Brit Stevens 1/7/23

AWS Labs 4-6

**Purpose:** The purpose of these labs is to expose and prepare us for more AWS concepts and learning how to configuration them for real life scenarios. These labs were focused on EBS, RDB, and Elastic Load Balancers. It taught us how to set up load balancing on EC2 instances, to configure EBS storage options for different scenarios with snapshots, to build databases, and many different features of automation given with AWS services.

**Background Information on lab concepts:**

**These labs use many similar concepts listed in the previous write up.**

**EBS:** Elastic Block Storage (EBS) offers persistent storage for EC2 instances. EBS volumes are network-attached and are an independent service from an EC2 instance. EBS volumes are highly available and highly reliable volumes like S3 that can be used as an EC2 instances boot partition or attached to a running EC2 instance as a standard block device. Unlike S3 each EBS is meant to be attached to only one EC2 Instance and is only accessible with EC2 instances. It is less accessible than S3 but has much lower latency which is essentially the speed at which you can receive your data from the storage option. As per AWS’s pay as you use model, if you stop an instance, you will only pay for the storage used to keep the instance’s state. They are also automatically replicated across a single availability to ensure your data is secure. Like S3, you can create snapshots in time of your EBS volumes then store them in an S3 volume.

**Snapshot:** A snapshot is like a picture in time of a certain volume. It is a backup of all the data in a volume you choose that can be stored in case of future corruption of data or accidental deletion of a document for say. This is an essential feature for data integrity on the user end as on the server end it is secured by Amazon.

**Tag Key:** A tag is like a label you can put on any instance or object created in AWS. It is used for more efficient documentation and management by administrators. The key of a tag is telling what you want this tag to be like Name, Role, ID, Date, etc. It used to tell the viewer what this tag represents.

**Tag Value:** Like the key, the value of tag can be anything, but it should specific what is listed in the key. For example, if the Key is name the value should be Webservice\_Volume or Video\_Files. If the Key is Date the value should be something like Jan-1-2021. This is how tags are meant to be used but are fully customizable to however you wish to organize your data. It is always better to be specific and use more tags than less tags for efficient documentation. Tags are case-sensitive so you must ensure you are consistent with how you type your tag between objects. A tag usually looks as such; “User:Owner”.

**SSH:** Secure Shell (SSH) is a protocol that allows securely sending commands over networks even if unsecure. It uses a cryptography to authenticate and encrypt connections between two devices. It does this encrypting using public and private keys. The owner and recipient of the file both have a public key which everyone on the internet can see and private keys that are only shared between those two. By matching public and private keys SSH ensure that’s only the owner and recipient can decrypt the data using those keys. This makes it so no one who is attempting to view this data can decrypt it. This is mainly used for controlling servers remotely, transferring secure files, managing infrastructure, etc.

**PPK:** PPK files are Private Key Files used for keys generated by or for PuTTY to access another party through the application. PuTTY is a software that allows remote connect to another device through a network or direct connection with an ethernet cable. PuTTY uses SSH to do this and consoles you into a CLI interface to enter commands or devices like a router or switch but also AWS services like an EC2 instance.

**RDS:** Relational Database Service (RDS) is a service that allows you to create a relational database in the cloud through AWS. By being relational it means that all datapoints have a predefined relation to each other that is recorded in a set of tables that show the relation. This system makes it easy to set up, operate, and scale to whatever extent you require. RDS also manages the administration tasks that would otherwise be a time-consuming job. RDS has six options of six familiar database engines to choose from: Amazon Aurora, Oracle, Microsoft SQL Server, PostgreSQL, MySQL and MariaDB.

**Amazon Aurora:** A database that is compatible with PostgreSQL and MySQL with high speed and availability of other high-end databases but has the simplicity of the AWS dashboard for management. It is faster than MySQL with greater security at a low cost. It has automatic backups, region replication, and has full integration with other AWS services. Its high availability through AWS is due to its automatic multi-AZ backups. Due to its full management by Amazon RDS, there is high performance security, and the administrator won’t need to do the time-consuming management task such as software upgrades, backups for disasters, or storage management.

**Oracle:** Like Aurora it is fully managed by AWS and has all the benefits of their high availability and speed. The main difference is its license flexibility. If you are moving your Oracle database to the AWS cloud, you can choose to use your already owned Oracle license and just pay for the Amazon RDS service with no cost on their Oracle license option. You can choose to reserve a database for multiple years making a significant cost decrease in the pay as you use model with an upfront price of the reservation.

**Microsoft SQL Server:** Like the previous options, this database is fully managed by Amazon RDS with the benefits it gives. MSQL has the same license flexibility plan as Oracle for all the versions available and the option to purchase a reserved DB. MSQL differs in that there are different version types including Express, Web, Standard, and Enterprise that each have different use cases and parameters.

**PostgreSQL:** Like the previous options, this database is fully managed by Amazon RDS with the benefits it gives. This is the preferred Relation database due to its vast features and options to run stored procedures in many coding languages like C++ or Java for example. It is a very powerful database with the emphasis on extensibility of enterprise class systems.

**MySQL:** Like the previous options, this database is fully managed by Amazon RDS with the benefits it gives. It is an open-source relational database used by many web-based applications. Amazon RDS keeps the familiar aspects and configurations of standard MySQL but in the cloud. It has full compatibility to move a MySQL server to the cloud.

**MariaDB:** Like the previous options, this database is fully managed by Amazon RDS with the benefits it gives. Developed by the same developers of MySQL it has many of the same features improved upon and is essentially the same option as MySQL for your RDS engine. It relies on user preference which engine to choose.

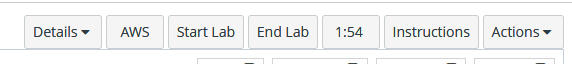
**Amazon RDS Multi-AZ deployments** provide enhanced availability and durability for Database instances with the automatic creation of a primary DB instance and synchronously replicates the data to a standby instance in a different Availability Zones.

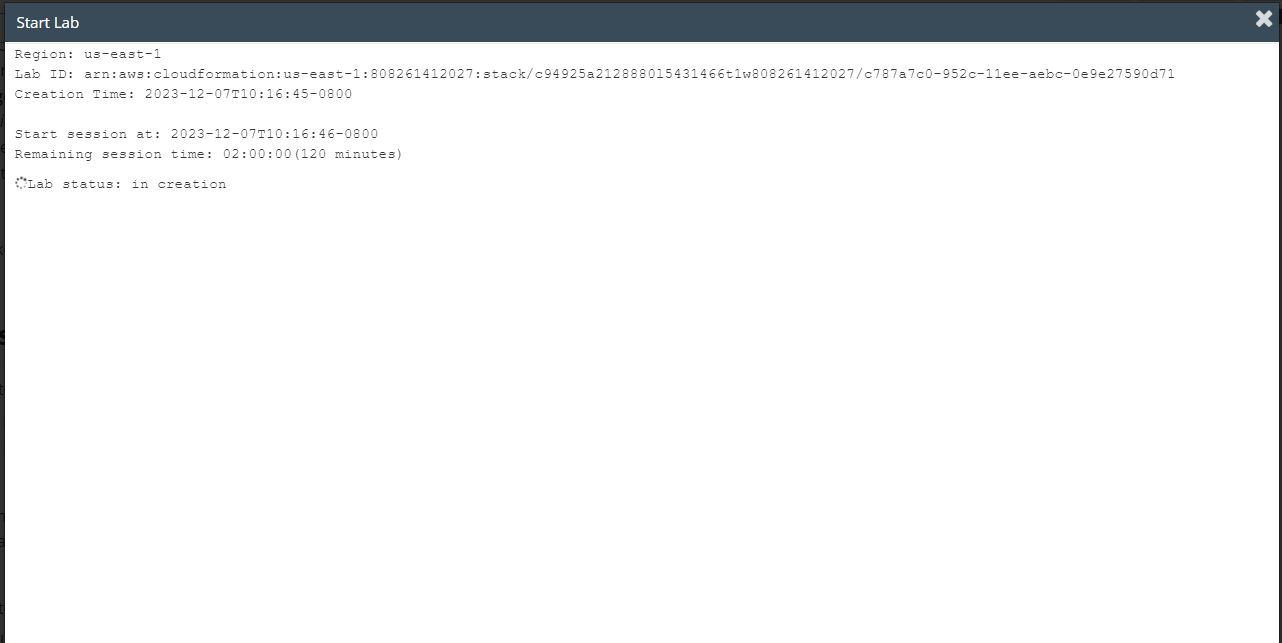
**Elastic Load Balancing:** an option is some services that automatically distributes incoming application traffic across multiple Amazon EC2 instances. It enables you to achieve fault tolerance in your applications by providing the required amount of load balancing capacity needed to route application traffic. We completed this with

**Auto Scaling:** An AWS service that is not its own entity but is under many different services. It helps you maintain application availability and allows you to scale your EC2 capacity out or in automatically according to conditions you define. You can use Auto Scaling to help ensure that you are running your desired number of Amazon EC2 instances. Auto Scaling can also automatically increase the number of EC2 instances during demand spikes to maintain performance and decrease capacity during lulls to reduce costs. Auto Scaling is well suited to applications that have stable demand patterns or that experience hourly, daily, or weekly variability in usage.

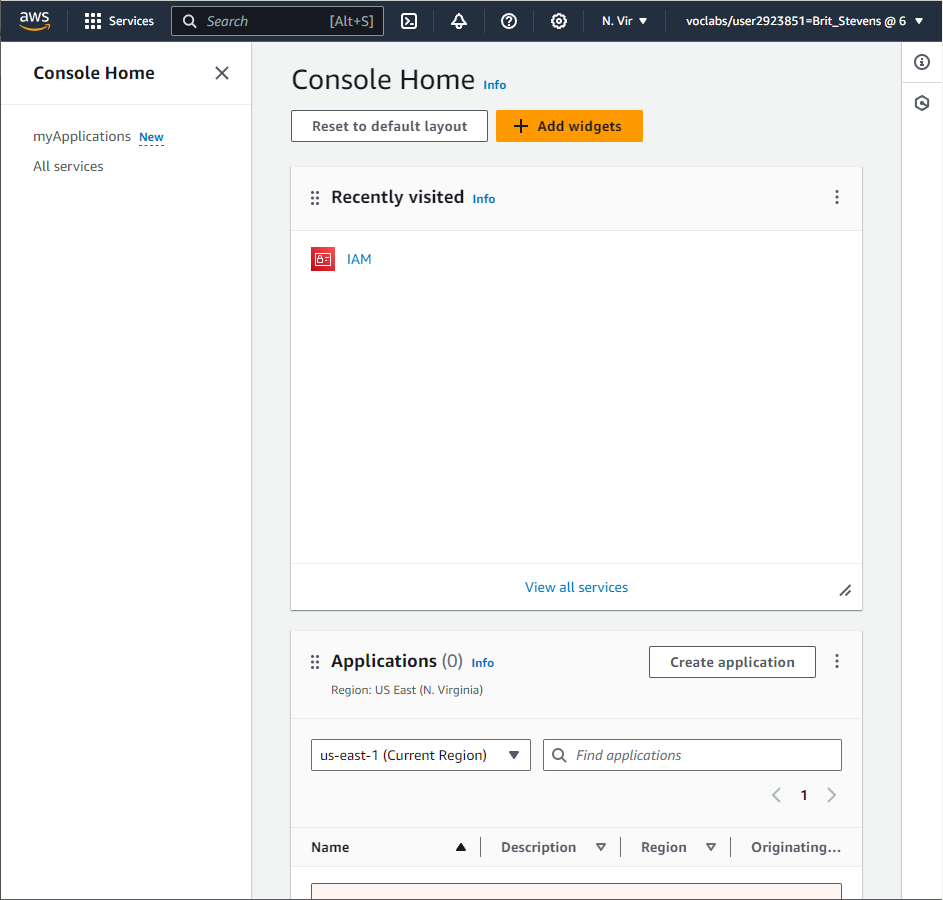
**CloudWatch:** An AWS service that is used to monitor resources and applications to automatically create reports on that data and react to data. Users can set alarms as well to alert them if the alarm triggers. This could be if a certain file is moved that shouldn’t have been or if your EC2 instances are at max capacity and need to scale up. It can automate resource planning and lowering costs when your set parameters are met to make a change. If an alarm occurs, it will give you reports on the root cause of that alarm to reduce the time to fix the issue. If set up it will also automatically scale for you to a set amount in an alarm.

**Lab Summary: AWS EBS Practice Scenarios**

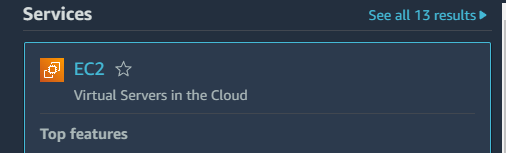
****

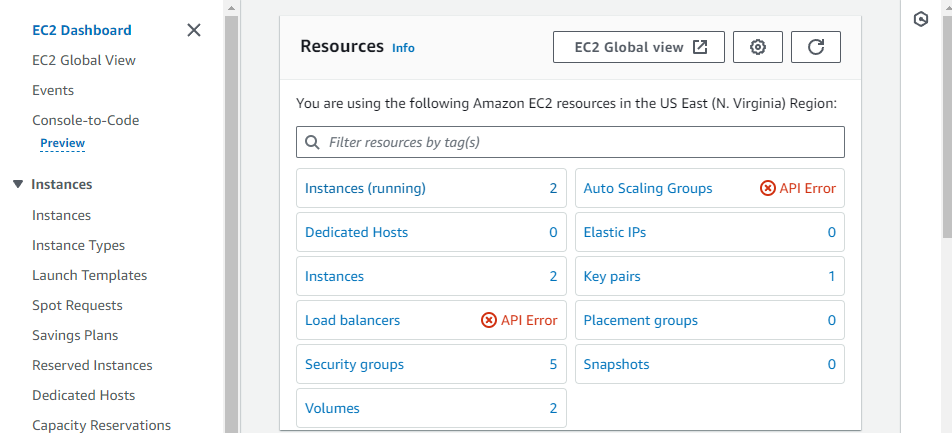
****

click start lab and wait until the prompt box says lab is ready.



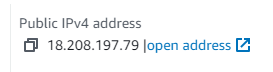
Click AWS and you will see a website like this. Click on services and search for EC2.



Once in the dashboard select instances (running).

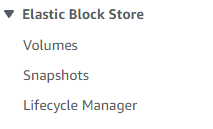


You will see the instance Lab already created for you.



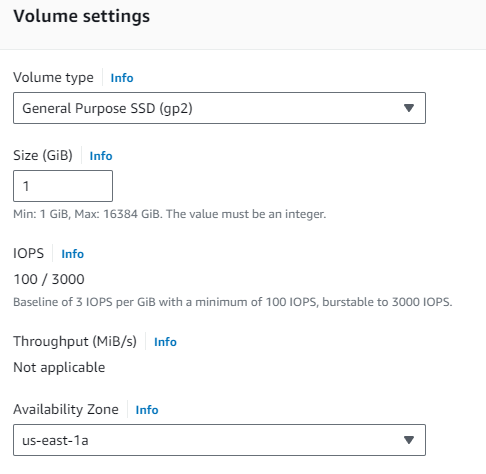


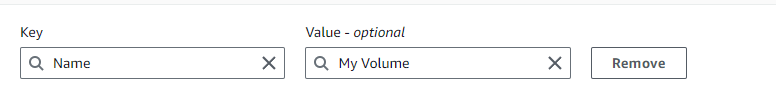
Note your Availability Zone and IP of the instance for future reference.



On the lefthand side select volumes under EBS.

Once in the volumes select Create Volume to create a new volume for this lab.



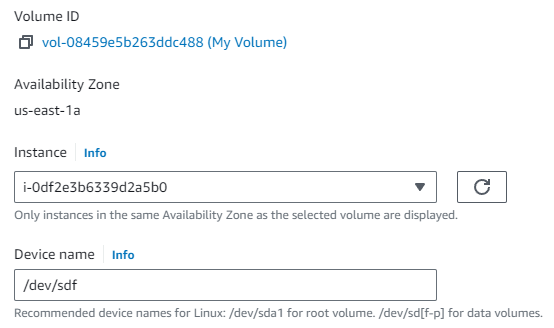


Use the above settings with the Availability zone differing depending on your AWS and add a tag.

Create the volume.



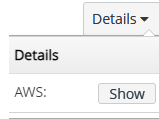
We are now going to attach the volume we created to our EC2 instance. Select the volume you just configured and select actions -> attach volume.



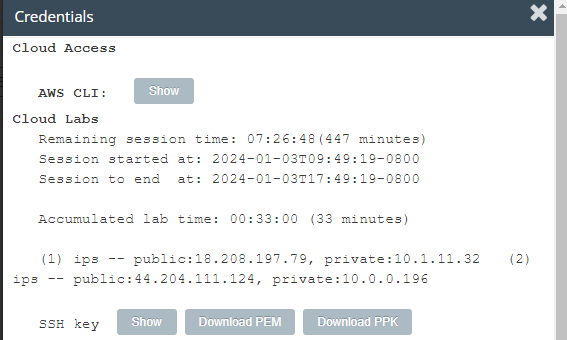
Now attach the volume to the Lab instance and note the device name.

Once attached you will connect to the EC2 instance using an SSH program like Putty.

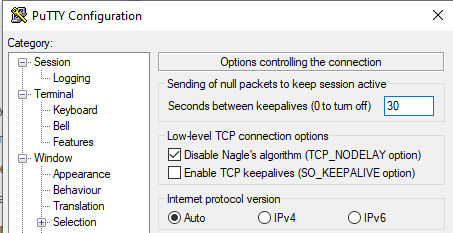
To do so go back to the AWS academy page and select details -> show.

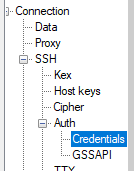


In the details download the PPK of the SSH key.

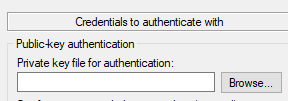


Open PuTTY -> select connections and turn the keepalives to 30 to ensure PuTTY does not disconnect from the EC2 Instance.

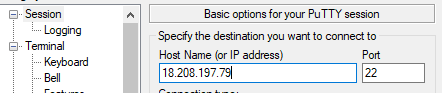




Still under Connections select credentials.



Click browse and use the PPK you downloaded earlier as the Private Key file.



Click Session and type in the IP of recorded earlier of the Instance. Select open in the bottom right.

If a security prompt comes up, select accept to access the instance.

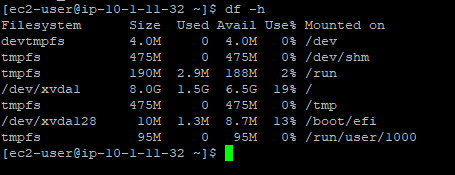


Login as ec2-user.



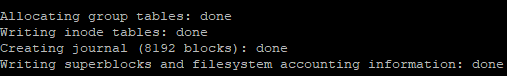
If successful, you should see a prompt like this.

You will now add a new volume to the instance through this SSH connection.



Type in df -h to view your instance’s storage. You will notice that your 1G EBS volume is not listed so you must add it.

To do so type “sudo mkfs -t ext3 /dev/sdf” to create a file system on the linux instance. This allows us to allocate space using the volume.

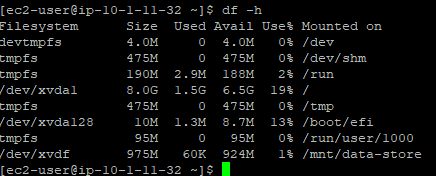


After allocation is complete type “sudo mkdir /mnt/data-store” to allow mounting of volume by creating a directory which gives the volume reference to other files.

Now that you have a directory you can mount the volume with “sudo mount /dev/sdf /mnt/data-store”.

Your volume is now mounting.

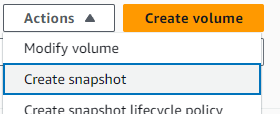
To allow for automated mounting type “echo "/dev/sdf /mnt/data-store ext3 defaults,noatime 1 2" | sudo tee -a /etc/fstab”



Type df -h again to confirm your volume is enabled.

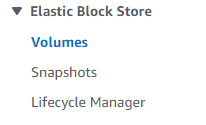
Type “sudo sh -c "echo some text has been written > /mnt/data-store/file.txt"” to create a text file on your volume for the next experiment.

You will now learn how to create an amazon EBS snapshot.

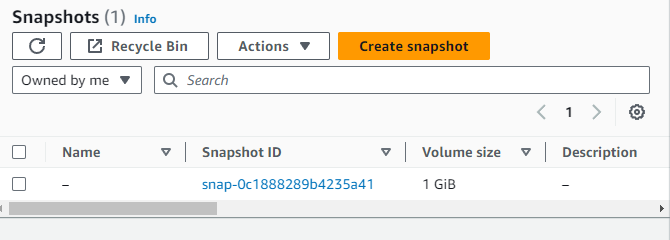


Under volumes where you were before select your volume -> actions -> create snapshot

While creating your snapshot give it the tag key of Name and value of My Snapshot. Click create snapshot.



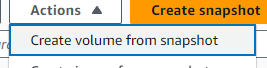
Select snapshots under EBS and look for your snapshot.



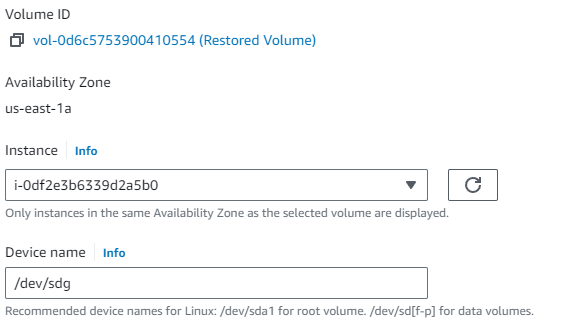
Back in the SSH PuTTY connection type “sudo rm /mnt/data-store/file.txt” to delete the text file on the instance. We will confirm the snapshot is functional by doing this.



By typing “ls /mnt/data-store/” you can confirm your file is deleted.



To restore your snapshot, go to your snapshot -> actions -> create volume from snapshot. Configure it the same as the previous volume with a tag key of Name and Value Restored Volume. Create the volume.



Attach your volume to the lab instance.

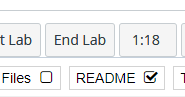
Go back your PuTTY window and create a directory to mount your new storage volume with “sudo mkdir /mnt/data-store2”.

Mount with “sudo mount /dev/sdg /mnt/data-store2”

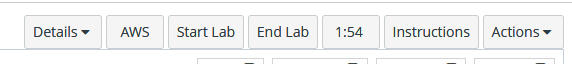
Use “sudo mount /dev/sdg /mnt/data-store2” to verify the file is there on the restored volume.

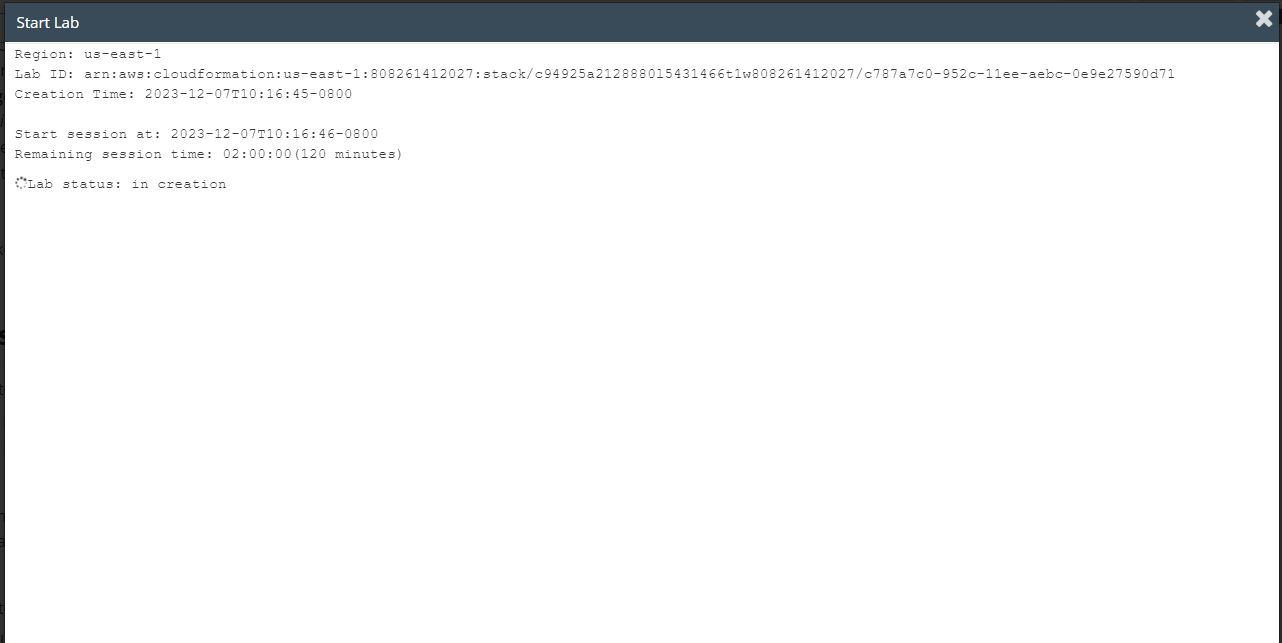


Now click end lab and it’s complete.



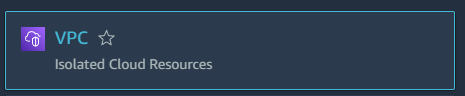
**Lab Summary: AWS Building DB Server**

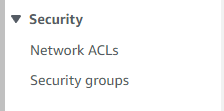
****

****

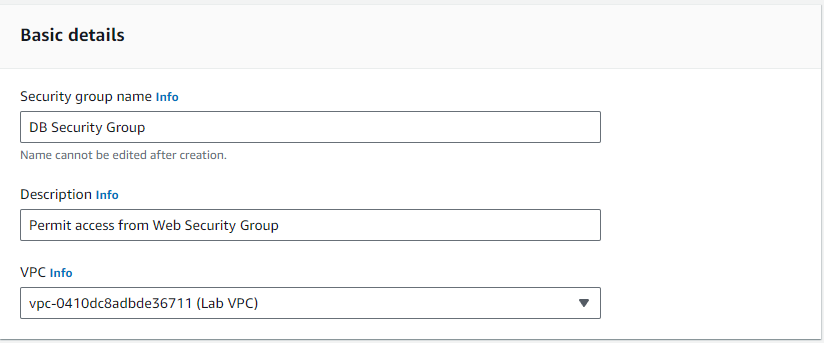
click start lab and wait until the prompt box says lab is ready. Once lab is ready click AWS search VPC for instance.

You will see a website like this. Click on services and search for VPC.

****

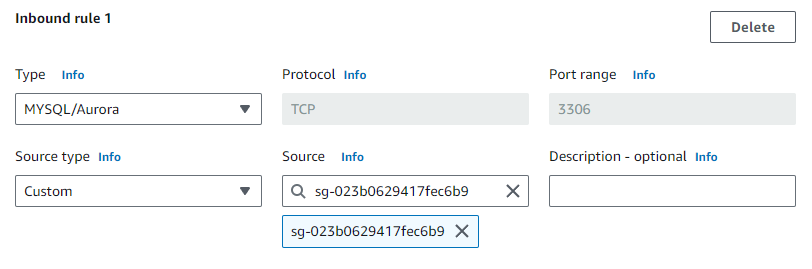
****

Once on the VPC dashboard look at the left-hand navigation pane and select Security Groups.



Click create Security group and use the parameters above.

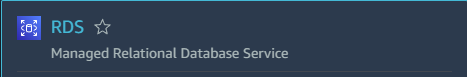
Under Inbound Rules while creating the security group choose add rule.

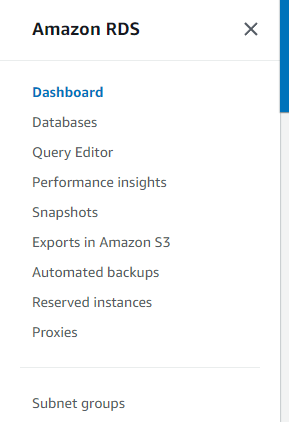


Use the above settings for the inbound rule. For source type sg and click on Web Security Group. This rule will allow will permit any inbound traffic on port 3306 from any EC2 instance that is in the Web Security Group.

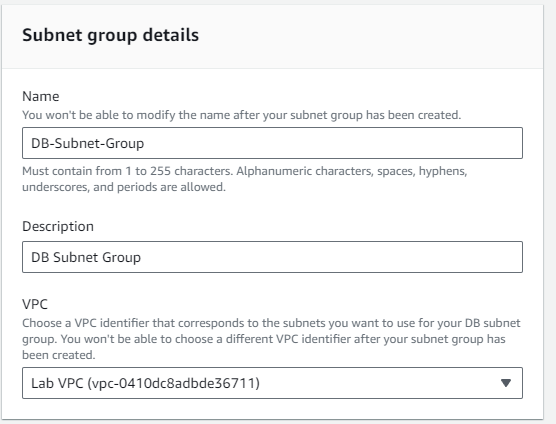
Choose create rule to apply the rule to the security group.

Now that we have created this rule, you’ll use on your RDB we need to create a subnet for the DB group. To do this search RDS in services menu.

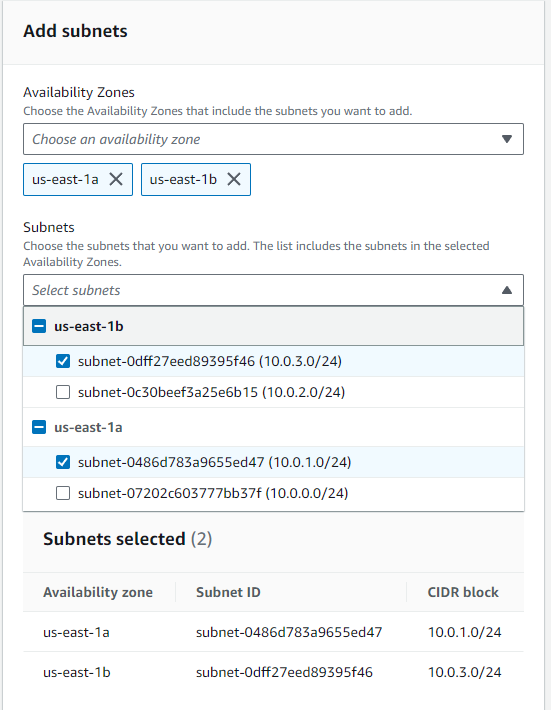




Select subnet groups on the left-hand navigation once in the RDS dashboard. Choose create DB Subnet Group.

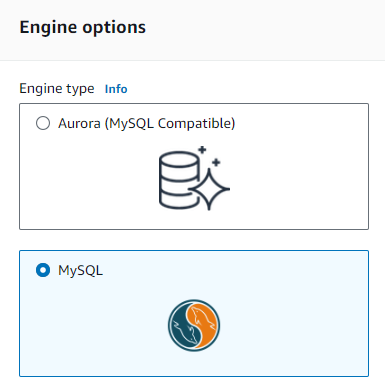


For details use the above settings.

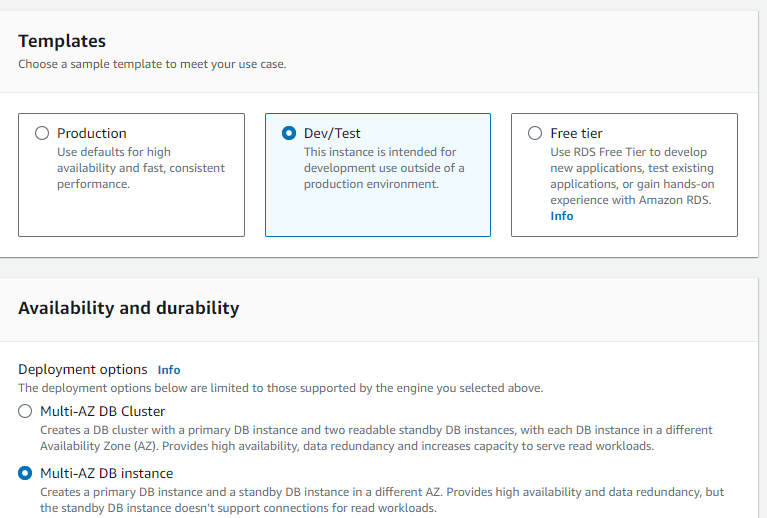


For availiblity zones select us-east-1a and 1b to choose subnets associated with those zones. Select the addresses ranges of 10.0.3.0/24 and 10.0.2.0/24. Click create.

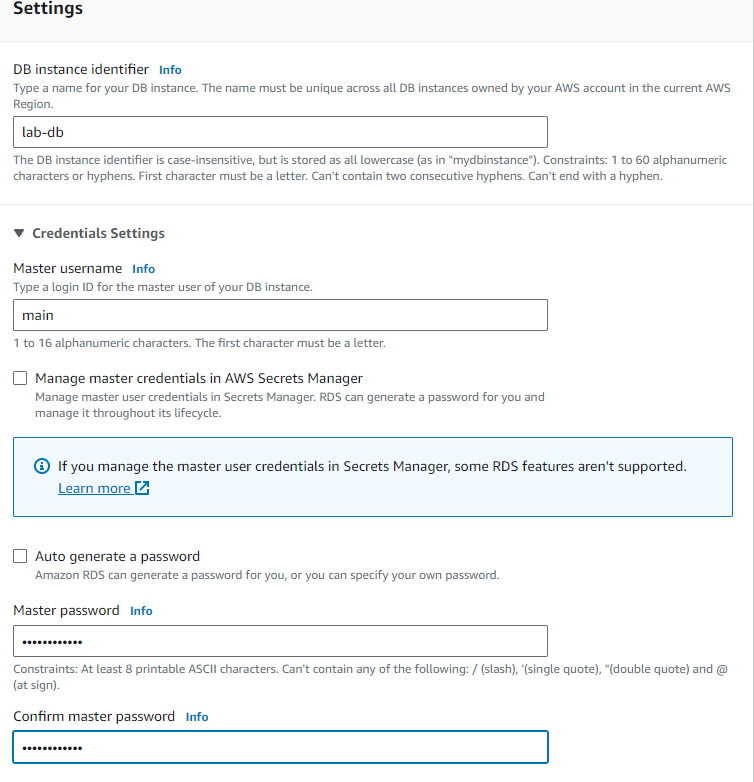
Now you will launch your RDS for your MySQL database instance. Still in RDS for services click database on the left hand menu and choos create database once in.



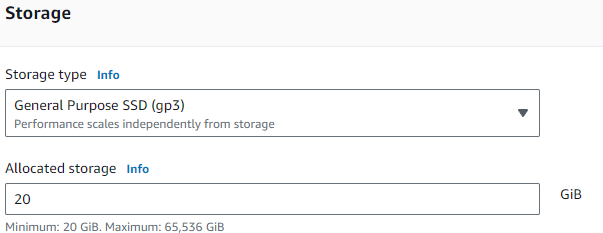
Select MySQL for engine options.



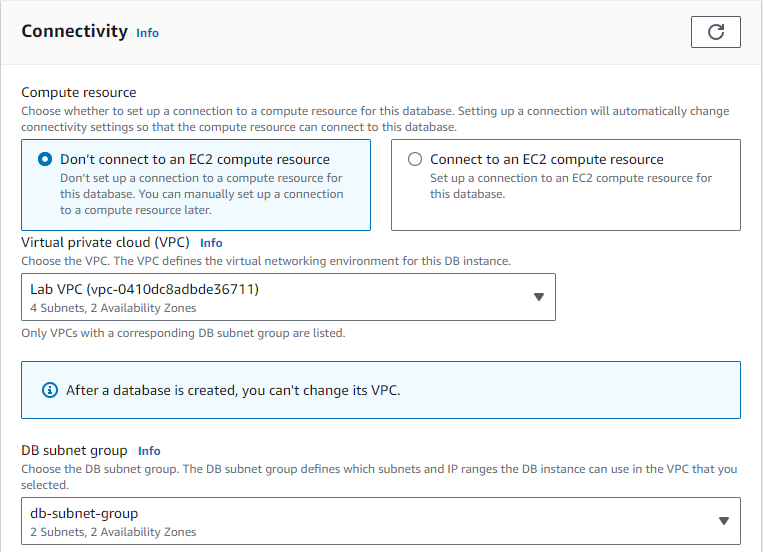
Scroll down to see templates to Dev/Tests. Under Availability and durability select Multi-AZ DB instance.

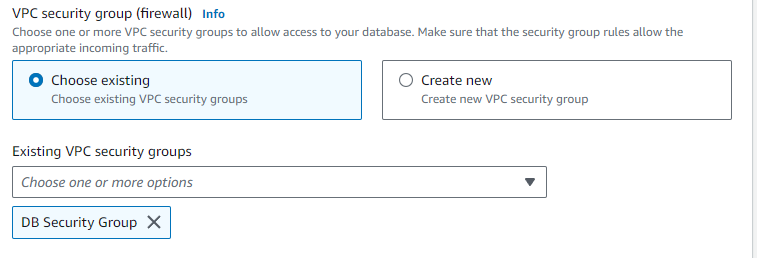


Scroll down to see the settings and use the following as the configuration: DB instance identifier: lab-db, Master username: main, Master password: lab-password, Confirm password: lab-password.

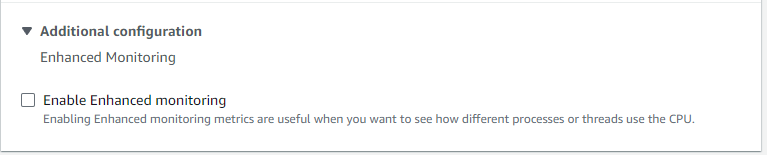


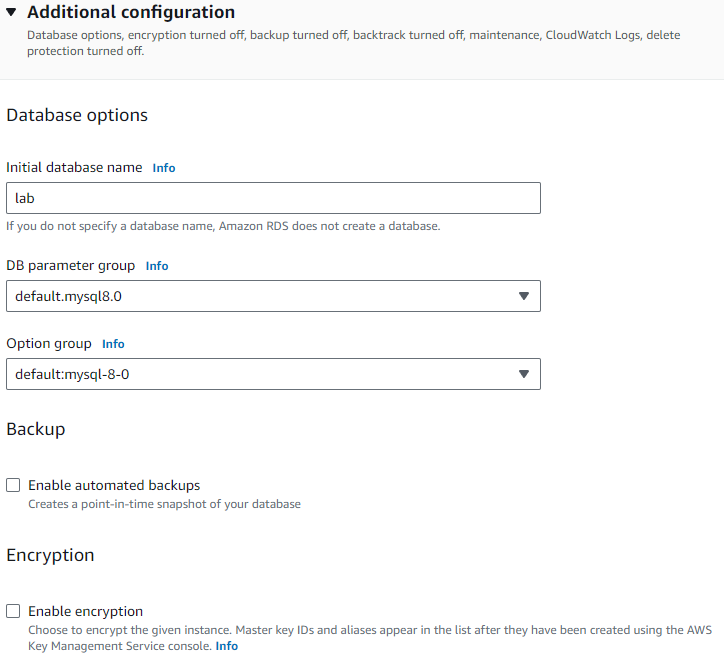
Under Storage configure with the following: Storage type: General Purpose (SSD), Allocated storage: 20.





Under Connectivity configure with the following: Virtual Private Cloud (VPC): Lab VPC. under Existing VPC security groups, from the dropdown list: Choose DB Security Group, deselect default.

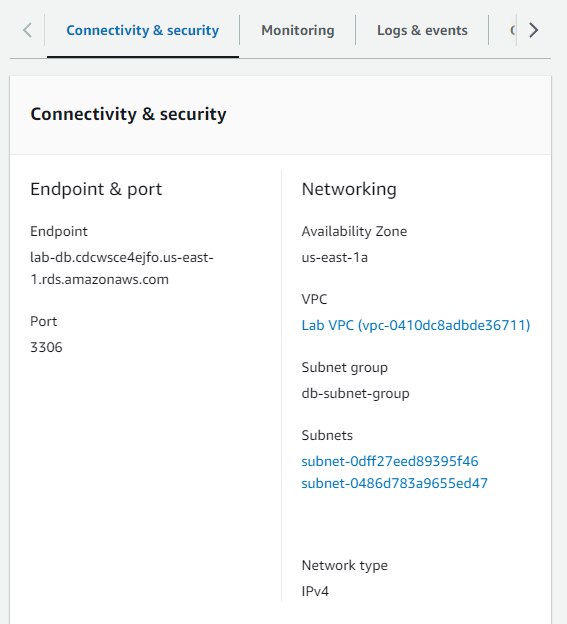




Expand Additional configuration and configure with the following: Initial database name: lab, Uncheck Enable automatic backups, Uncheck Enable encryption, Uncheck Enable Enhanced monitoring.

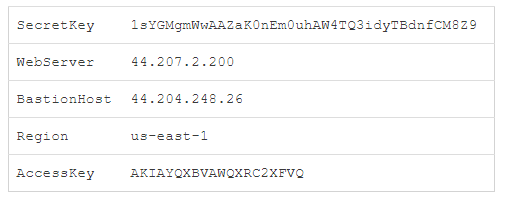
This is not recommened for real life scenarios but will make the database launch faster for this lab. Create the database.

After a few minutes the database will be avaiable, when it is click on the link of your database.

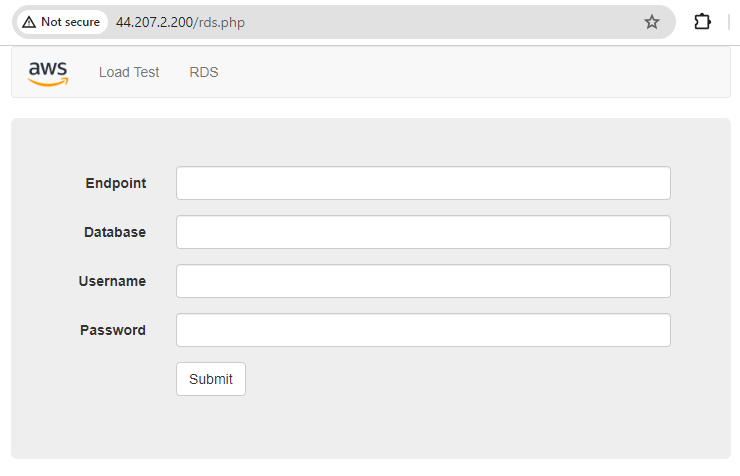


Copy the end point of your DB and save it for later.

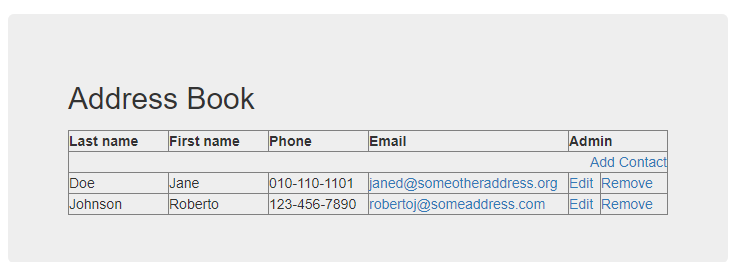
Next to access the database go to your AWS academy page and open details.



Copy the WebServer and type it in the browser.

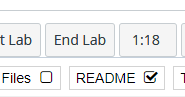


* Click RDS and
* **Endpoint:** Paste the Endpoint you copied to a text editor earlier
* **Database:** lab
* **Username:** main
* **Password:** lab-password
* Choose **Submit**

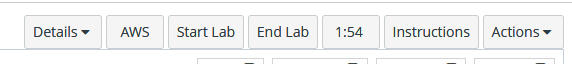


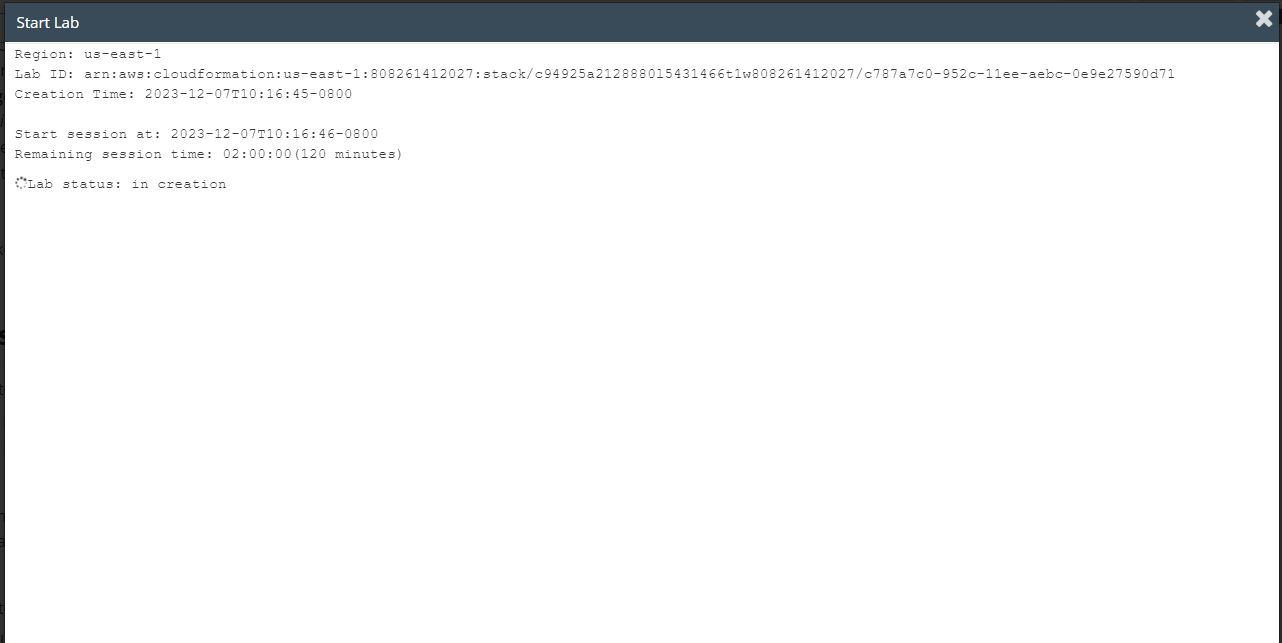
You should see this and you can test your application by editing or removing contacts.

Now click end lab and it’s complete.

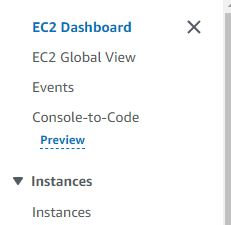


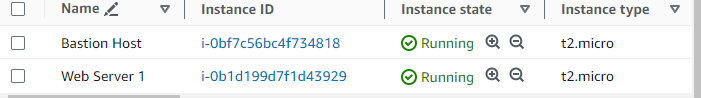
**Lab Summary: AWS Scaling and Load balancing your Architecture.**

****

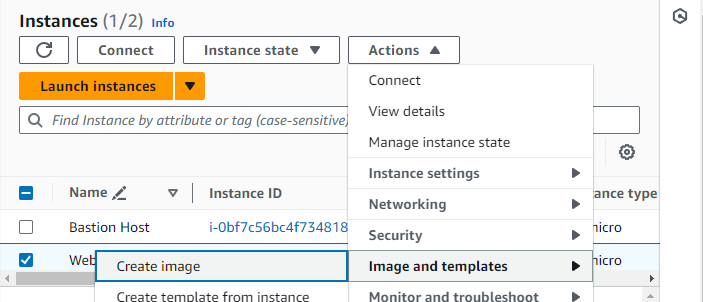
****

click start lab and wait until the prompt box says lab is ready. Once lab is ready click AWS to go to the AWS webpage. Search EC2 and select EC2 instances once in the EC2.

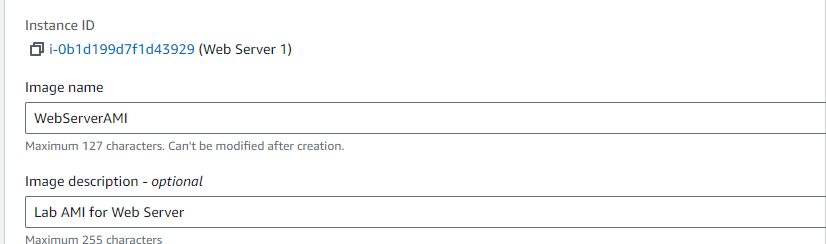




Confirm your instances are running then select Web Server 1. You will be creating an AMI based off this instance.

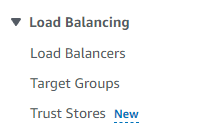


Select create image under actions.

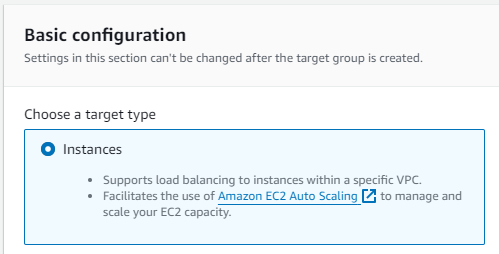


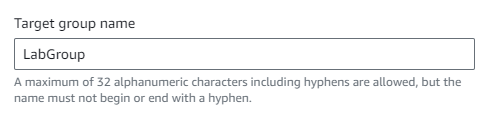
Use the above name a description to make your image easily recognizable. Click create image at the bottom of the screen. This image will be used when launching auto scaling later.

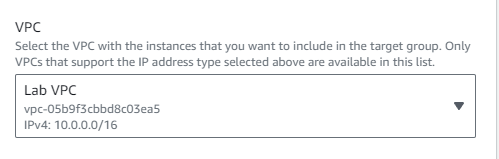
You will now set up a load balancer to balance traffic between EC2 instances and Availability zones.



Still under EC2 select target groups under load balancing. Click create target group.







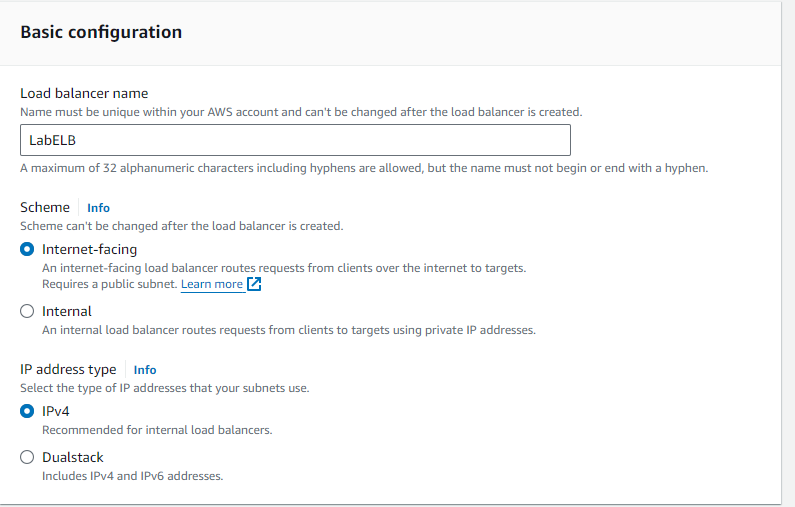
For the basic configuration use the default settings and select instances as target type. Use the name LabGroup. Select Lab VPC as your VPC. Choose next.

Skip the steps shown on the next page because you have no web application instances yet. Click create target group.

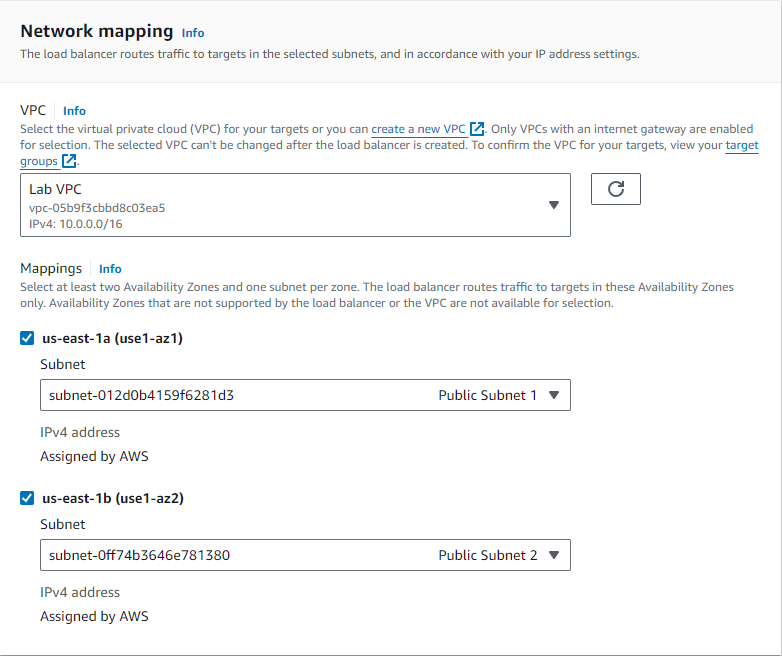
Under Load Balancing select load balancers and click create load balancer.



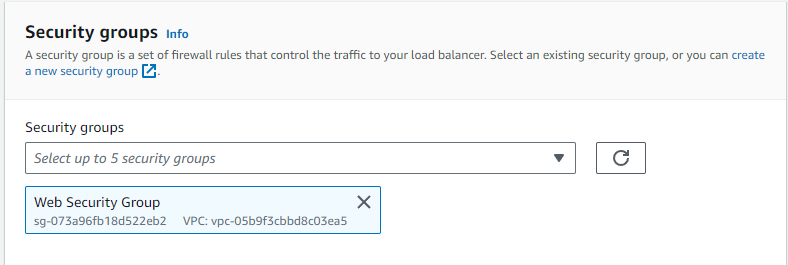
Choose Create application load balancer as your type.



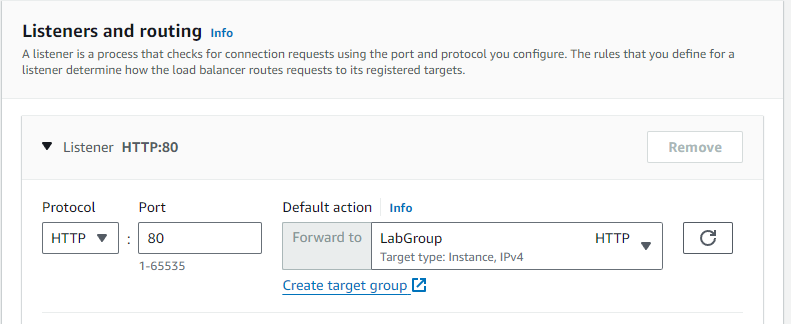
Use the name load balancer name of LabELB and leave the other basic configuration settings as default.



Select Lab VPC as your VPC. Choose the two Availability zones listed and the public subnets associated with them.

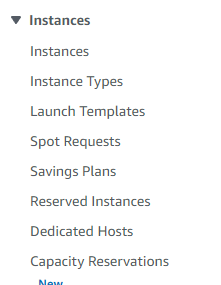


Select Web Security Group for your security group and disable default by clicking the X next to it.

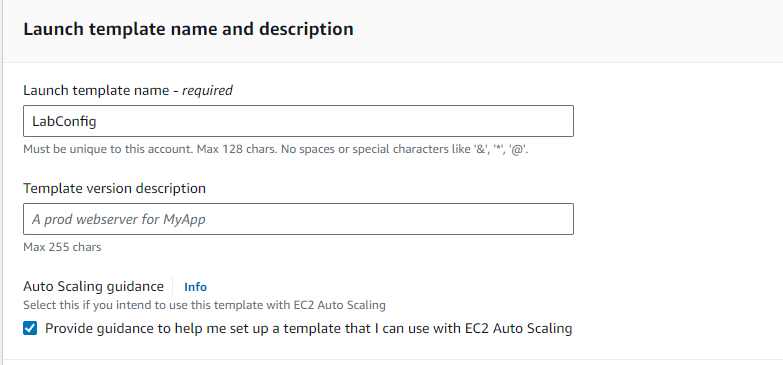


Forward the data from HTTP:80 to the Labgroup EC2 instance. Scroll down and click create load balancer.

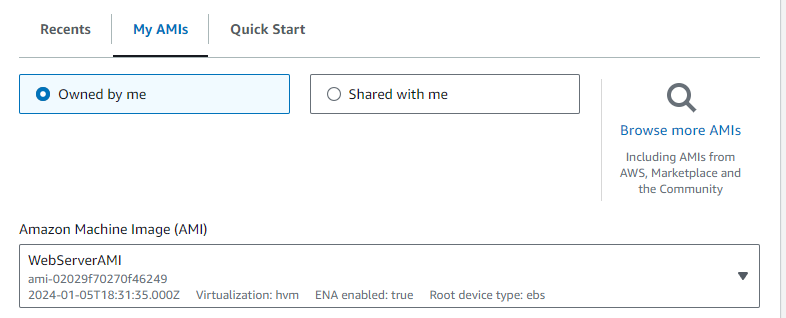
After creating your target group select Load Balancers in the EC2 service. Click create load balancer.



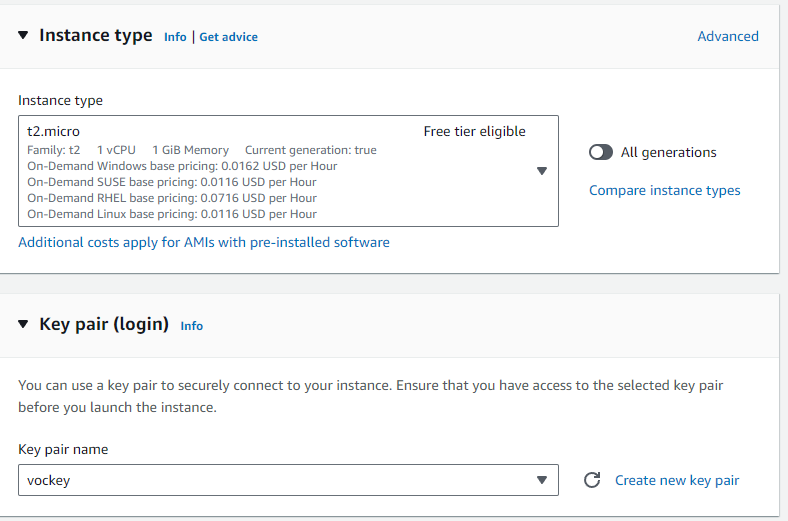
Select launch template and click create launch template once in the dashboard.



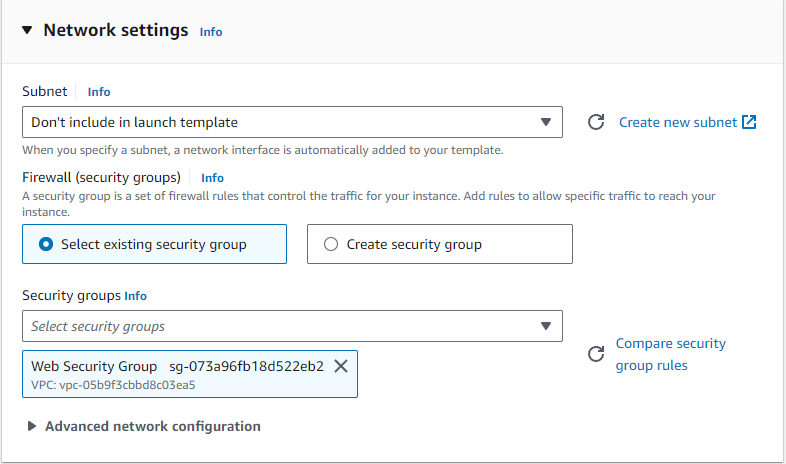
Use the Launch template name of LabConfig. Check the box under Auto Scaling Guidance.



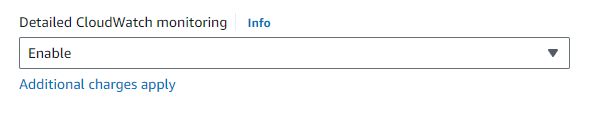
Choose My AMIs and the select the WebServerAMI you configured earlier.



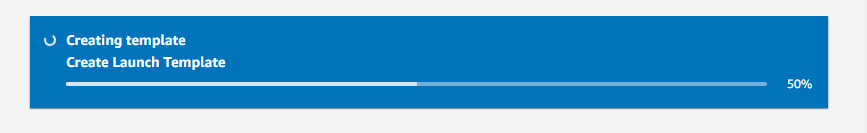
Under Instance type select t2.micro and use the key pair name of vockey.

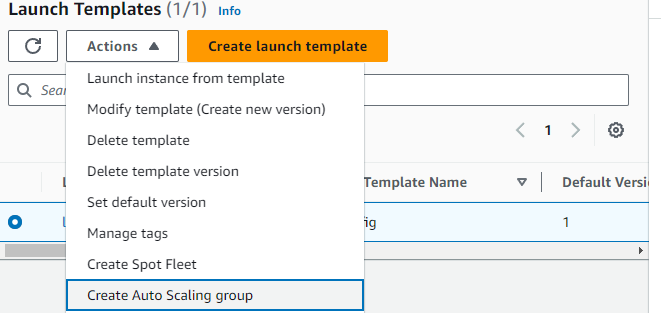


Under Firewall (security groups) choose existing security group. From there them go to security groups and select the Web Security Group you configured earlier. Press the X next to the default security group to delete it.

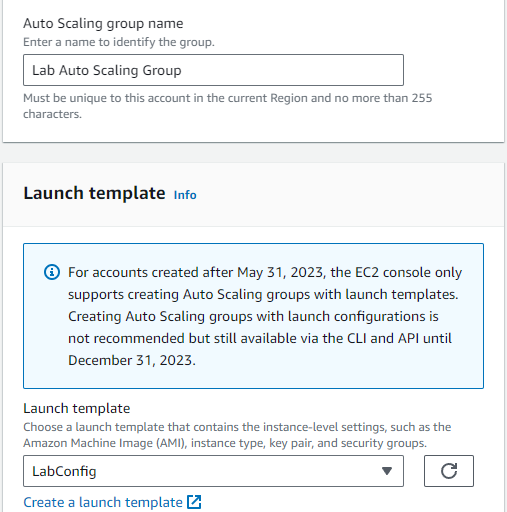


Scroll down to advanced details and expand. Once in there find detailed CloudWatch monitoring and enable. Create launch template.

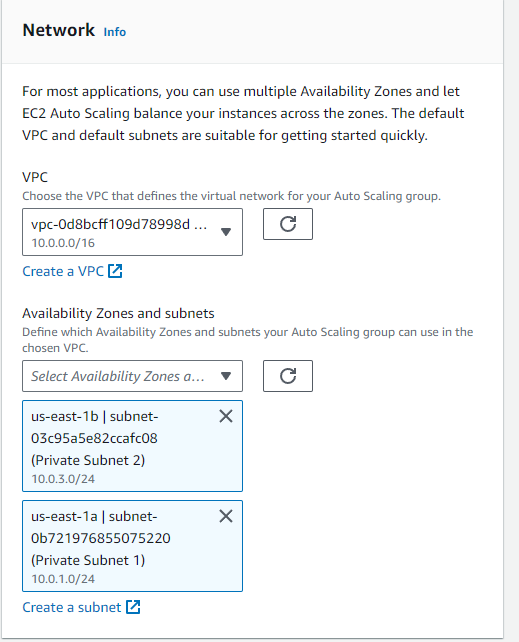




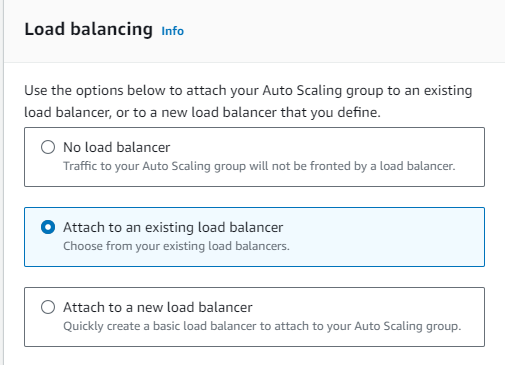
Select your launch template and create auto scaling group for it.

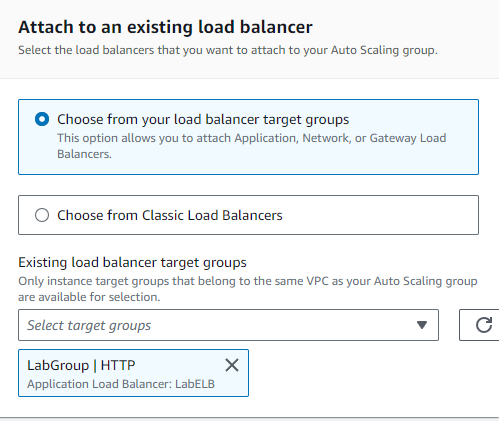


Use the group name of Lab Auto Scaling Group and the launch template of LabConfig. Click next.

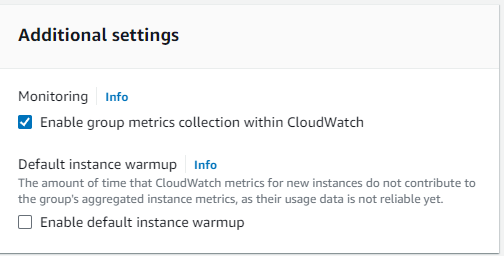


On the next page select your lab VPC as the VPC and select the two availability zones with private subnet ranges. Click Next.

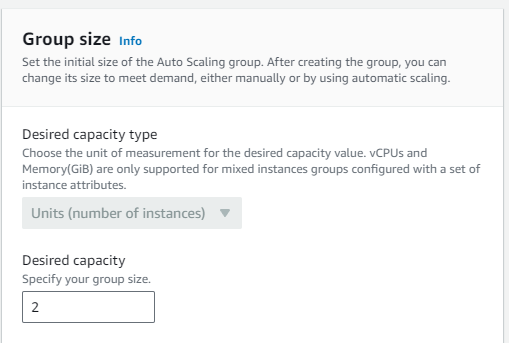


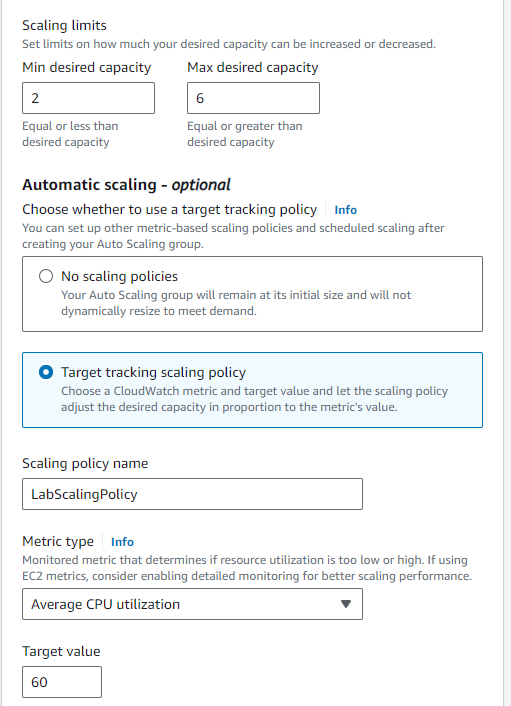


Choose to attach to an existing load balancer to attach your auto scaler to the lab group we configured. Select the target group LabGroup | HTTP. Click next.



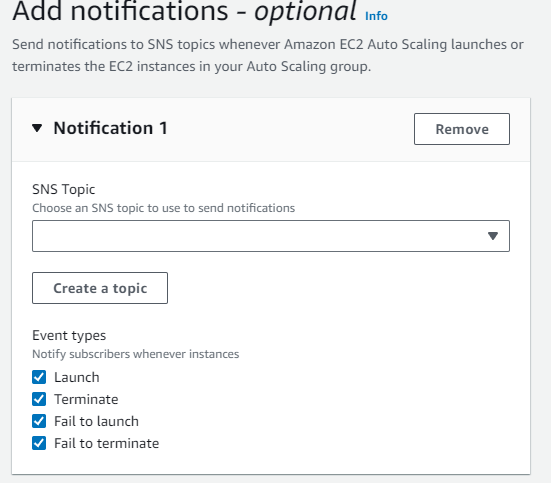
Under additional settings select enable group metrics collection within CloudWatch. This will capture metrics at 1-minute intervals, which allows Auto Scaling to react quickly to changing usage patterns. Choose next.



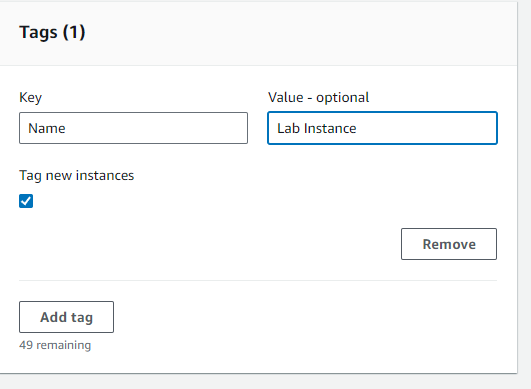


Under **Group size set the desired capacity to** 2, the mi**nimum capacity to 2, and the maximum capacity to 6.** This will allow Auto Scaling to automatically add/remove instances, always keeping between 2 and 6 instances running.

under **Scaling policies**, choose Target tracking scaling give the name LabScalingPolicy, m**etric type of** Average CPU Utilization with a target value of 60. This tells Auto Scaling to maintain an average CPU utilization across all instances at 60% and will adjust EC2 instances accordingly. Choose next.

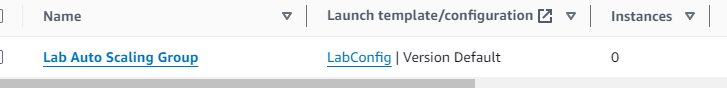


Click add notification and leave as default. Auto Scaling can send a notification when a scaling event takes place. You will use the default settings. Choose next.



Create a tag with the key Name and value Lab Instance. Choose Add tag and next.

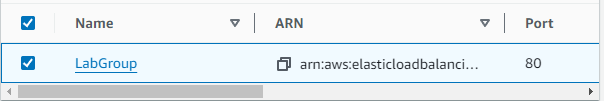
Review your Auto scaling group details and click create auto scaling group. Your Auto Scaling group will initially show an instance count of zero, but new instances will be launched to reach the desire count of 2 instances as we set previously.

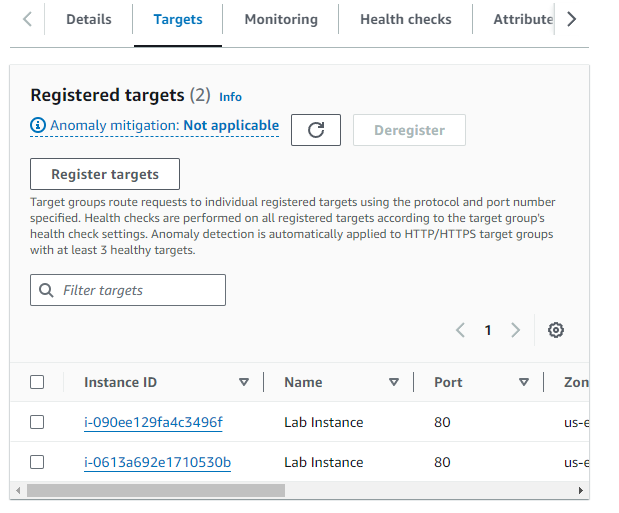
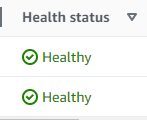


Go to instances to watch if autoscaling occurs.

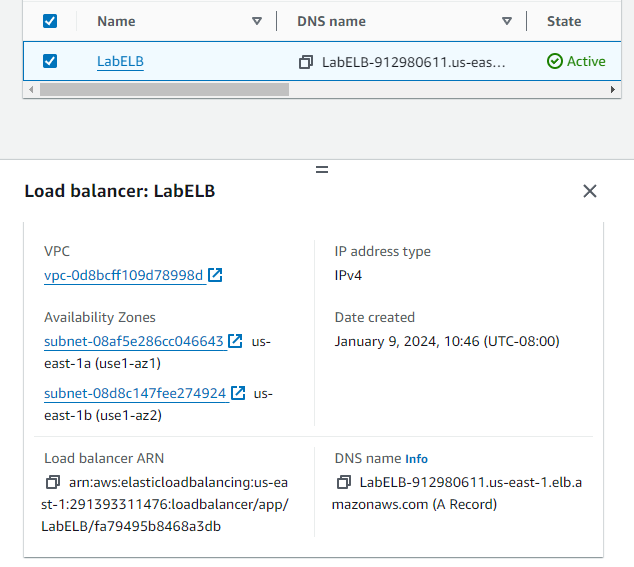


If successful, you will see two new instances both named Lab Instances as we set.



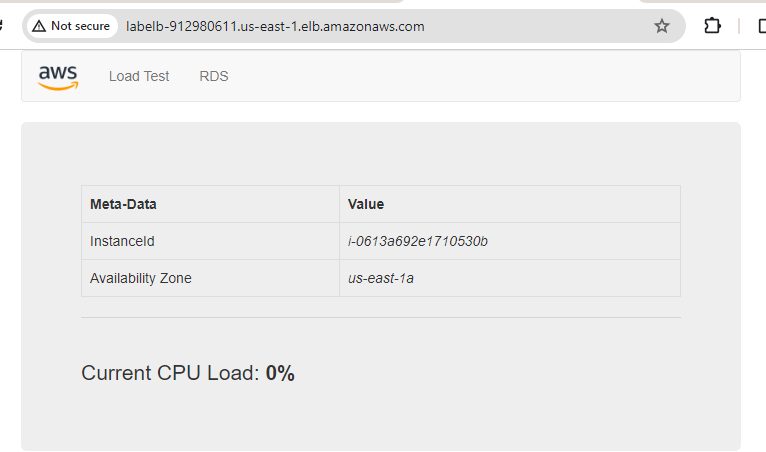
 

Go to target group and select the lab group. Go to targets refresh to see the two new created instance targets. Refresh until they both show healthy.



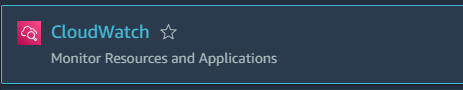
Go to load balancers and copy the DNS name of your LabELB load balancer.

Go to a web browser and enter it.

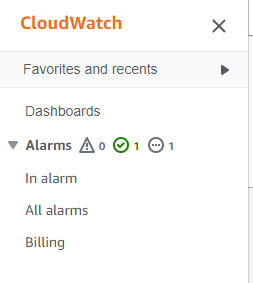


You will see a page like this indicating that the Load Balancer received the request, sent it to one of the EC2 instances, then passed back the result.

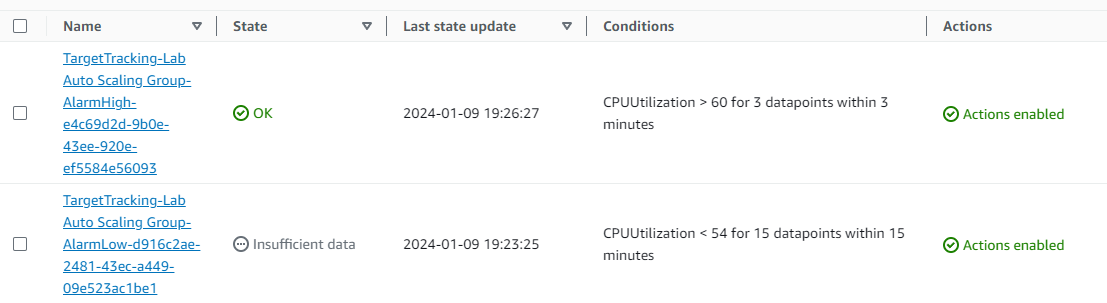
Now we will test the auto scaling.



Search and select CloudWatch from services.



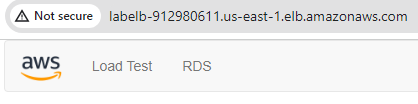
Select all alarms from the left-hand side.



Two alarms will be displayed. These were created automatically by the Auto Scaling group. They will automatically keep the average CPU load close to 60% while also staying within the limitation of having two to six instances.

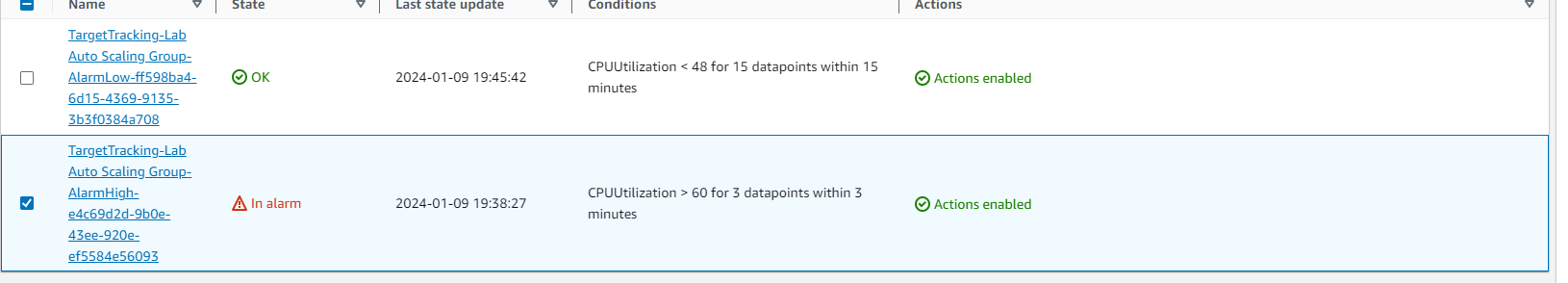
The OK indicates that the alarm has not been triggered. It is the alarm for CPU Utilization > 60, which will add instances when average CPU is high. The chart should show very low levels of CPU at the moment.

We will now put stress onto the load balancer and see how the alarms react.

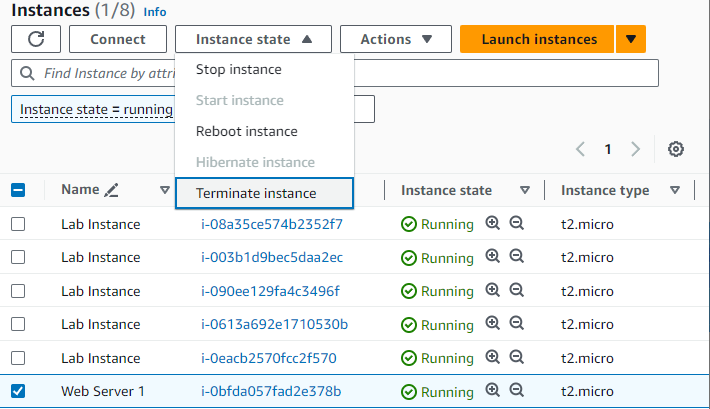


Go back to your webpage and click load test in the top left.

You should see the AlarmHigh chart indicating an increasing CPU percentage. Once it crosses the 60% line for more than 3 minutes, it will trigger Auto Scaling to add additional instances. Once AlarmHigh reaches in alarm you can view the instances it creates with auto scaling.

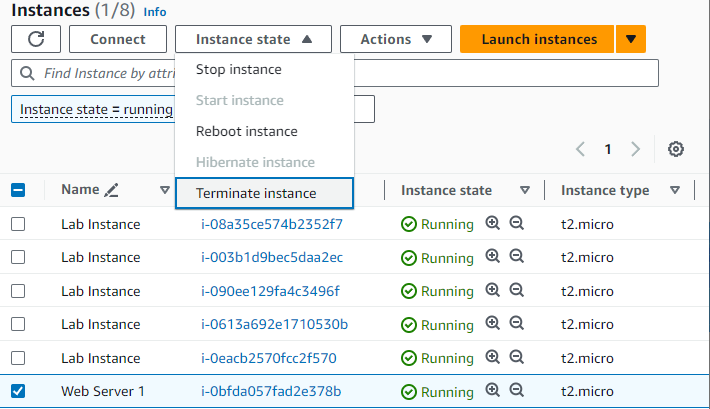


Search up EC2 again and look at your instances.



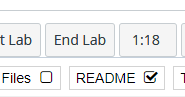


Your auto scaler created more instances to recover from the traffic we forced into it.



in this task, you will terminate Web Server 1. This instance was used to create the AMI used by your Auto Scaling group, but it is no longer needed.

Now click end lab and it’s complete.



**Problems:**

* No problems
  + The lab is step by step so if you followed instructions there was no room for error.
* Thinking the lab failed
  + To start the lab, you must wait 5-10 minutes as it loads so I have closed it accidently.

**Conclusion:**

These were important labs because they showed us more features of AWS and how to configure them. These are important hands-on experiences as we may choose to do something similar in the feature as it is a highly demanded job. We had a focus on automation and databases in these labs with how to configure them as well. The use of SSH and PuTTY brought us back to what we have learned in CCNA made me realize that these are two very closely related industries with overlapping concepts. My most important take away from these labs was the use of automation to assist in the upkeep of your applications or code on the different services. Instead of needing to manually go in and create EC2 instances to accommodate for the increased traffic, you can set parameters for AWS to do it automatically to ensure your application always runs efficiently. Although all concepts in these labs were important, I believe that is the most important takeaway because of the fault tolerance that it adds to the service you provide to the customers with your application etc.