AWS 2

CCNP Lab 7

Jason Liu

CCNP – Mr. Mason & Mr. Hansen

Period 6, 7, 8

*Lab 7: AWS Continued*

Once again, Amazon Web Service (AWS) is a “Cloud” platform. The Cloud provides on-demand IT resources via the Internet, with the goal of providing easy and scalable access to those computing services. AWS provides many Amazon proprietary services through their AWS cloud, including computing and storage software.

This lab will continue to explain the processes and procedures of Amazon AWS, namely, the next three labs of AWS. This will include Amazon Elastic Block Store (EBS), Database tools, Auto Scaling, and Load Balancing. These services are major components of AWS fundamentals for the practitioner course.

Lab 4: AWS EBS

**Explanation**

The objective of the lab is to learn the process of creating an EBS storage volume and attaching it to an EC2 instance. We will manipulate the several parameters used in the creation process as presented in the Amazon Management Console. Also, we will create a new volume from a snapshot.

Amazon Elastic Block Store is an adjacent service of Amazon EC2 used primarily for persistent storage. This is due to the redundancy and reliability of the EBS volumes. Volumes are independent of the EC2 instance, replicated on the backend of an Availability Zone, and are unformatted blocks, compatible with any operating system. EBS is most recommended when the data requires long term persistence and encryption. This is because volumes persist even after an instance is stopped.

The instance will be accessed and interacted with ***remotely*** through SSH. A PuTTY private key (PPK) is provided, and many Linux commands will facilitate the usage of the launched volume block.

**Key Terms**

**Amazon Elastic Block Store (EBS):** Raw, unformatted block storage service that can be attached to EC2 instances.

**Volumes:** The block-level storage device unit that is to be attached to the instances.

**Snapshots:** A point-in-time copy of the data. Used to move data across regions and accounts or to provide general redundancy.

**Amazon Elastic Compute Cloud (EC2):** AWS service that runs virtual computers for the user’s computer applications.

**Topological Overview**

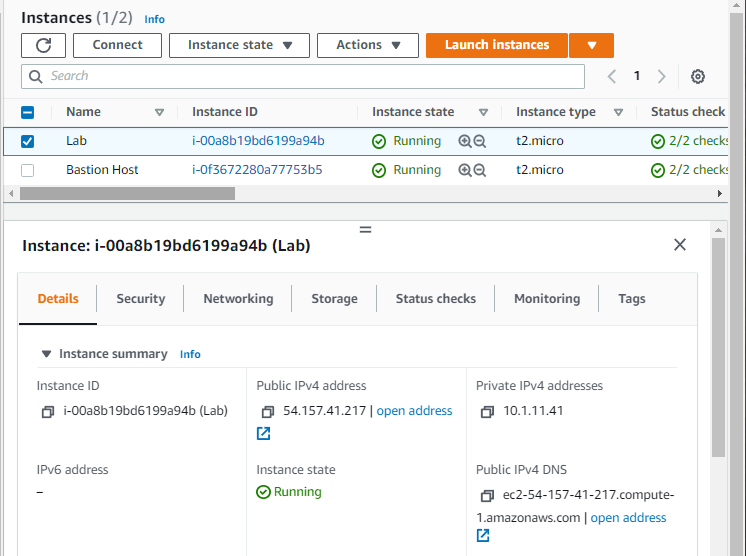
**General Procedure**

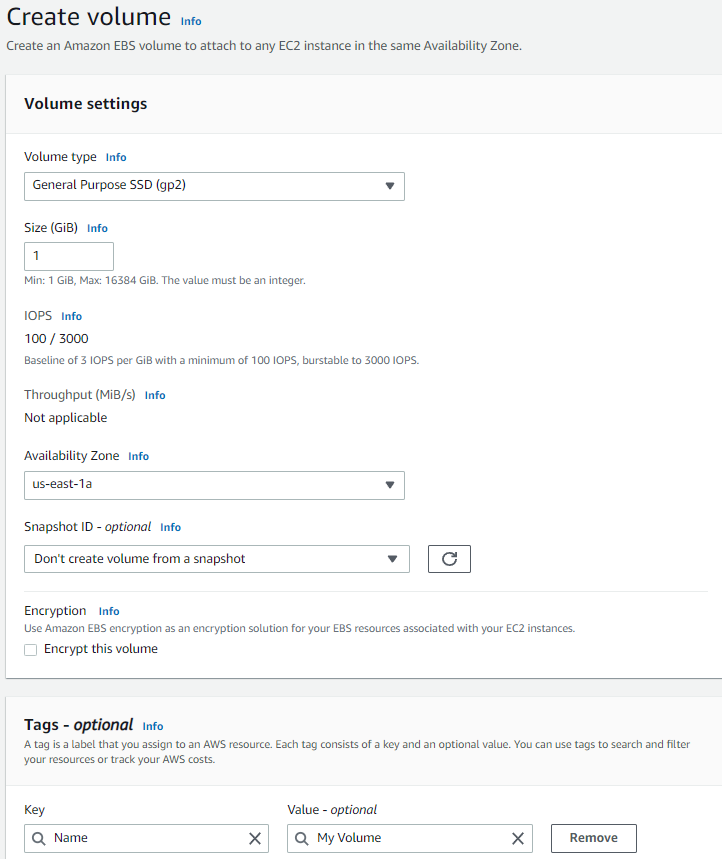
1. Create an EBS volume via Amazon Management Console interface.
2. Attach newly made volume to an EC2 instance.
3. Connect to the EC2 instance via SSH.
4. Configure and mount the volume.
5. Create a snapshot of the volume.
6. Retrieve data stored in the snapshot.

**Accessing Lab Version of AWS Management Console**

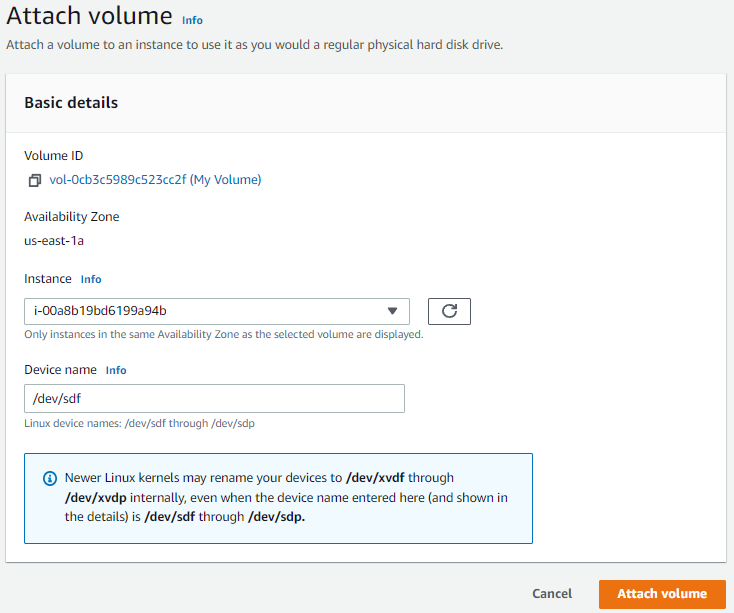
As lab contents are AWS course propriety, you must access their interactive lab via the AWS Academy Cloud Foundations course, Module 7: Lab 4 – Working with EBS.

**Task 1: Creating an EBS Volume and Attaching It**

You will create and attach an Amazon EBS volume to a new Amazon EC2 instance.

1. In the AWS Management, access EC2.
2. On the navigation panel on the left, see Instances.
   1. A pre-built lab was constructed for the purpose of the lab, named *Lab* (see right.)
   2. The availability zone is *us-east-1a*, the default lab availability zone.
3. On the navigation panel on the left, open Volumes. We will create a 1 GiB volume. Choose Create Volume and configure:
   1. Volume Type: *General Purpose SSD (gp2)*
   2. Size (GiB): 1
   3. Same availability zone as the instance.
   4. Add tag of:
      1. Key: *Name*
      2. Value: *My Volume*
   5. Finalize and create. The new volume has been established and will move from the Creating to Available state.

Volume Creation Configurations

1. We will now attach the newly created volume to the pre-built *Lab* instance. Select volume named “My Volume”.
   1. In the Actions menu, choose Attach Volume.
   2. In the Instance field, select the corresponding *Lab* instance.
   3. Finalize and attach. The volume should now be *In-use*.

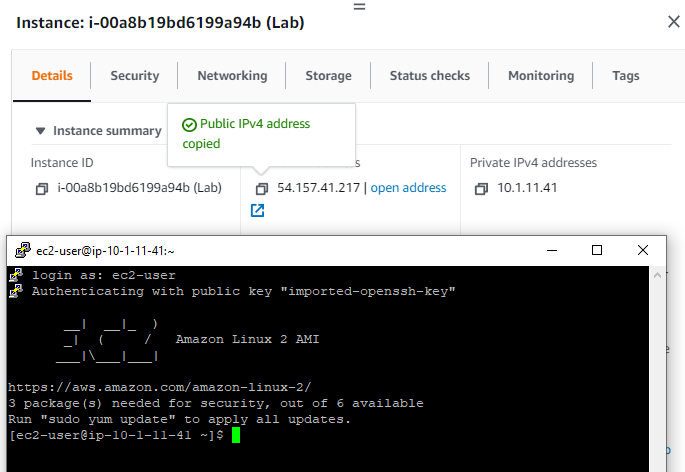
Attaching Volume to *Lab* Instance

**Task 2: Connect to the EC2 Instance**

Use PuTTY to SSH into the instance. This only applies to Windows Users.

1. From the interactive lab menu in the AWS academy, download and save the “labsuser.ppk” file by choosing the “Download PPK” button from the details tab.
2. Open PuTTY. Configure the session:
   1. Under Connections, set “Seconds between keepalives” to 30 to allow the session to be open for a longer period of time.
   2. Under Session, paste the IPv4 public IP address of the *Lab* instance. This can be found in the description tab of the instance (EC2 page)
   3. Back in PuTTY, expand SSH and select “Auth,” to browse and select the “labsuser.ppk” file.
   4. Open the session. When prompted to login, enter *ec2-user*.
   5. You have successfully connected to the EC2 instance.

**PPK files**: PuTTY Private Key files hold private keys that are used to enable communication with another user having the corresponding public key.

We used the key to access the lab instance via SSH.

Signing into PuTTY session with SSH, using PPK file and IPv4 address of instance

1. View storage available on the instance. Enter command “df -h”. You will see:

Filesystem Size Used Avail Use% Mounted on

devtmpfs 484M 0 484M 0% /dev

tmpfs 492M 0 492M 0% /dev/shm

tmpfs 492M 460K 491M 1% /run

tmpfs 492M 0 492M 0% /sys/fs/cgroup

/dev/xvda1 8.0G 1.5G 6.6G 19% /

tmpfs 99M 0 99M 0% /run/user/0

tmpfs 99M 0 99M 0% /run/user/1000

Note that the new volume of 1 GiB is not shown.

1. We will now add a new volume as an ext3 file system under a mount point.
   1. Create an ext3 file system using the command “sudo mkfs -t ext3 /dev/sdf”
   2. Create a directory for mounting the volume using the command “sudo mkdir /mnt/data-store.”
   3. Mount the volume. Use “sudo mount /dev/sdf /mnt/data-store.”
2. View the storage again with “df -h.” You will see an additional line of “-/dev/xvdf”

Filesystem Size Used Avail Use% Mounted on

devtmpfs 484M 0 484M 0% /dev

tmpfs 492M 0 492M 0% /dev/shm

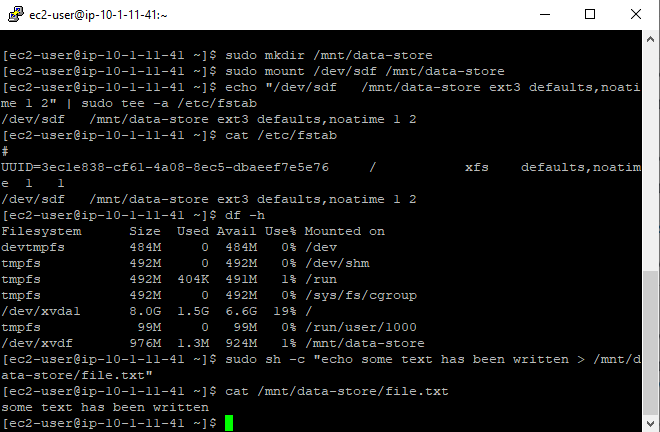
tmpfs 492M 460K 491M 1% /run

tmpfs 492M 0 492M 0% /sys/fs/cgroup

/dev/xvda1 8.0G 1.5G 6.6G 19% /

tmpfs 99M 0 99M 0% /run/user/0

tmpfs 99M 0 99M 0% /run/user/1000

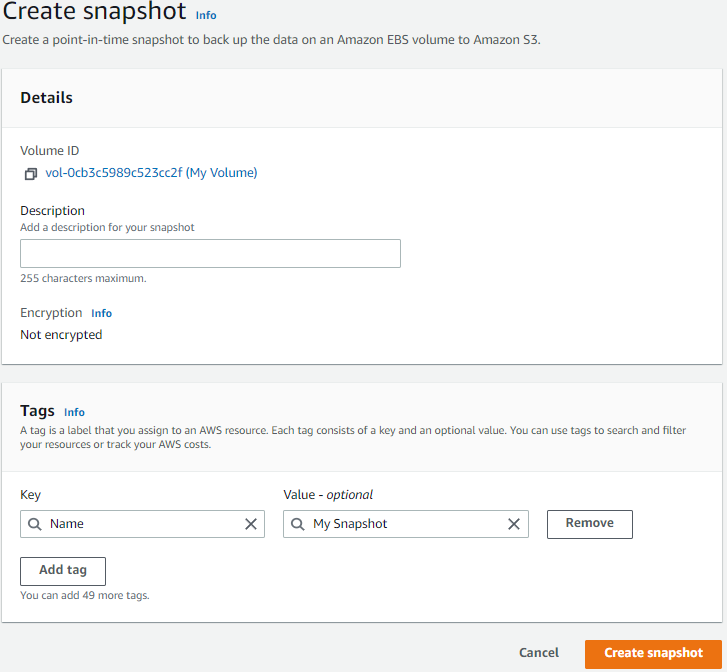
/dev/xvdf 976M 1.3M 924M 1% /mnt/data-store

Using PuTTY session to add new volume to instance under a mount point. Also, text file has been verified.

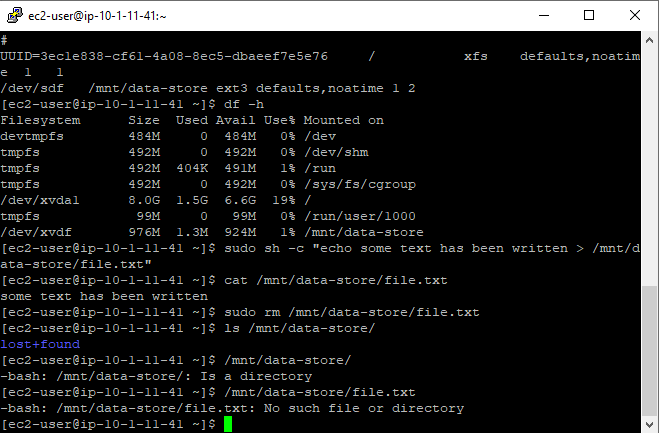
1. On the mounted volume, create a file and add some text to it to verify functionality.
   1. Use command “sudo sh -c "echo some text has been written > /mnt/data-store/file.txt"”
   2. Verify using the command “cat /mnt/data-store/file.txt.”

**Task 3: Create and Use an EBS snapshot**

You will create a snapshot of the current EBS volume. New Amazon EBS volumes can then be created out of snapshots for cloning or restoring backups. In this scenario, we will use it to retrieve data of the captured moment.

1. In the AWS Management Console, choose Volumes and select “My Volume.”
   1. Under the Actions menu, select Create Snapshot.
   2. Configure a tag consisting of:
      1. Key: *Name*
      2. Value: *My Snapshot*
   3. Finalize and create snapshot.

Creating Snapshot and adding a Tag

1. On the navigation panel on the left, choose Snapshots. The snapshot will be displayed. Status may display *pending* before *completed*.
2. In the remote SSH session, delete the file that you created on your volume. We will now attach the snapshot-based volume to the same instance to verify that the file was saved there.
   1. Delete file with the command: “sudo rm /mnt/data-store/file.txt”
   2. Verify deletion with command: “ls /mnt/data-store/”

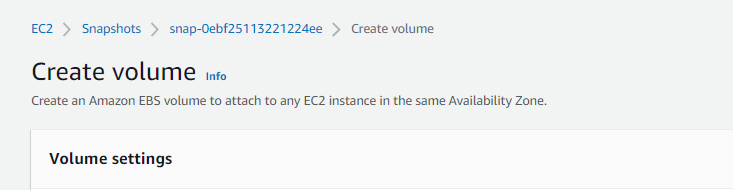
Deletion of file that was created on the original volume.

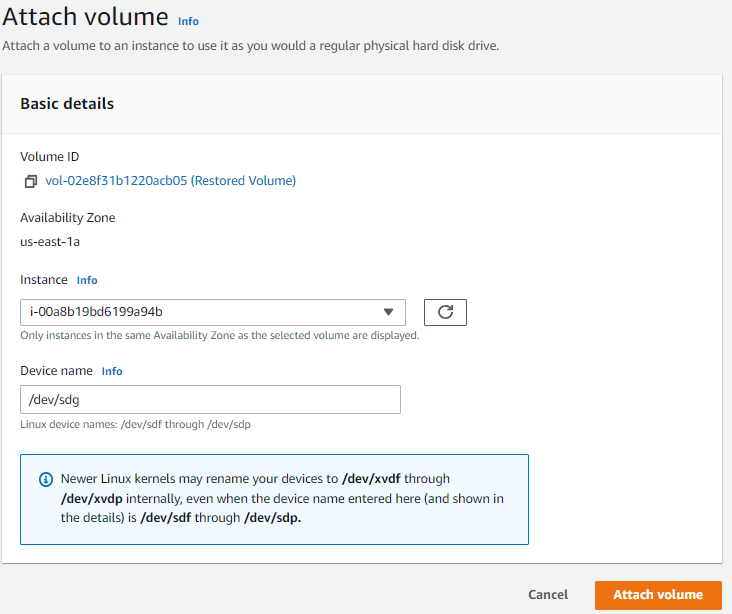
1. Create volume using your snapshot. In the AWS Management Console, select “My Snapshot.”
   1. Under the Actions menu, select “Create volume from snapshot.”
   2. Use the same availability zone as used earlier.

You do not need to change the volume type, size, or availability zone, but you always can.

* 1. Configure a tag consisting of:
     1. Key: *Name*
     2. Value: *Restored Volume*

Creating Volume from Snapshot

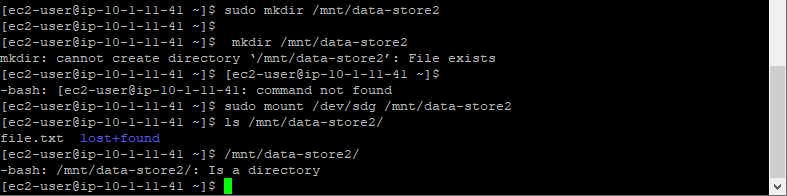
* 1. Finalize and create volume.

1. On the navigation panel on the left, select Volumes and select “Restored Volume.”
   1. Under the Actions menu, choose Attach Volume.
   2. In the Instance field, select the corresponding *Lab* instance.
   3. Finalize and Attach Volume. The volume should now be *In-use*.

Attaching Restored Volume to EC2 Instance

1. Mount the restored volume.
   1. Create a directory for mounting the new storage volume with the command

“sudo mkdir /mnt/data-store2”

* 1. Mount the volume with the command “sudo mount /dev/sdg /mnt/data-store2”
  2. ****Verify that volume you mounted has the file that you created earlier. Upon entering the command “ls /mnt/data-store2/” you should be able to see the file.txt.

Mounting Restored Volume and seeing file text created earlier.

**Conclusion**

AWS EBS is an essential storage service for long-term redundancy alongside EC2 instances. In this lab, we learned how to interact with an EC2 instance remotely through SSH, getting to know some of the storage-related commands, such as mounting and display commands. We also learned how to view and use snapshots to retrieve previous sets of data. EBS volumes are a flexible and reliable storage tool that will be fundamental to AWS usage.

Lab 5: AWS RDS

**Explanation**

Lab five presents Amazon’s Database Service (Amazon RDS), used to create relational database servers that run off of commonly used database engines, such as Amazon Aurora, Oracle, and MySQL. These structured query languages (SQLs) help run the software for the database. We will create a database (DB) instance and assign it to certain subnets created in the Availability Zones, telling RDS which subnets can be used for the database. Then, we will interact with the database through the Web Server that it is assigned to. Note that both the Security Group that permits the web server to access the DB instance and the DB Subnet Group that assigns the subnets, are created before the DB instance itself, as those components are selected in the instance’s creation process.

This lab will begin with a topology last established from the VPC (see AWS lab 2). A VPC, named *Lab VPC* has already been created in the academy lab generation for this module. It consists of 2 availability zones, 4 subnets, and a security group for an already established Web Server. We will assign the security and subnet groups for the DB instance within the ***private*** subnets (see Topological Overview).

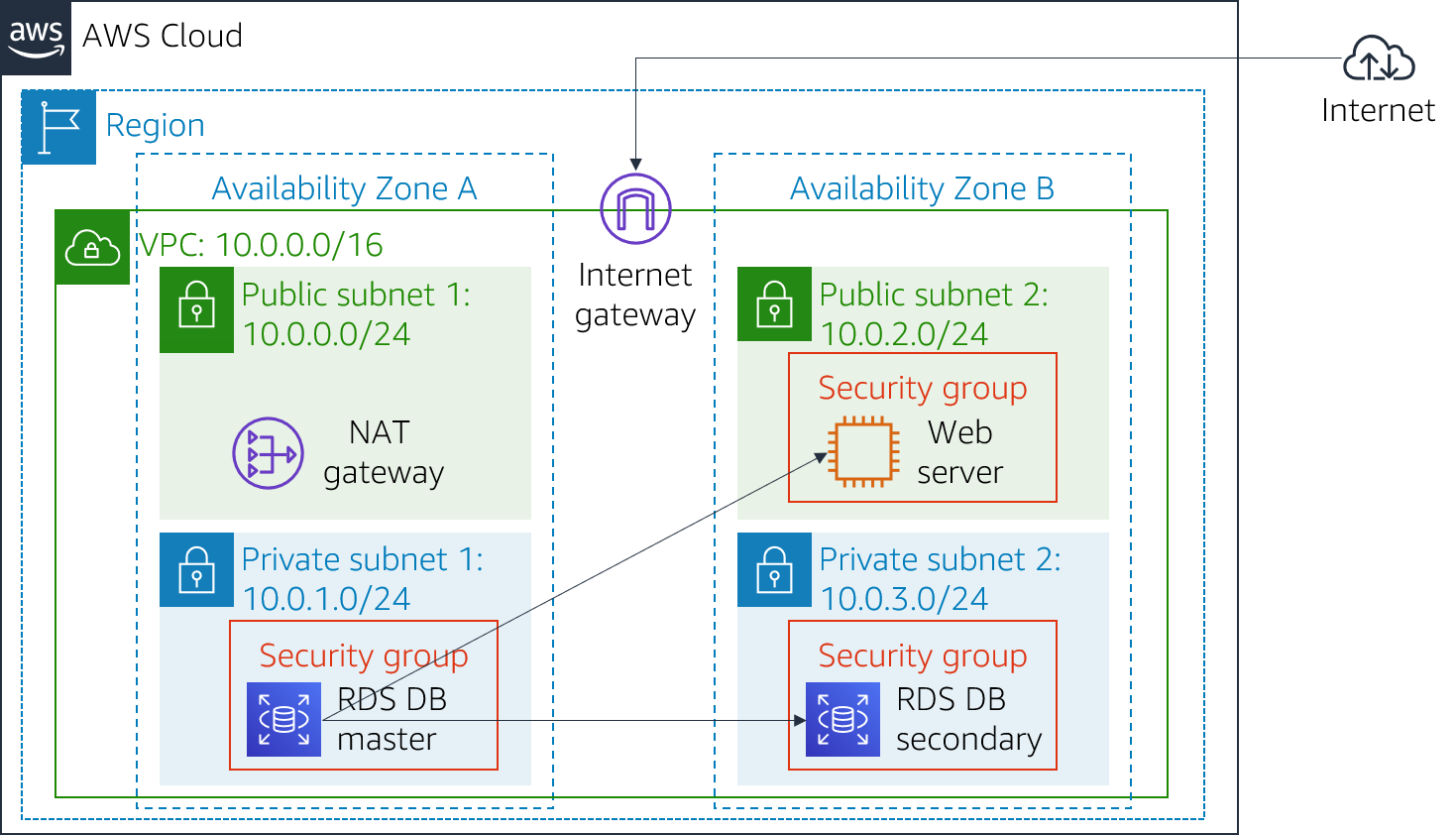
**Key Terms**

**Amazon Relational Database Service (RDS):** AWS service that creates, operates, and scales a relational database in the cloud. It manages database administration tasks.

**Relational Database:** A type of database that stores data with items of pre-defined relationships between the data points. It is formatted as a set of tables with columns and rows, each row marked with an identifier called a primary key. Each column holds a certain kind of data. The data can be accessed in different ways ***without*** reformatting the tables.

**Database Engine:** The underlying software that a database management system uses to create, read, update and delete data from a database. Amazon RDS uses 6 popular engines, including MySQL, Oracle, Amazon Aurora, Microsoft SQL Server, PostgreSQL, and MariaDB.

**Security Groups:** Virtual firewalls, created in the AWS VPC service, to filter incoming and outgoing traffic. It is used to control traffic to and from an instance. We will use a security group to allow the web server to access a DB instance.

**Topological Overview**

**General Procedure**

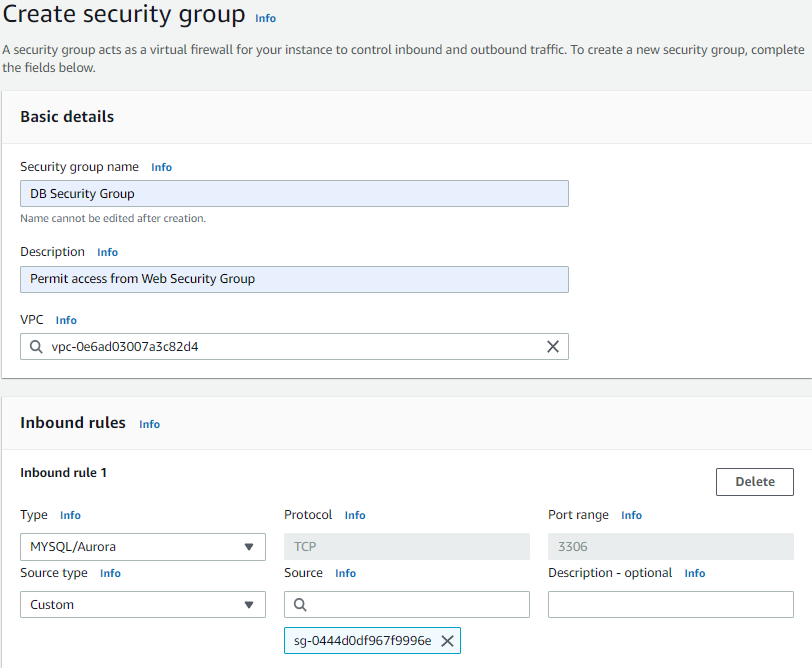
1. Create a security group for the RDS DB instance.
2. Create a DB subnet group for the DB instance.
3. Create the DB instance, using the security groups and subnet groups previously configured.
4. Launch and interact with the database via the assigned web server.

**Accessing Lab Version of AWS Management Console**

As lab contents are AWS course propriety, you must access their interactive lab via the AWS Academy Cloud Foundations course, Module 8: Lab 5 – Build a Database Server.

**Task 1: Create a Security Group for the DB Instance**

The security group will allow the web server to access the RDS DB instance.

1. In the AWS Management Console, choose VPC.
2. On the navigation panel on the left, select Security Groups. Choose “Create security group” and configure:
   1. Security group name: *DB Security Group*
   2. Description: *Permit access from Web Security Group*
   3. VPC: *Lab VPC*
3. You need to add a rule to the security group to permit inbound database requests. On the Inbound rules panel, select “Add rule” and configure:
   1. Type: *MySQL/Aurora (3306)*
   2. CIDR, IP, Security Group or Prefix List: *Web Security Group* (Typing “sg” helps search)
   3. Finalize and create security group.

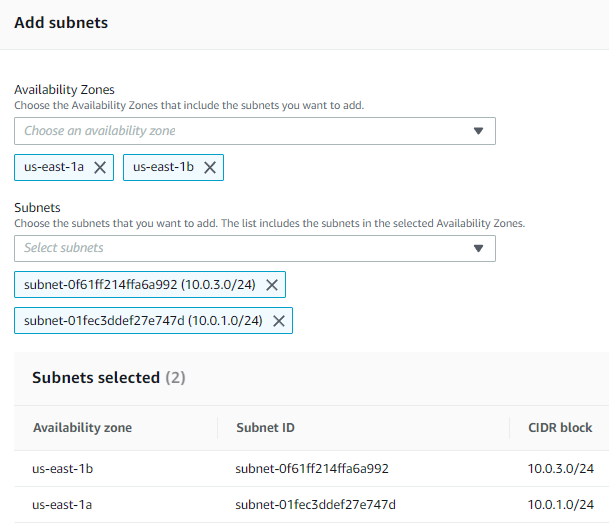
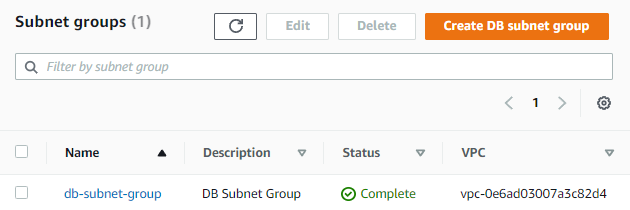
Security Group Configuration.

**Task 2: Create a DB Subnet Group**

**Subnet Group**: Informs RDS which subnets are used for the database. Note that each subnet group requires subnets in at least ***two*** availability zones.

1. On the services menu, choose RDS.

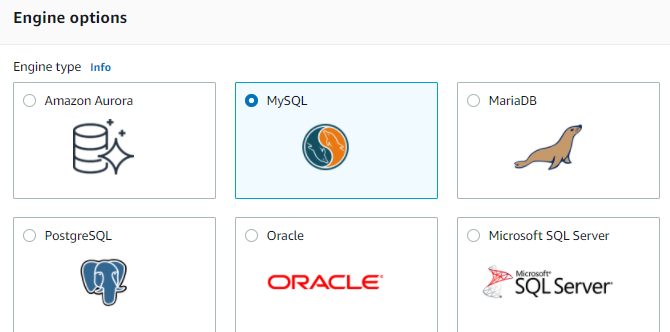
Adding subnets from 2 different AZs

1. On the navigation panel on the left, select Subnet groups. Choose “Create DB Subnet Group and configure:
   1. Name: *DB-Subnet-Group*
   2. Description: *DB Subnet Group*
   3. VPC: *Lab VPC*
2. Scroll down to the Add Subnets section. Under the list of Availability Zones, select the zones: “us-east-1a” and “us-east-1b”. The 2 required availability zones are now added.
3. Finalize and create the DB subnet group.

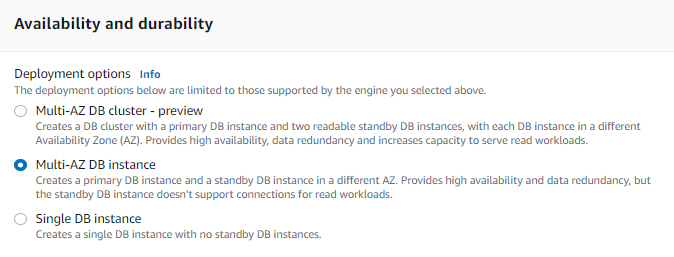
Established Subnet Group.

**Task 3: Create the DB Instance**

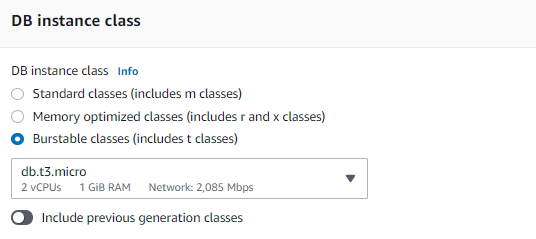
You will now launch the RDS DB instance. Specifically, a Multi-AZ instance running off of the MySQL database engine.

1. On the navigation panel on the left, select Databases. Choose “Create Database” amd select “MySQL”.

Selecting MySQL Engine for the database.

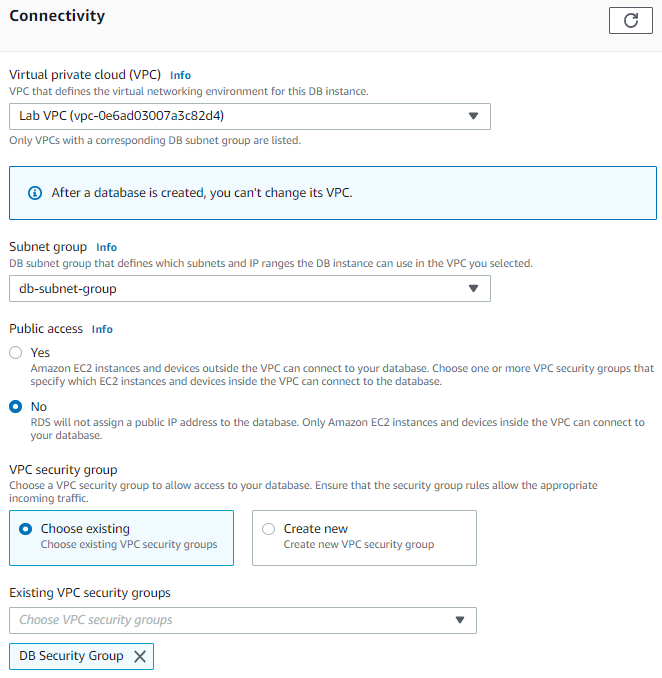
****Multi-AZ**: Deployments of enhanced availability and durability for DB instances. RDS automatically creates a primary instance and synchronously translates the data to a standby instance of the other AZ.

Selecting Multi-AZ option.

1. Under its Settings, configure username and password:
   1. DB instance identifier: *lab-db*
   2. Master username: *main*
   3. Master password: *lab-password*
   4. Confirm password: *lab-password*
2. Under DB instance class, configure:
   1. Select Burstable classes (includes t classes)
   2. Select *db.t3.micro*.

Selecting DB instance class.

**RDS instance types**: Types of instances consisting of several sizes and varying combinations of CPU, memory, storage, and networking capacity to serve relational database uses.

1. Under Storage, configure:
   1. Storage type: *General Purpose (SSD)*
   2. Allocated storage: *20*
2. Under Connectivity, configure:
   1. Virtual Private Cloud (VPC): *Lab VPC*
3. **Under Existing VPC security groups, choose:
   1. *DB Security Group*
   2. Deselect *default*.
4. Under Additional Configuration, configure:
   1. Initial Database name: *lab*
   2. Uncheck “Enable encryption” and “Enable Enhanced monitoring”.

Note that turning off backups is not recommended but will make the database deploy faster.

1. Finalize and create the database. The database should now be launched.
2. Clink on the lab-db link. Wait until Info changes to Modifying or Available. This will take a while, as the it is deploying the database in two different availability zones.

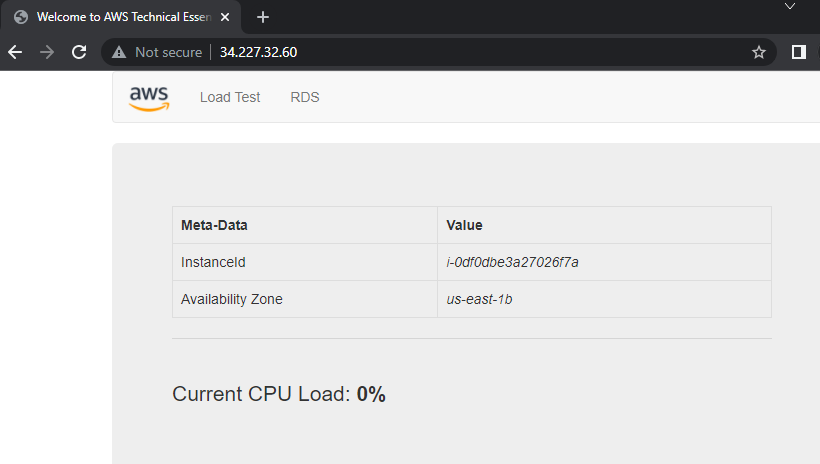
Connecting to VPC, DB subnet group, and previously made security group.

1. Then, scroll to the “Connectivity & Security” section and copy the “Endpoint” field. Example: *lab-db.cggq8lhnxvnv.us-west-2.rds.amazonaws.com*
2. Keep this value to access the database later. You may save it on a text editor.

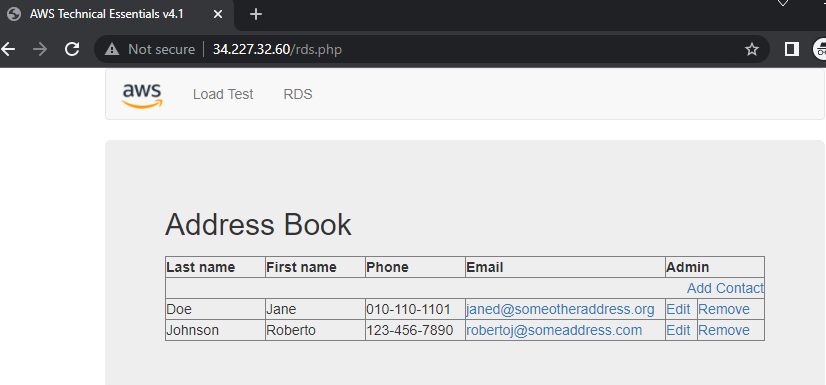
**Task 4: Interact with the Database**

You will access your web server and configure it to use the database.

1. The Web Server IP address is provided in the interactive lab module. Under the Details drop down menu, choose show. Paste the Web Server IP address into a standard incognito internet browser.

Note the web application will be showing information about the EC2 instance.

Web browser view of entering Web Server IP address.

1. You will configure the application to connect to your database. Select the RDS tab at the top of the page. Configure the following:
   1. Endpoint: Paste the Endpoint of the database.
   2. Database: *lab*
   3. Username: *main*
   4. Password: *lab-password*
   5. Finalize and submit.
2. The application is an ***address book***, and uses the information accessed from the RDS to database to display the data. It also then stores its information to the database.

Address book application uses RDS database.

1. Test the web application by adding, editing, and removing contact. Note that it is not only updating with the database, but also replicating data to the second availability zone.

**Conclusion**

Amazon’s Relational Database Service gives first insight on the technologies of databases. Learning the basic components of how a relational database is comprised gives a solid understanding of the configurations that were introduced in AWS. Security groups and subnet groups are important pre-requisite units that define the boundaries and limits of the database instance. Lastly, through the aid of an interactive web application, we were able to see the successful connectivity of the database firsthand. Prior knowledge of VPC and subnets helped establish the topological structure that was developed in this lab. There are still many concepts of databases yet to explore and understand in advanced AWS usage.

Lab 6: AWS Load Balancing and Auto Scaling

**Explanation**

Lab six is an introduction to AWS’s load balancing and auto scaling. These services automate data handling and administration during periods of high demand. They provide instantaneous flexibility and provide higher overall efficiency during those times.

Elastic Load Balancing (ELB) distributes the load of incoming data across several EC2 computing instances. It automates and scales the amount of load balancing capacity needed to route the traffic. When a connection goes down or the load balancing sees the target as unhealthy, it will stop the traffic and reroute it to a known healthy target. It will resume flow when the connection is healthy again. ELB also needs ***listeners***, a process that checks for connection requests, to accept incoming traffic. Finally, there are 3 types of load balancing: Application, Network, and Classic. Each operate at different layers (e.g., HTTP/HTTPS and TCP/SSL) and have different features.

AWS auto scaling builds scaling configurations that automate how different resources respond to changes in demand, adjusting the capacity of an operation to maintain steady, healthy performance. It can respond by adding/terminating instances and enforcing a minimum/maximum number of instances when beyond a stated desired capacity. In this lab, it is configured to add instances up to a set number when demand is high.

We will use Amazon CloudWatch to monitor an instance’s traffic load. It provides feedback through alarm statuses when certain data guidelines are pushed to far. This indicator helps us determine if the auto scaling’s response is functioning or not. In general, CloudWatch tracks and collets metrics from RDS and EC2, allowing governance and auditing uses.

**Key Terms**

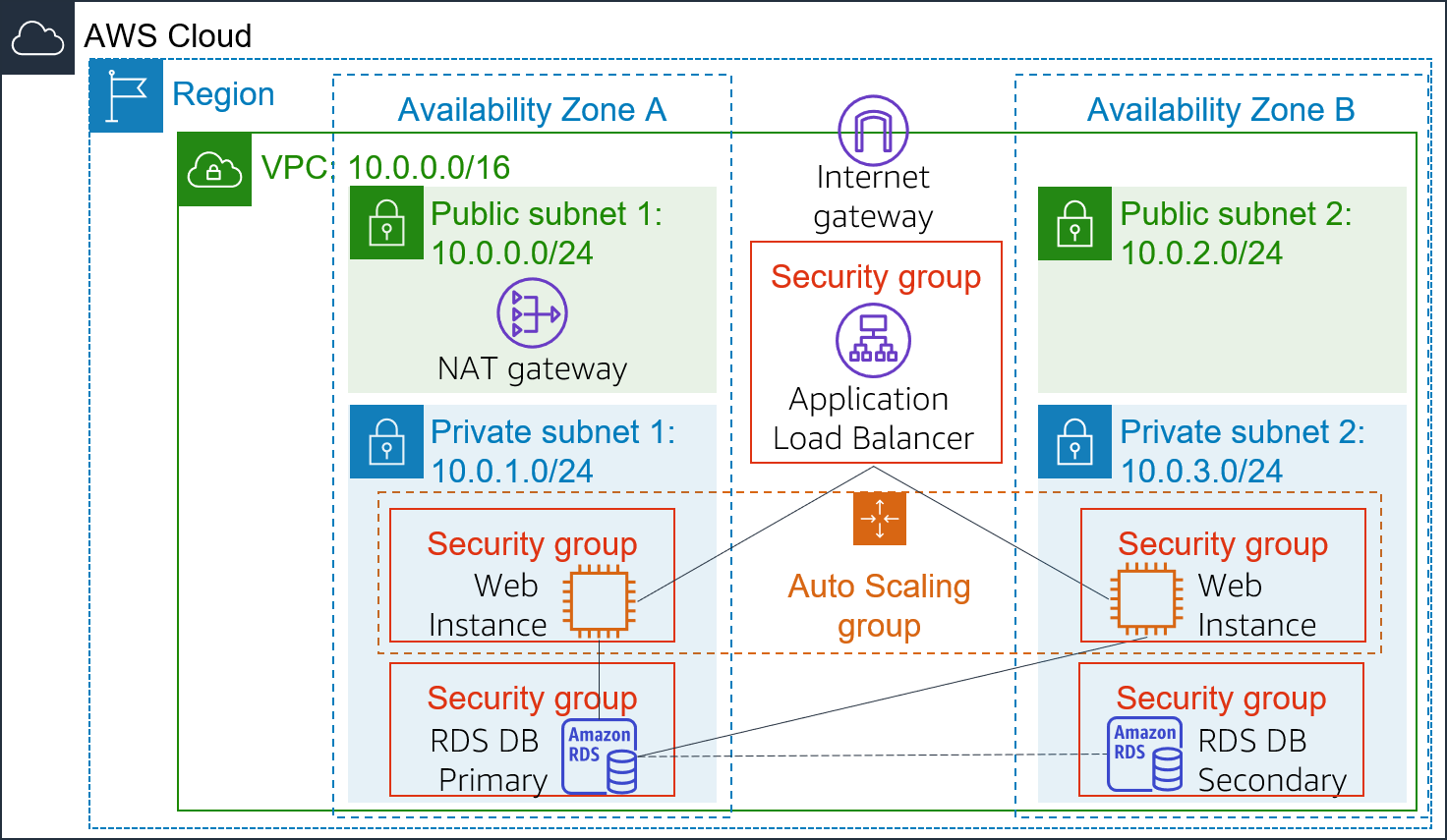
**Elastic Load Balancing (ELB):** Distributes incoming traffic across multiple EC2 instances automatically. Aids in fault tolerance.

**Auto Scaling:** Checks for and maintains availability by scaling the EC2 capacity automatically. You can manipulate the conditions of its automation. For example, it can increase the number of EC2 instances when faced with periods of high demands. It is good for applications with varying levels of usage.

**Amazon CloudWatch:** A monitoring service that provides data for AWS-related resources. We will use this service to set alarms to provide feedback on the instance and to purposely activate Load Balancing.

**Amazon Machine Images (AMI):** Mandatory information required to launch an instance. It is used for the Launch Configuration of the auto scaling group.

**Launch Configuration:** A template that the Auto Scaling group refers to when launching new EC2 instances. You will have to specify the AMI, its instance type, a key pair, and security group for this template.

**Topological Overview**

**General Procedure**

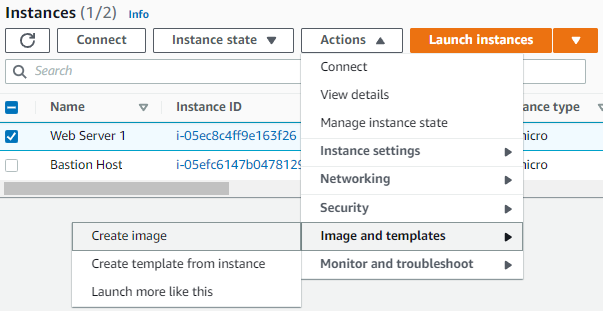
1. Create an AMI for Auto Scaling. This is taken from the Web Server instance.
2. Create a target group and load balancer.
3. Create a Launch Configuration and the Auto Scaling group.
4. Verify Load Balancing functionality.
5. Test Auto Scaling.

**Accessing Lab Version of AWS Management Console**

As lab contents are AWS course propriety, you must access their interactive lab via the AWS Academy Cloud Foundations course, Module 10: Lab 6 – Scale & Load Balance your Architecture.

**Task 1: Create an AMI for Auto Scaling**

You will create an AMI that will be used later for launching the Auto Scaling group. You will take an image from an existing EC2 instance named Web Server.

1. In the AWS Management Console, access EC2.
2. First, let’s confirm that the instance is running. On the navigation panel on the left, select Instances.
   1. Wait until the instance named *Web Server 1* has a status check of *2/2* checks passed.
3. We will create an AMI from this instance. Select the instance.
   1. In the Actions dropdown menu, select “Image and templates > Create image.”

Creating AMI (image) from existing instance.

* 1. In the image settings, configure:
     1. Image name: *WebServerAMI*
     2. Image description: *Lab AMI for the Web Server*
  2. Finalize and create the image. A confirmation banner will display the AMI ID.

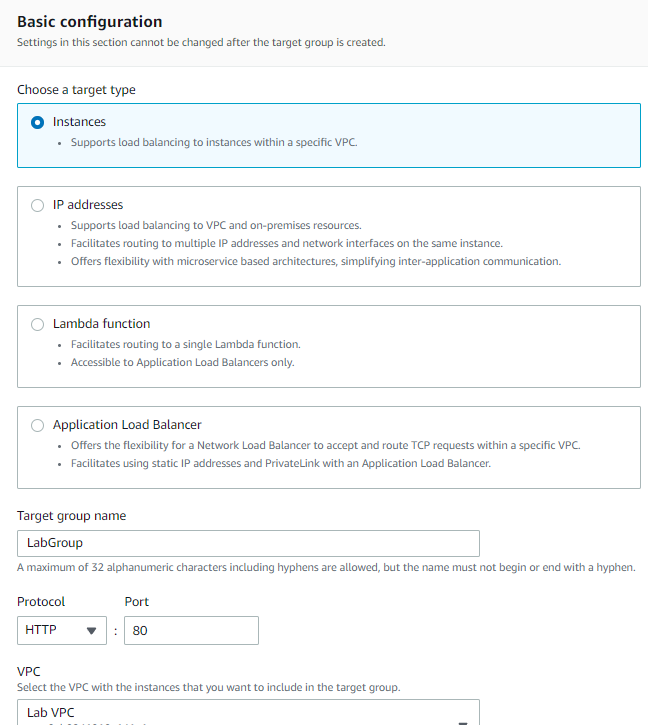
**Task 2: Create Load Balancer**

You will create a load balancer that balances traffic across multiple EC2 instances and Availability Zones.

1. On the navigation panel on the left, access Target Groups.

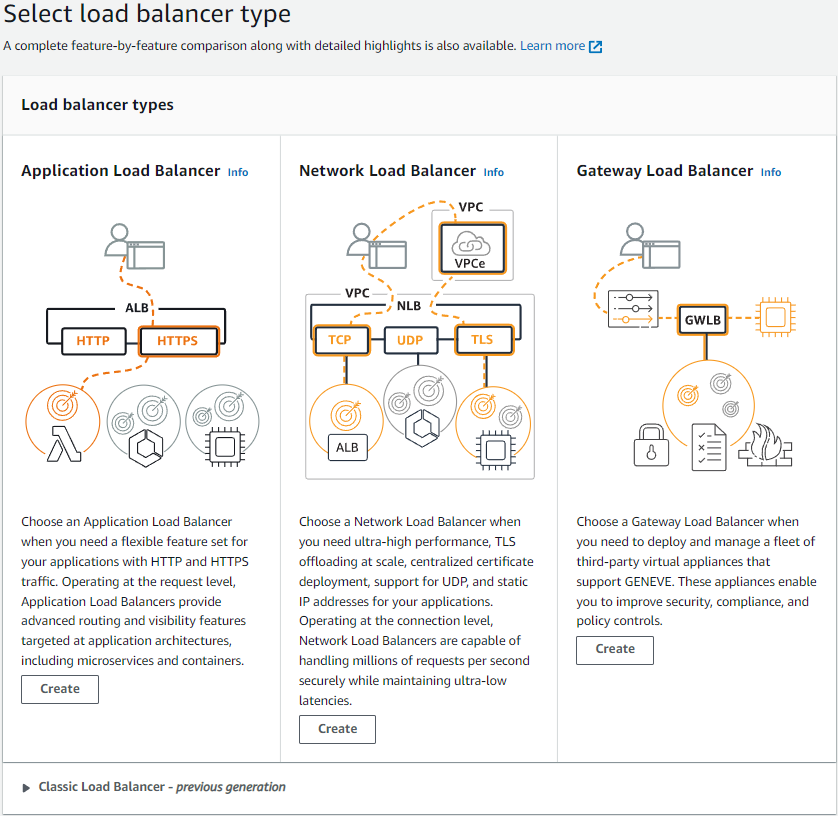
**Target Groups:** Defines for the load balancer the locations of where to send traffic. The *Application* Load Balancer can send traffic to multiple Target Groups of different URLs.

**Targets:** The instances that will respond to requests from the Load Balancer. The targets of where to send traffic.

1. Configure the target group:
   1. Target type: Instances
   2. **Target group name: *LabGroup*
   3. VPC: *Lab VPC*

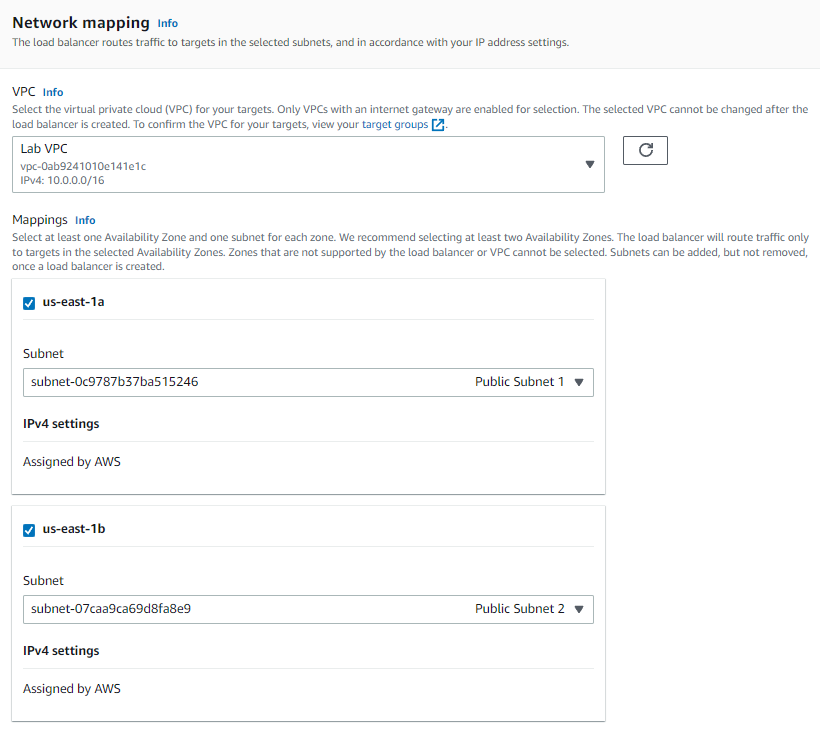
Configuring Target Group.

Selecting between different types of Load Balancers.



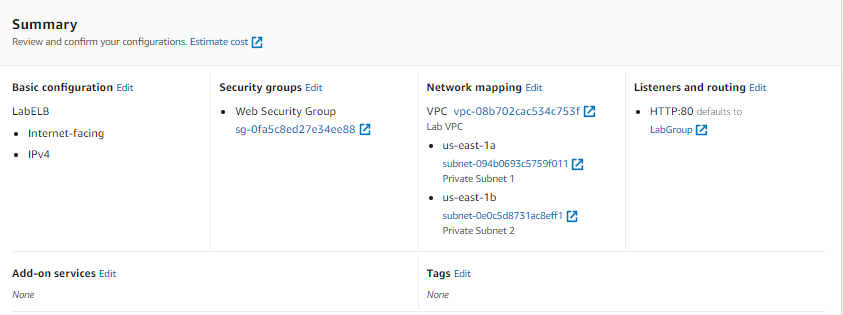
1. Continue by pressing Next. There are no web applications yet, so you do not need to register targets as of now. Finalize and create target group.
2. On the navigation panel on the left, access Load Balancers.
3. At the top, select Create Load Balancer. You will use an Application Load Balancer.

**Application Load Balancer:** A type of load balancer. It operates at the application layer (layer 7) that routers traffic to EC2 instances, containers, IP addresses, or Lambda functions.

1. Choose Create and configure:
   1. Load Balancer name: *LabELB*
   2. Under Network Mapping:
      1. VPC: *Lab VPC*
      2. The load balancer will face the internet. Select Public Subnet 1 and Public Subnet 2. Each one is displayed under their own Availability Zone.

Network mapping and defining the Public subnets of the availability zones.

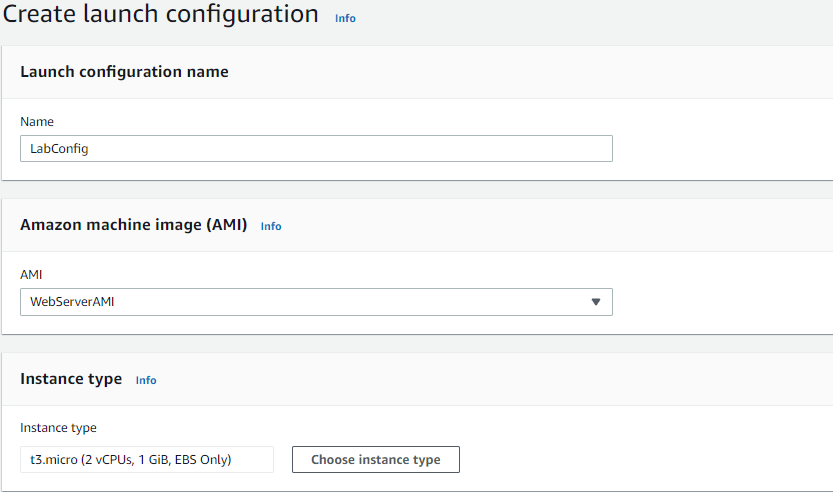
* 1. Under Security Groups:
     1. Select Web Security Group from the dropdown menu.
     2. Remove the default security group.
  2. For the Listener HTTP:80 row, set the default action to forward to LabGroup.

1. Finalize and create the load balancer. You may view the load balancer. It will be in a state of provisioning.

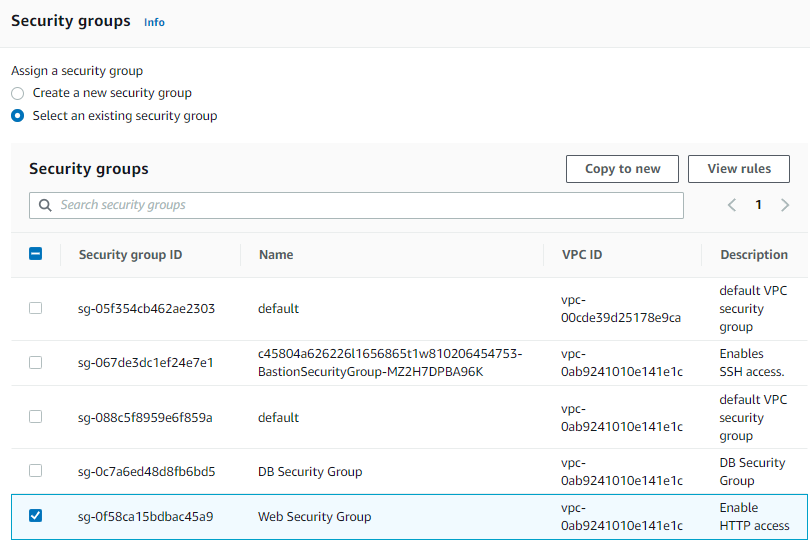
Summary of Configuration before finalization.

**Task 3: Create a Launch Configuration and the Auto Scaling Group**

You will create a launch configuration for the auto scaling group. It acts as the template that the Auto Scaling uses to launch EC2 instances.

1. On the navigation panel on the left, access Launch Configurations.

Creating AMI (image) from existing instance.

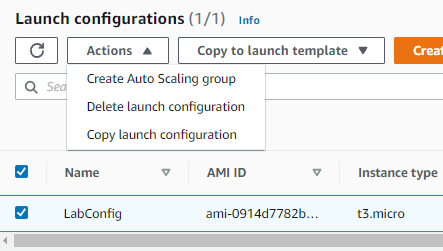
1. Launch Configuration and configure:
   1. Launch configuration name: *LabConfig*
   2. Amazon Machine Image (AMI): *Web Server AMI*
   3. Select Choose instance type and choose *t3.micro*.
   4. Monitoring: *Enable EC2 instance detailed monitoring within CloudWatch*
2. Under Security groups:
   1. Select an existing security group.
   2. Select *Web Security Group*.

Selecting Security Group.

1. Under Key pair, configure:
   1. Key pair options: *Choose an existing key pair*
   2. Existing key pair: *vockey*
   3. Select the acknowledgement checkbox.
   4. Finalize and create launch configuration.

1. Now you will create the Auto Scaling group that uses this launch configuration. Select the *LabConfig* Launch Configuration.

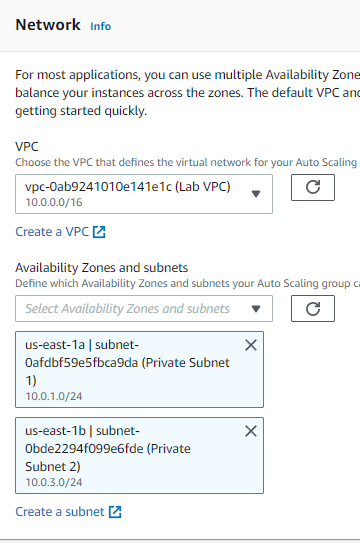
From the Actions menu, choose *Create Auto Scaling group* and name it *Lab Auto Scaling Group.*



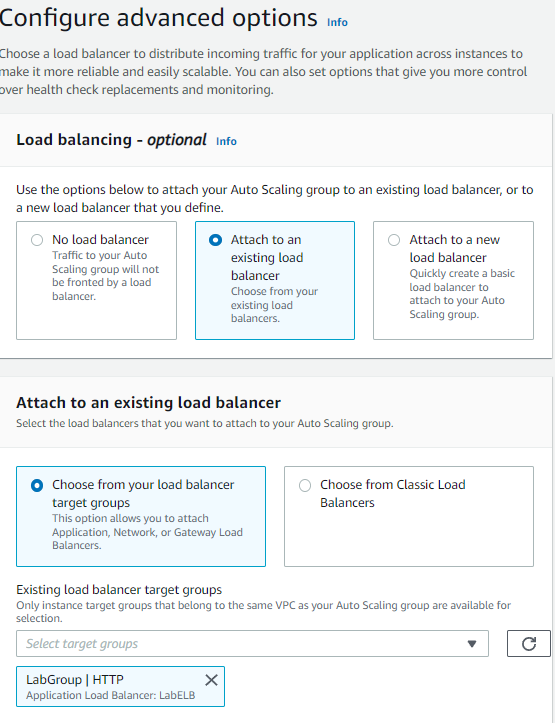
Creating an Auto Scaling Group using the launch configuration.

1. Continue by pressing next. On the Network page configure:
   1. Network: *Lab VPC*
   2. Subnet: Select *Private Subnet 1 (10.0.1.0/24)* and *Private Subnet 2 (10.0.3.0/24)*

This will launch new EC2 instances in private subnets across the availability zones.



Selecting networks for the Auto Scaling Group.

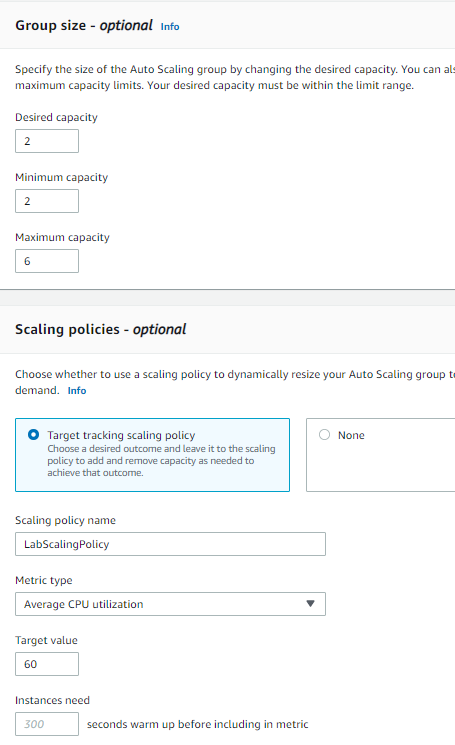
1. Continue by pressing next. On the Load Balancing panel, choose “Attach to an existing load balancer”. Use the dropdown menu to select *LabGroup.*
2. On the Additional settings panel, *Enable group metrics collection within CloudWatch*.

Attaching to an existing load balancer.

CloudWatch captures metrics at 1-minute intervals, allowing Auto Scaling to react quickly to changing usage patterns.

1. Continue by pressing next. Under Group Size, configure:
   1. Desired capacity: 2
   2. Minimum capacity: 2
   3. Maximum capacity: 6

This means that Auto Scaling will automatically add or remove instances so there will always be between 2 and 6 instances running.

1. Under Scaling policies, select *Target tracking scaling policy* and configure:

Lab Policy name: *LabScalingPolicy*

Metric type: *Average CPU Utilization*

Target value: 60

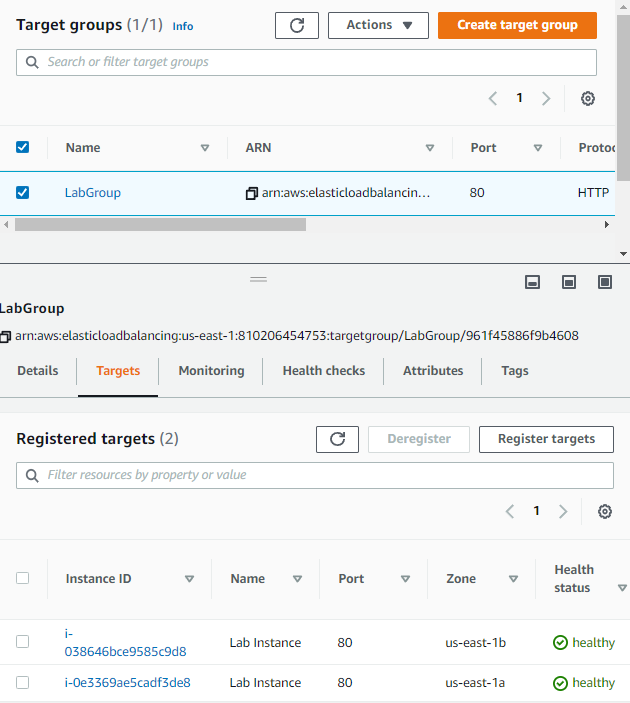
Configuring desired group size and scaling policies.

This means that Auto Scaling will maintain an Average CPU Utilization across all instances at 60%. It will add or remove capacity to keep that metric at 60%, responding to fluctuations in load patterns.

1. Continue by pressing next. You will use default settings for scaling events. Continue again. Tags are sent applied to the instances that are launched as well. Add a tag and configure:
   1. Key: *Name*
   2. Value: *Lab Instance*
2. Continue by pressing next. Finalize and create the auto scaling group. As of now, new instances will be launched to reach the desired count of 2.

**Task 4: Verify Load Balancing Functionality**

You will verify if Load Balancing is working correctly by checking targets and sending a request in the form of a DNS name browser search.

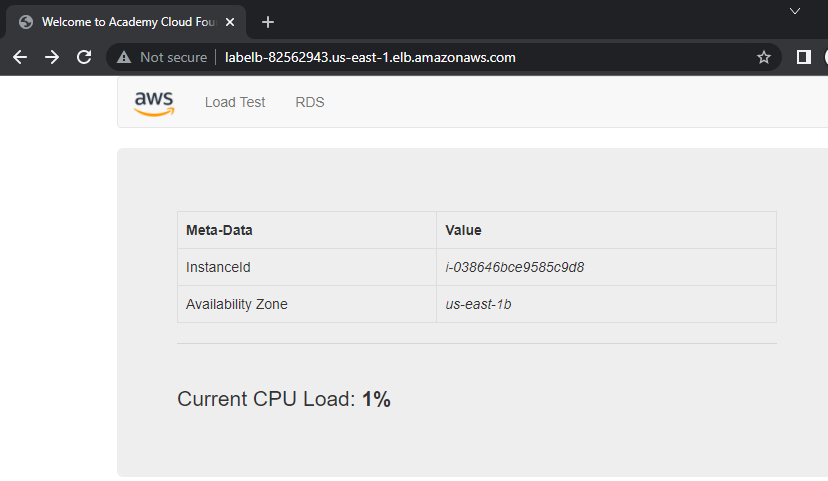
1. On the navigation panel on the left, access Instances. There should be 2 new instances named Lab Instance that were launched by Auto Scaling.
2. Let’s confirm that these instances passed their Health check. On the navigation panel on the left, access Target Groups.
   1. Choose LabGroup and Targets tab. Two targets should be listed.
   2. The statuses of these instances should appear healthy. You may have to wait a moment and refresh for updates.

Lab Instance Targets are listed in the target group.

1. We can now access the Auto Scaling group via the Load Balancer. Access Load Balancers on the left navigation panel.
2. In the lower panel, copy the DNS name of the load balancer, omitting “A Record”.

It should look similar to: *LabELB-1998580470.us-west-2.elb.amazonaws.com*

1. Past the DNS name into a standard incognito internet browser.

The application indicates that the Load Balancer received the request, sent it to one of the EC2 instances, and passed back the result.

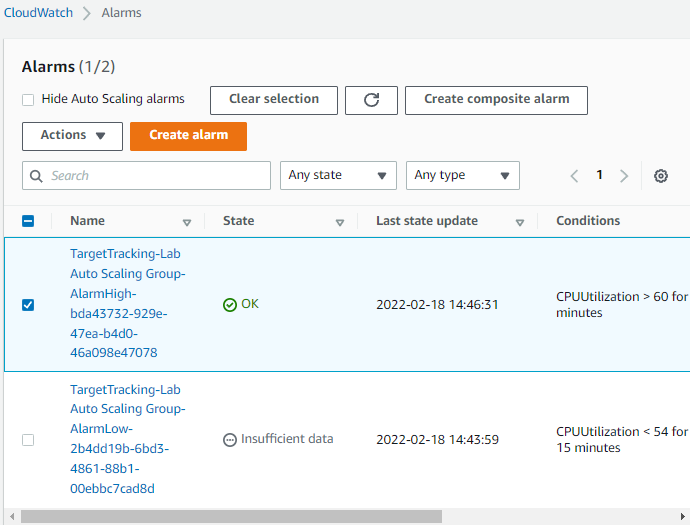
DNS address entered into an internet browser. Displaying application information about load balancer.

**Task 5: Test Auto Scaling**

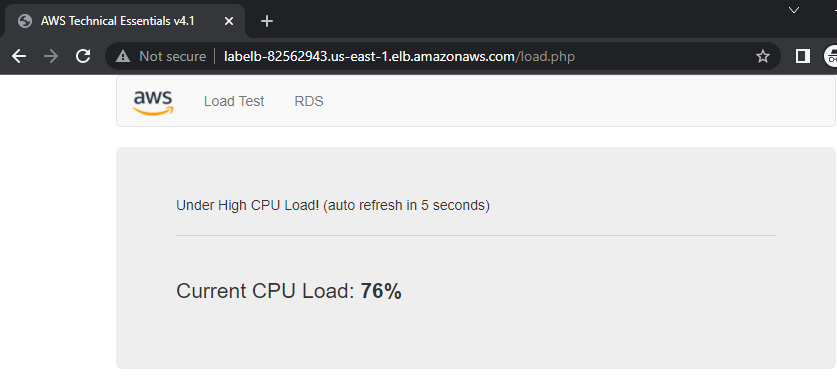
You will now increase the load to cause Auto Scaling to add additional instances.

1. On the AWS management console, access CloudWatch. Do not close the application tab of the Load Balancer.
2. On the left navigation panel, select All alarms.

These alarms were created by the Auto Scaling group to track the CPU load. Auto Scaling will attempt to keep the average load at 60% while also staying between 2 to 6 instances.

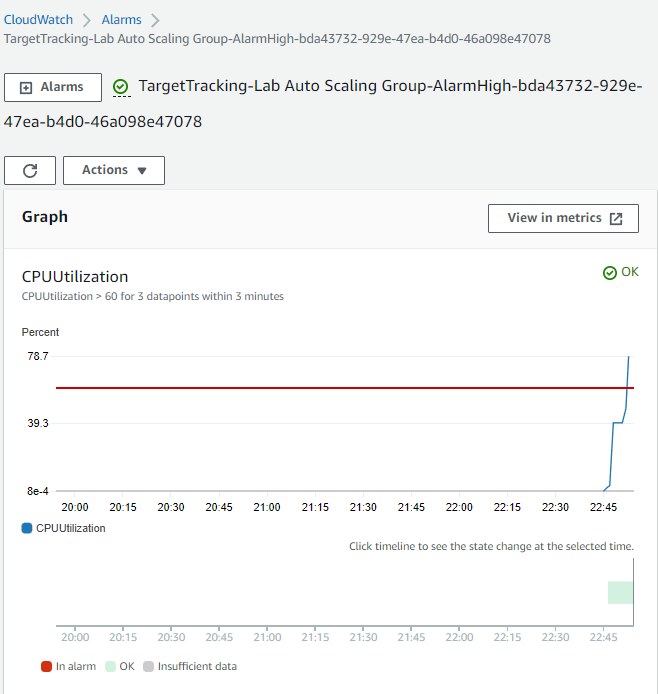
1. Click the OK alarm, named *AlarmHigh*. “OK” indicates that the alarm hasn’t been triggered. It triggers when CPU utilization is greater than 60%.

Two alarms. Selecting the *AlarmHigh* one.

1. Return to the browser tab and click Load Test at the top. This will manually trigger high loads, activating the alarm.

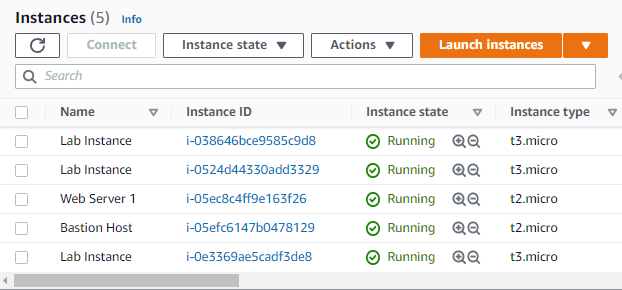
Activating Load Test.

1. Return to the CloudWatch console. In less than 5 minutes, the AlarmHigh status should change to *In alarm*.

Once the AlarmHigh chart crosses the 60% CPU percentage for more than 3 minutes, Auto Scaling will add additional instances.

CloudWatch AlarmHigh Alarm CPU utilization chart displays use over 60%

1. Once the AlarmHigh enters the *In alarm* state, you can view the instances that were launched.
2. Access EC2 in the management console.

More than two instances, labeled Lab Instance, are now running because Auto Scaling created them in response to the Alarm.

Additional Lab Instances were created due to Auto Scaling Response.

**Task 6: Terminate Unneeded Instance**

You will now terminate the instance that was used to create the AMI for the launch configuration. It is unneeded.

1. Select Web Server 1.
2. Under the Instance State dropdown menu, select “Instance State > Terminate Instance”.
3. Confirm and terminate.

**Conclusion**

The AWS Auto Scaling and Elastic Load Balancing are complex, yet efficient tools required for more sophisticated levels of optimization. They provide flexibility during fluctuating data usage, ultimately reducing cost and management while providing convenient service. In this lab we explored the multi-step configuration of creating an application load balancer, one of many types of load balancers for traffic management. We also used CloudWatch to monitor and see the thought process of the Auto Scaling response to a trigger that we defined. Auto Scaling has great flexibility and several parameters that serve to be a flexibility and multi-purpose service for quality performance.