
    - Click **Next: Configure Security Group**. Select an existing group <user-id>_lab8_SG
- On the **Review** page, read the warning about security group settings, but ignore for the purposes of this lab. Click **Create launch configuration**.
- In the **Select an existing key pair or create a new key pair** dialog box, select Choose an existing key pair and select <user-id>_lab8. Click the acknowledgement check box and click **Create launch configuration**.

## Create An Auto Scaling Group

**Specify the settings for when and how your instances are launched**

An Auto Scaling group can help you make sure that your application always has the right amount of capacity to handle the current traffic demands. You can also use Auto Scaling to make your applications more highly available and fault tolerant.

- On the **Configure Auto Scaling group details** page, do the following:
- In Group name, enter a name for your Auto Scaling group (for example, "**<user-id>_lab8_ASG**").
  b) Set Group size: Start with 2 instance for this lab.
  c) Click in the **Subnet** field to reveal subnet selections, and choose any.
  d) Click **Next: Configure scaling policies**.
- In the **Configure scaling policies** page, select **Keep this group at its initial size**
- Click Next: Configure Notifications. Keep defaults.
- Click Next: Configure Tags. Add tag **Name: <user-id>_lab8_asg_instance** and click **Review**.
- On the **Review** page, click **Create Auto Scaling group**.
- Click **Close**.

### Verify your Auto Scaling group

Now that you have created your Auto Scaling group, you are ready to verify that the group has launched your EC2 instance.

- On the Auto Scaling Groups page, select the Auto Scaling group that you just created. The **Details** tab provides information about the Auto Scaling group.
- Select the **Activity History** tab. The Status column contains the current status of your instance. When your instance is launching, the status column shows In progress. The status changes to Successful after the instance is launched. You can also click the refresh button to see the current status of your instance.

Name: _____ Username: _____

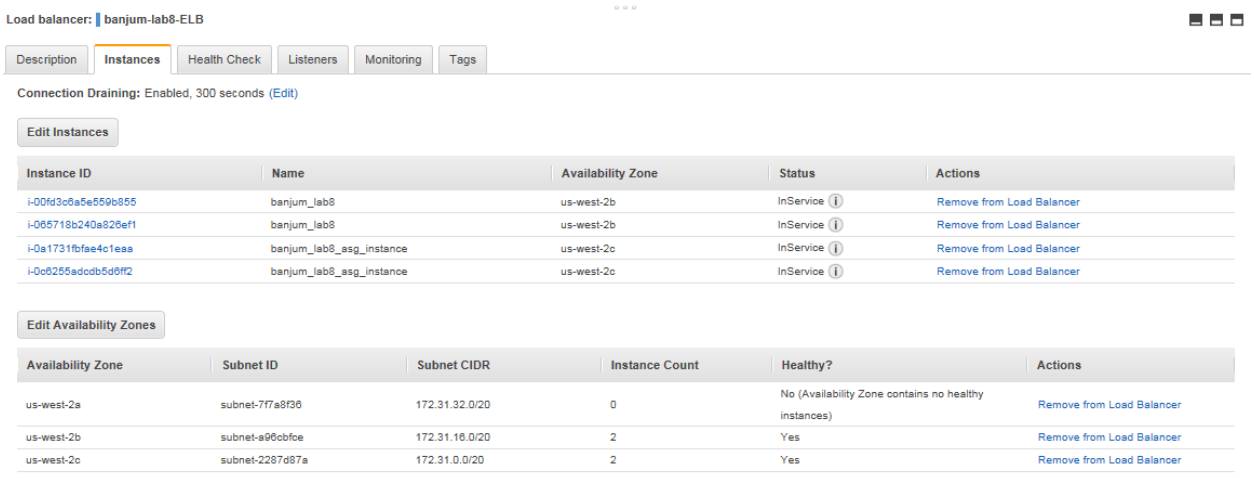
- Select the **Instances** tab. The Lifecycle column contains the state of your newly launched instance. You can see that your Auto Scaling group has launched your EC2 instance, and it is in the InService lifecycle state. The Health Status column shows the result of the EC2 instance health check on your instance.

## Add the Auto Scaling Group behind your Load Balancer

- On the navigation pane, under Auto Scaling, choose **Auto Scaling Groups**.
- Select your group.
- On the **Details** tab, choose Edit. Click next to the field "Load Balancers" and select your load balancer created earlier in this lab, <user-id>-lab8-ELB.
- Click Save.
- Go back to Load Balancers Dashboard and select your Load Balancer.

**Submission: When the status of all 4 Instances is *InService*, take a screenshot of the dashboard, with Instances tab selected and both InstanceID and AvailabilityZones visible underneath it. Make sure the name of the Load Balancer is also visible.**

My screenshot looks like this:




## Run a Test of Auto Scaling

Try the following experiment to learn more about Auto Scaling. The minimum size for your Auto Scaling group is 1 instance. Therefore, if you terminate the running instance, Auto Scaling must launch a new instance to replace it.

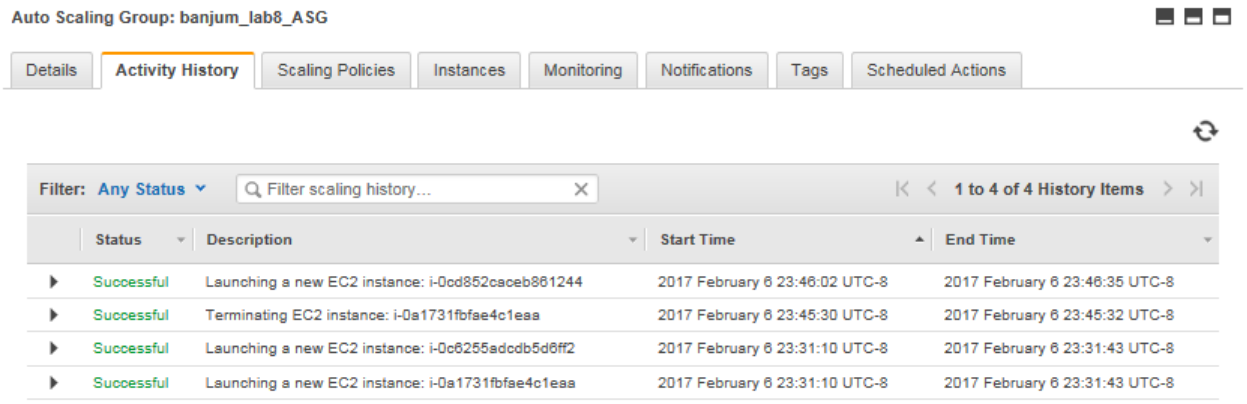
- On the **Instances** tab, click the ID of the instance. This takes you to the Instances page and selects the instance.
- Click **Actions**, select Instance State, and then click **Terminate**. When prompted for confirmation, click **Yes, Terminate**.
- In the left-hand navigation pane, select **Auto Scaling Groups** and then select the **Instances** tab. You will see the initial instance status as *Terminating*. Soon thereafter you

will see a new instance appear with a status of *Pending* or *InService*. The default cooldown for the Auto Scaling group is 300 seconds (5 minutes), so it may take about 5 minutes until you see the scaling activity.

- Return to the **Activity History** tab. All scaling activities are logged here. After the scaling activity starts, you'll see entries for the launch and termination of the first instance and then an entry for the launch of the new instance.

**Submission: Take snapshot of complete Activity History. Make sure the name of the Auto Scaling Group visible in the screenshot.**

My screenshot looks like this:




## Gather CloudWatch Statistics

- Return to the AWS Management Console and, on the **Services** menu, click **CloudWatch**.
- Graph the three metrics as done before (Latency, HealthyHostCount, UnHealthyHostCount)
- The HealthyHostCount graph should have started with 2 (initial configuration), jumped to 4 (when we added the hosts of the auto scaling group to the load balancer), decremented to 3 (when we terminated one host in the ASG) and finally be reporting 4 instances (after ASG automatically launched a new instance)
  Tip: You may have to wait 5 minutes for the graph to update itself with latest statistics.

**Submission: Take a screenshot of the Metrics Graph**