**AWS ACADEMY LABS 4 - 6**

******

***BY: ALEKSIE SANCHEZ MEDEIROS***

******

***Purpose***

To gain an understanding of the basics of AWS by doing labs which focus on Elastic Block Store (EBS), Database Servers, and Elastic Load Balancing on EC2 instances

**Lab 4 - Introduction to EBS:**

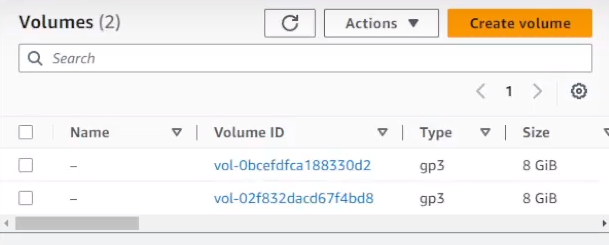
***Background***

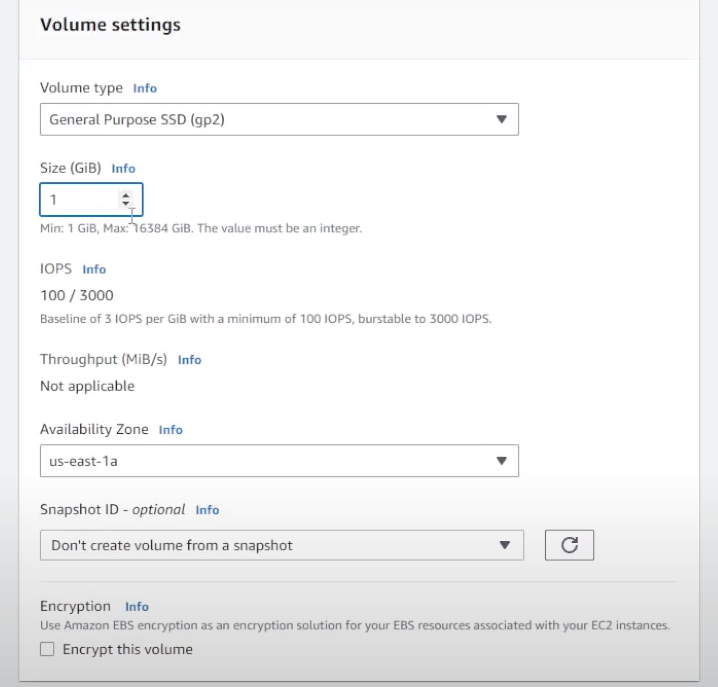
Elastic Block Storage (EBS) is specifically used for storage which can be attached to an EC2 instance to provide a form of impermanent storage. This storage is provided as raw blocks and typically requires the user to give it a filesystem and mount it. In addition, EBS also supports the ability to take snapshots of volume and create clones of existing ones with the snapshot system. However, all this storage is usually lost when the EC2 instance is terminated or stopped.

***Lab summary***

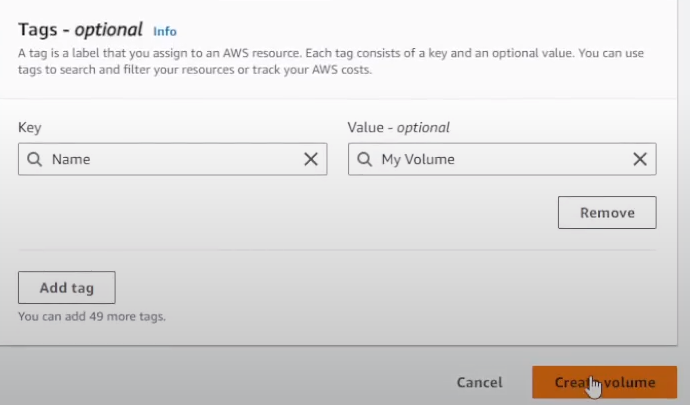
In this lab, we set up a new EBS volume, which we will then mount into an EC2 instance. Using this new volume, we’ll create a file which will be used to determine if we correctly create a backup of the newly created volume using snapshots.

***Step by Steps***

To begin, after starting the lab, enter the EC2 dashboard through the services menu. Then, move to the Instances page using the left navigation bar. Write down what availability zone the EC2 instance named Lab is in. This will be useful later, in my case the instance was in us-east-1a. Then move into the Volumes menu underneath the Elastic Block Store on the left navigation bar. 

There should be two existing instances which have a size of 8gb. We will make a new one that is 1gb in size. To do this, select the orange create volume button.

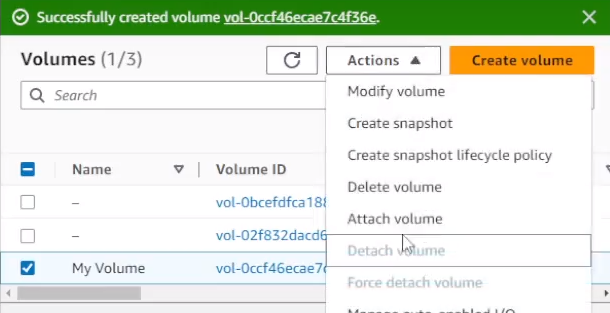
This leads to a page that looks like the one up above. Set the purpose type to gp2 and the size to 1gb ensure that the AZ is set to the same one as the Lab instance.



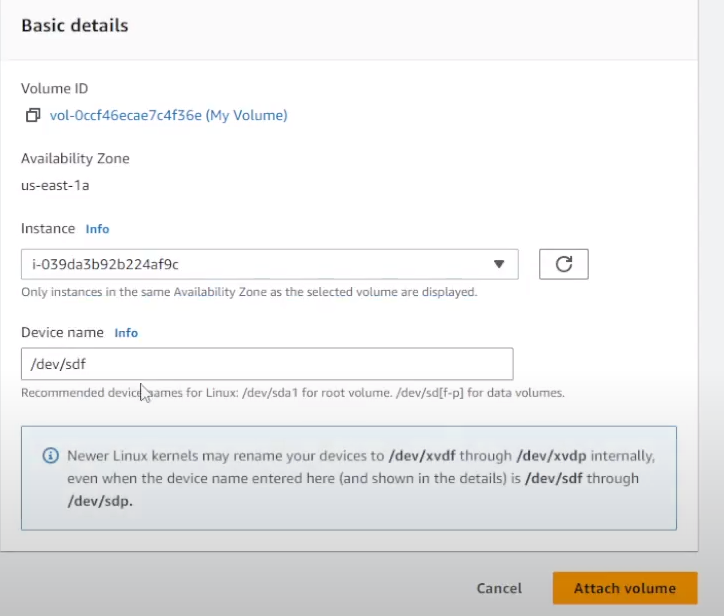
Then near the bottom create a tag with a key of Name with the value of My Volume. Tags are used to differentiate individual resources that may appear similar. Once you hit create volume, you should see a green successful mark and a new volume in the volumes page.

**Attaching the new volume to the instance**

With our new volume we can attach it to an instance by selecting the actions menu and choosing attach instance. It is possible that the volume may not let you attach it. Reloading the volumes list or waiting around 1 minute will usually fix the issue.



Choosing this action will lead you to a menu like the one below.



Set the instance field to the lab instance using the drop down and ensure the device name stays as /dev/sdf. This will be useful to tell volumes apart later. Once done select the orange Attach volume button to attach the volume to the EC2 instance.

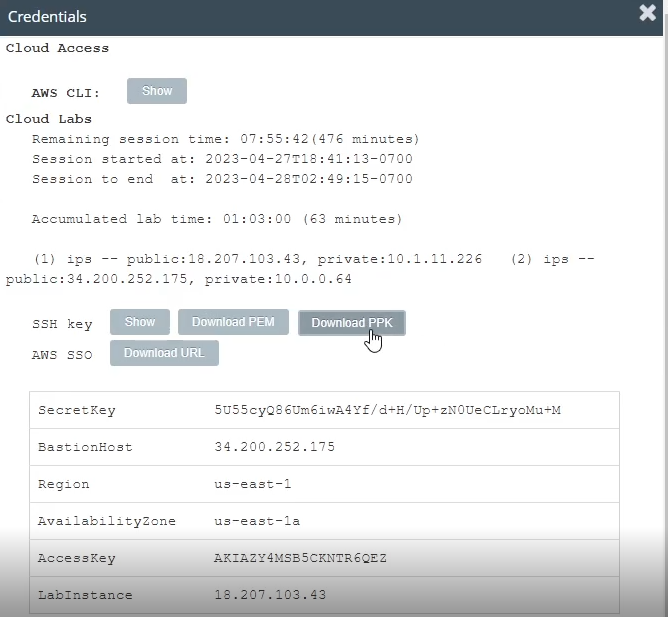
**Connecting to the EC2 instance**

We can also directly manage our EC2 instance, although the lab directions don't tell you that there are two possible ways to connect to the instance. The first is via SSH using PuTTY, which I've only gotten to work sometimes but shows the fact that you can connect to instances from anywhere on the globe.. The other is by using EC2 Instance Connect which has worked without fail. Both methods will accomplish the same goal.

**Using PuTTY (the intended way)**

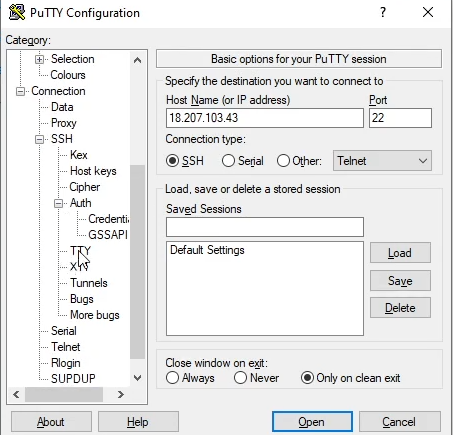
To begin, ensure that you have PuTTY installed on your computer and that there are no firewalls stopping you from using the SSH port [This was the issue].

Then from the webpage where you started the lab press the details button and then download PPK.

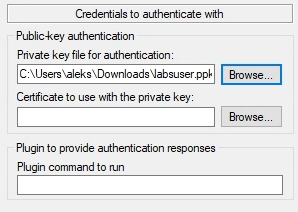


Then head to the Lab EC2 instance and copy the public DNS or IP address from the details heading.

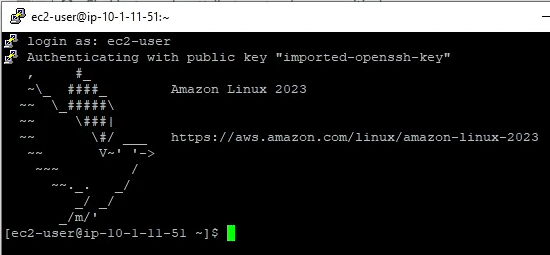


Then head back into PuTTY and copy this address into the Host Name bar. Ensure that SSH is still selected and the port is set to 22.

Then expand the SSH section underneath connections and expand the Auth section and select Credentials. Here is where you will put the PPK file most likely stored in your downloads folder.

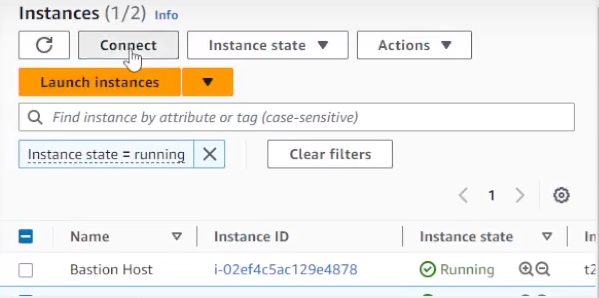
.

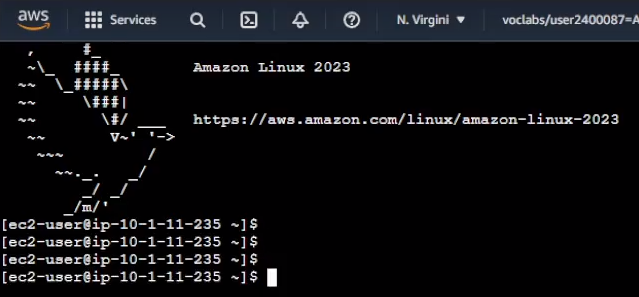
Then hit the Open button near the bottom of the PuTTY window and you should see a screen asking for your login. Here you will insert ec2-user and you should see a page like this one indicating a success.



**EC2 Instance Connect**

If for any reason this doesn’t work and you don’t want to spend an hour or two troubleshooting. You can enter the instance using EC2 Instance Connect. To do this head into the Instance menu and select the Lab instance then choose Connect



This leads to the EC2 instance connect window where you just have to press Connect again which will then lead you into the same window

Although, you’ll feel less dopamine than if you just troubleshooted.

From here you’ll be copying and pasting some Linux commands.

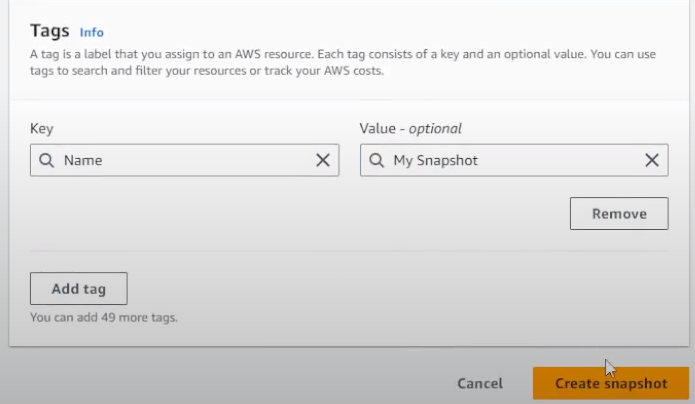
To begin you can check how much storage is available on your instance by using the ***df -h*** command. The current storage is the 8gb storage as you haven’t mounted the new volume yet. This is made clear by the fact that the volume’s filesystem is /dev/xvda1 and it has a size of 8.0G.

First, we create an ext3 file system on our new volume by using the ***sudo mkfs -t ext3 /dev/sdf*** command. Then we’ll create a directory for mounting the volume by using ***sudo mkdir /mnt/data-store***. Then we can mount the volume by using sudo mount **/dev/sdf /mnt/data-store.** We want to make sure the Linux instance will mount our new volume every time the instance is started so we use the command. ***Echo “/dev/sdf /mnt/data-store ext3 defaults.noatime 1 2” | sudo tee -a /etc/fstab*** Next we can ensure that it has been configured correctly by using ***cat /etc/fstab***. We can look at our storage now by using the ***df -h*** command again, this new list should show our 1gb volume called /dev/xvdf at the bottom of the list. As a final check we can add a txt file to the mounted volume using ***sudo sh -c “echo some text has been written > /mnt/data-store/file.txt”***. We can view this file by using ***cat /mnt/data-store/file.txt*** which should display some text has been written.

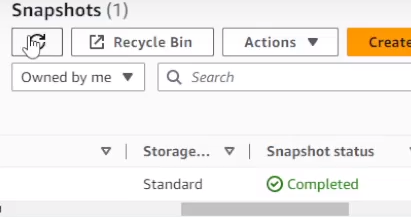
**Creating an EBS Snapshot**

Snapshots are similar to backups of EBS volumes. These can be useful for creating duplicates of volumes if needed and can be easily moved to other regions or distributed by users.

To create a snapshot head to the volumes page from the left navigation bar and select My Volume then from the actions menu select create snapshot. Then similar to what we did for the volume create a tag called Name with the value of My Snapshot



Now let's check on our snapshot by going to the Snapshots menu from the left navigation bar and check on the Snapshot status until it says completed. Note that sometimes AWS will not update the Snapshot status until you reload the menu, don’t spend 10 minutes waiting like I did.

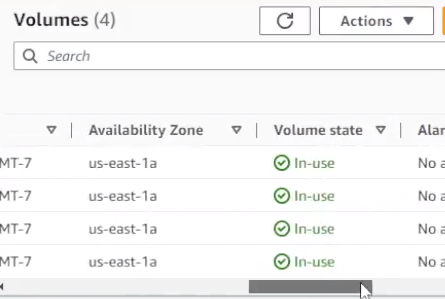


Since the Snapshot saved our volume with the text file we created, let’s remove the txt file to ensure that when we restore the volume using the Snapshot it works properly. To delete the txt file head into the SSH session or EC2 Instance Connect and use the command ***sudo rm /mnt/data-store/file.txt*** then check that the file was deleted by using ***ls /mnt/data-store*** which should just show lost+found



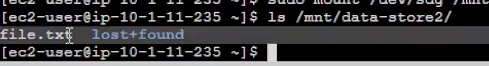
Now let’s restore the volume, go back into the Snapshots menu and select actions, then Create volume from the snapshot. Ensure that the AZ is set correctly and create a tag called Name with the value of Restored Volume.

Heading back into the volumes page you should see a new volume with the name of Restored Volume. Attach this volume to our Lab Instance using the Actions menu like before and selecting the Lab instance. Note the device name which says /dev/sdg. Select the orange Attach Volume button.



The device state should now be in use.

Now we have to mount the volume in EC2 Instance Connect or the PuTTY SSH session using the commands ***sudo mkdir /mnt/data-store2, sudo mount /dev/sdg /mnt/data-store2***. Then let’s check if the snapshot properly saved the file using ***ls /mnt/data-store2***. If every step has been followed correctly you should see a file called file.txt



[We can look at the file using ***cat /mrt/data-store2/file.txt***]

With that you should have set up a new EBS volume, mounted and created a file within the volume to an EC2 Instance and created a snapshot containing the file which was then converted into a volume containing the file.

**Lab 5 - RDS Database:**

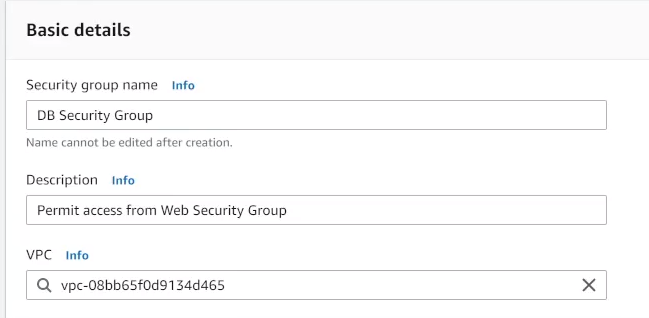
***Background***

AWS RDS is a service which is used to make it easier to scale relational databases, which at its simplest are databases that use tables to organize data, in the cloud as opposed to doing it manually like in traditional architecture. In addition, most of the back end of the process of maintaining these databases.

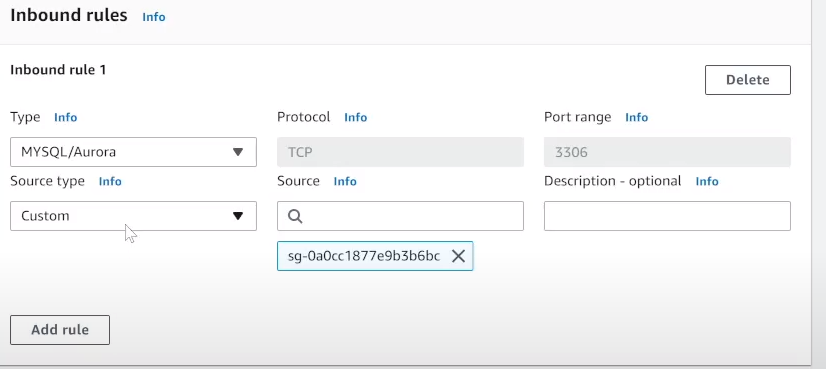
***Lab summary***

In this lab, you create a database using RDS which is connected up to the EC2 instance and allows communication between these two services via VPC.

***Step by Steps***

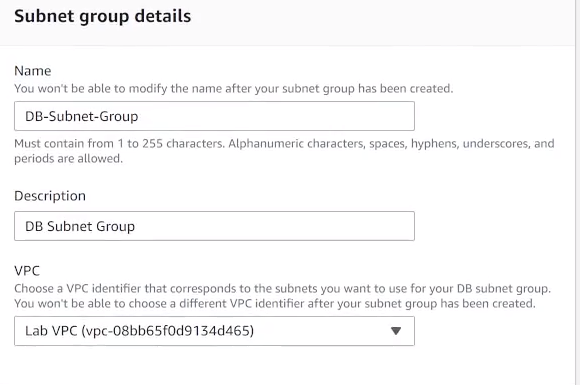
To start off, enter the VPC service where we will be creating a security group which will be used to ensure that our web server will access the DB instance. To do this, head into the Security Groups menu and select the orange Create security group button. Here we set the security group name to DB Security Group and the description to Permit access from Web Security Group and set the VPC to the already existing Lab VPC

Now we’ll set an inbound rule with a type of MYSQL/Aurora and set the source to the existing Web Security Group.

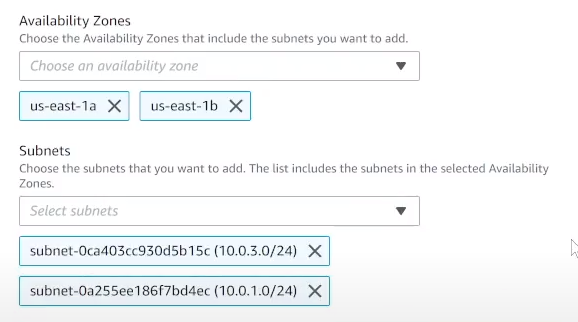


Now create the security group by selecting the orange button at the bottom.

**Creating a subnet group for the DB**

To create the new subnet group switch to the RDS service in the services menu and use the left nav bar to move to the Subnet groups menu. Set the name to DB-Subnet-Group and the description to DB Subnet Group using the Lab VPC.

Then to add the subnets choose the availability zones us-east-1a and us-east-1b and select the subnets 10.0.1.0/24 and 10.0.3.0/24

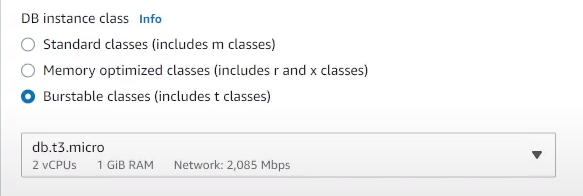


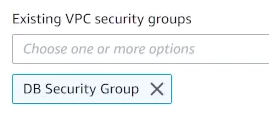
With that done select Create to finish setting up the DB subnet group.

**Creating the DB instance**

To begin the creation use the left nav pane to head into Databases and select create database. Keep the standard create selected and set the engine option to MySQL. For the template choose Dev/Test and choose the Multi-AZ DB instance under the Availability and Durability menu.

Under settings, set the DB identifier to lab-db the master username to main and the master password to lab-password.

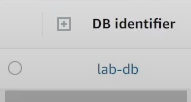
For the instance configuration choose burstable classes and select db.t3.micro

Set the storage type to general purpose SSD (gp3) and the allocated storage to 20gb. Set the VPC to the Lab VPC in Connectivity and choose the existing VPC security group whilst also unselecting the default one.

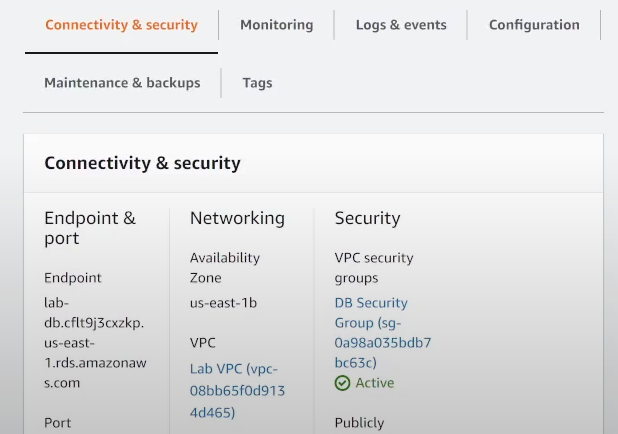
Finally, in additional configuration set the initial database name to lab and disable the following settings:

* Enable automatic backups
* Enable encryption
* Enable enhanced monitoring

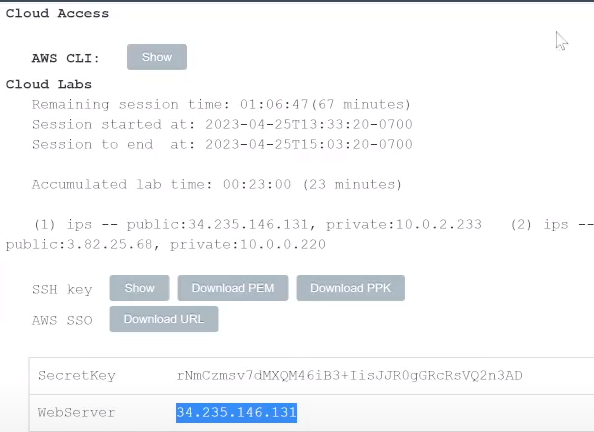
It’s not normally recommended to this but for the sake of this lab and for faster database deployment please do so. Then create the database.



Now you should see a lab-db in the Databases menu, select the hyperlink and wait until the Status changes to Modifying or Available. This may take around 7-8 minutes or longer. While you wait copy the endpoint value for later under connectivity and security.

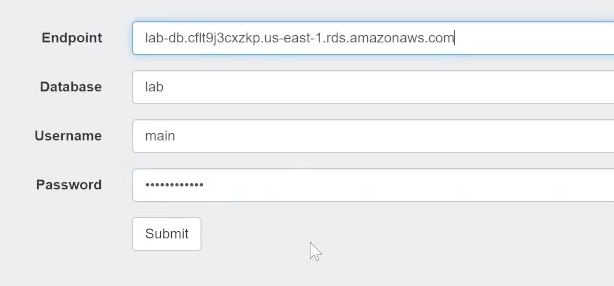


**Interacting with the DB**

With the DB and all its necessary components configured, head into the details menu of the lab instructions and copy the WebServer IP.

Enter this into a new tab to view information on the Web Server EC2 instance.

Select the RDS tab to the right of the AWS logo and copy in the endpoint and lab in the database field, main in the username field and the password lab-password



This will lead you to a pregenerated Address book which is using our newly created database to store the information. You can edit and fiddle around with it to change the names.



With that you have successfully created a database using RDS which is linked up to the EC2 instance and allows communication between these two services via VPC.

**Lab 6 - Creating instances with Elastic Load Balancing:**

***Background***

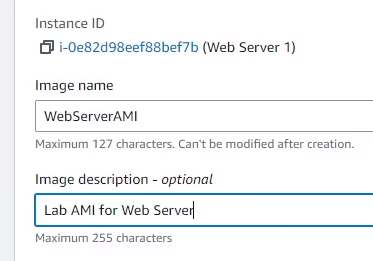
The Elastic Load Balancing service offered by AWS is used to distribute traffic to different EC2 instances. This is typically done to avoid high CPU loads. This can be mixed with the Auto Scaling feature to create new AWS instances which through the Load Balancer will receive equal traffic. In addition, the auto scaling feature will create new EC2 instances whenever CPU usage reaches a user set threshold.

***Lab summary***

In the final lab, we will create an AMI from an existing EC2 instance which will be used in conjunction with a Load Balancer and an Auto Scaler to ensure that we can create new EC2 instances without needing to lift a finger whenever CPU usage is high to ensure that anyone attempting to access our ‘service’ still has high availability.

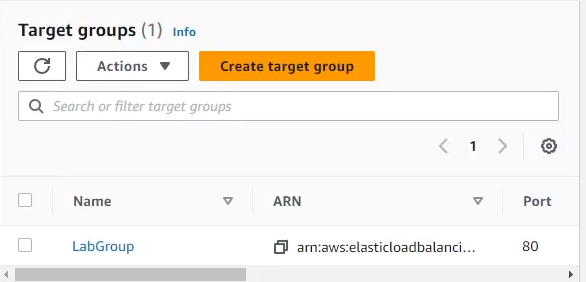
***Step by Steps***

To start off, enter the EC2 Instances menu where you should see an already running instance with the title of Web Server. Ensure that the status checks for this web server have passed. We’re going to be creating an Amazon Machine Image (AMI) based on this instance. To do that, select Web Server 1 and select Images and Templates > Create Image.

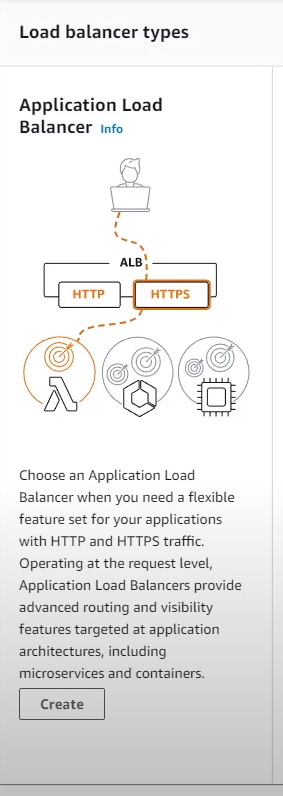


Here we’re going to add an Image name and description as shown above, then hit the orange Create Image button in the bottom right corner.

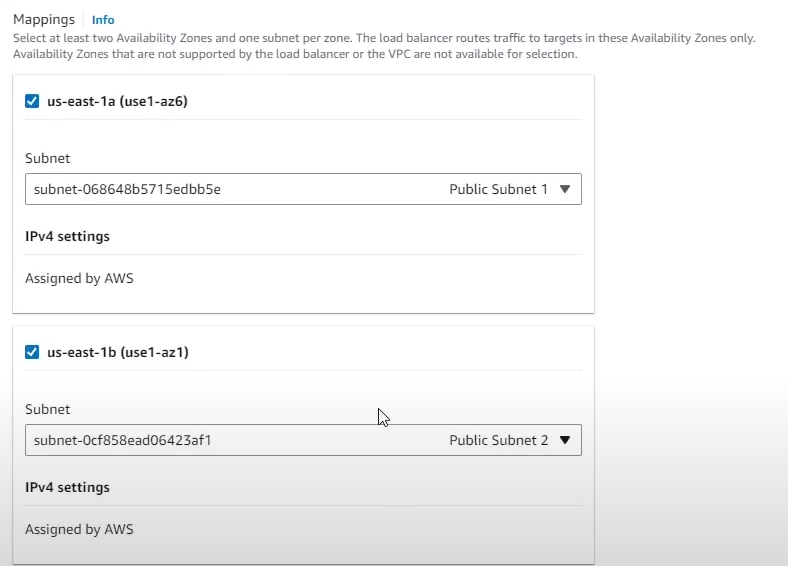
**Making the Load Balancer**

Now we need to make sure that we have a system in place which will balance the traffic across not only the EC2 instances but also the AZs. To do this head into the Target Groups menu from the left navigation bar. Select the Create target group button. Select target type Instances and set the Target group name to LabGroup. Then select the Lab VPC in the VPC dropdown. On the next page select Create Target group. Target groups are normally useful for directing traffic in a Load Balancer, such as when you want traffic coming from a certain device to go to a different server. In the case of this lab, we will only have one Target Group for our ‘web application’. 

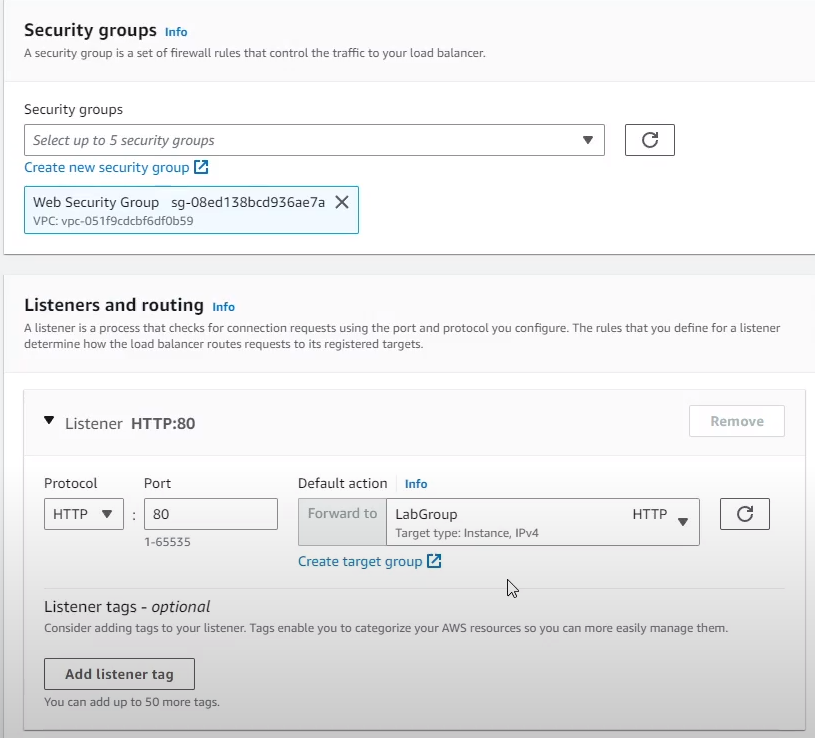
With the target group completed head to the Load Balancers menu from the left bar and choose Create Load Balancer.



Here, choose create under the Application Load Balancer option. Set the name to LabELB and in the network mapping section choose Lab VPC.

To map the zones for the first AZ set Public Subnet 1 and for the second choose Public Subnet 2

For security groups choose the Web Security Group and remove the default from the dropdown menu.

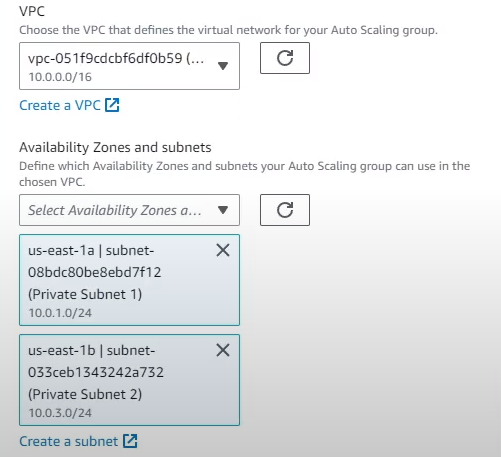


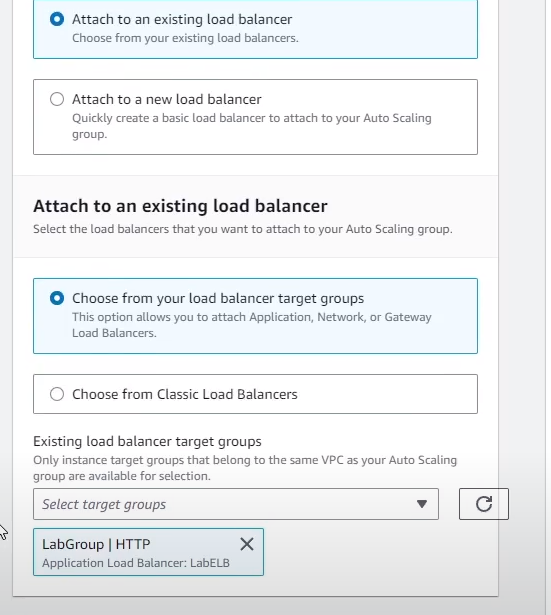
Then set the HTTP:80 listener to LabGroup. With all these settings complete, choose create load balancer and view load balancer after that.

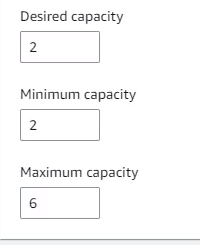
**Autoscaling Instances**

Before we can begin automating the process of launching instances we need to create a launch configuration. To begin, go to the Launch Configurations at the bottom of the left navigation bar underneath the Auto Scaling header. Then choose the orange Create Launch Configuration button. Set the configuration name to LabConfig and the AMI to the Web Server AMI from earlier. Our new instance type will all be t3.micro unless your region is set to us-east-1, in which case you will use the t2.micro instance type. Inside the additional configuration submenu check the Monitoring box which enables EC2 instance detailed monitoring. For the Security Group, select the existing security group called Web Security Group. For the key pair choose the vockey existing key pair and create launch configuration.

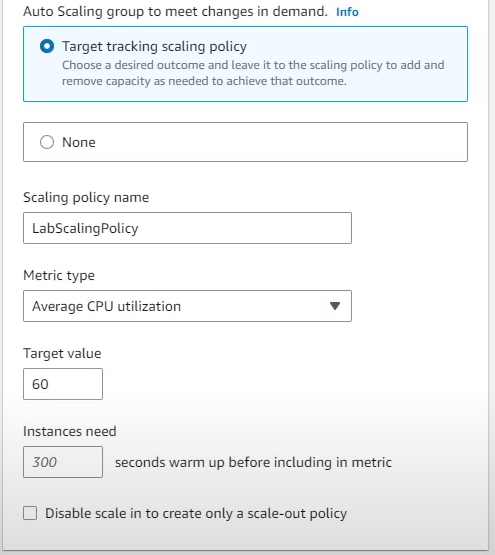
Now create an Auto Scaling group by selecting the launch configuration you created and selecting Create Auto Scaling group in the actions menu.

Set the name to Lab Auto Scaling Group and select next. Select the Lab VPC and choose Private Subnet 1 and 2 from the drop down

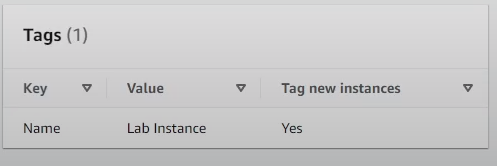
In the next page choose attach to an existing load balancer and select the LabGroup load balancer in the drop down menu. In additional settings enable group metrics collection and move on to the next page.

Here we will set the maximum and minimum scaling we want for our instances for this lab we will set the desired to 2, the minimum to 2 and the maximum to 6

In the scaling policies menu choose the Target tracking scaling policy and change the name LabScalingPolicy which uses the Avg CPU utilization metric type with a target value of 60. This tells the auto scaler that the average CPU usage across all the instances should be maintained at 60 whether that requires creating more or having less instances.

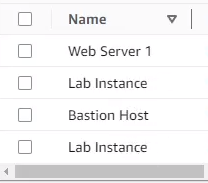


Move onto the next page. Keep the default settings for that page and move onto the next one. At the bottom before the orange Create Auto Scaling group button create a new tag with a Key of Name and a Value of Lab Instance.

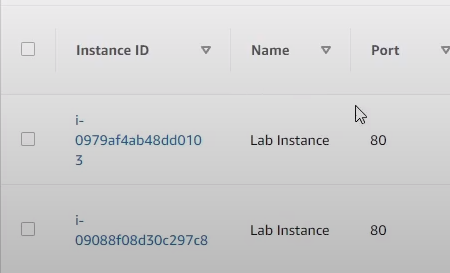


With all these settings select the orange button which should create the Auto Scaling group with no errors.

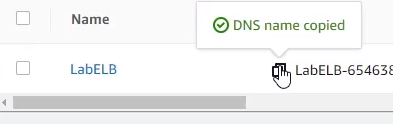
**Verifying proper setup**

Head back into the Instances menu and you should see two additional instances titled Lab Instance which have been created by the Auto Scaling

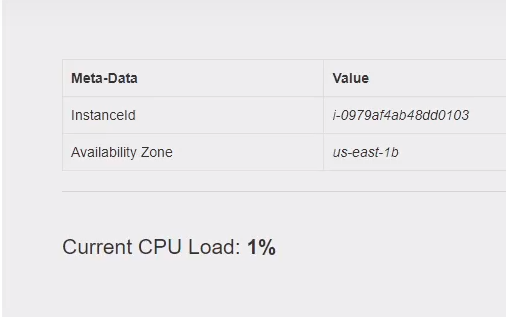
Head back into the Target Groups menu using the left bar and select the LabGroup and move to the Targets tab. There should be two Lab Instances here.



Wait until their health status shows up as healthy which only took around 10-15 seconds and move to the Load Balancers menu.

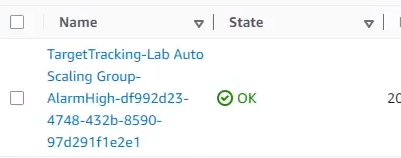


Copy the DNS name of the LabELB and open it in a new tab.



You should see something similar to the screen above. The fact that you can make it onto this page means that the Load Balancer saw your HTTP request and sent it to one of the EC2 instances to load the web page.

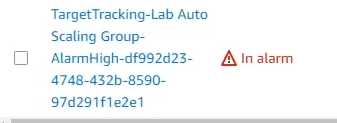
Head back into the AWS dashboard and go to the CloudWatch service via the Services menu. In the left navigation bar open up Alarms and choose All alarms.

Wait around 1-2 minutes reloading the menu every once in a while until the alarm which has AlarmHigh in the name has the OK state.

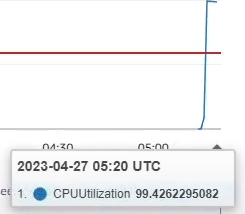
If the alarm is set to OK this means that the CPU usage is under 60. When the CPU usage reaches above 60, new EC2 instances will be created.

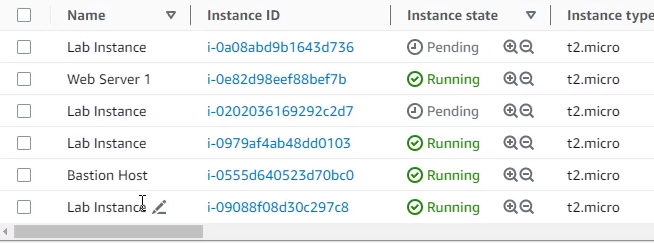
Heading back to the load balancing web page, we can stress test our Auto scaler by selecting the Load Test button next to the AWS logo.

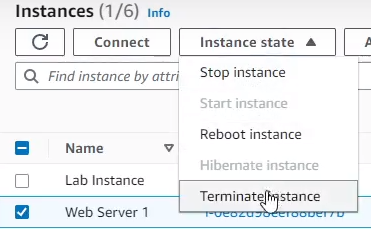
This will set the AlarmLow to OK and the AlarmHigh status to In Alarm in around 5 minutes.



We can also look the CPU utilization graph to ensure that our alarm is set correctly.



Now if we had back into the EC2 service and into the Instance menu you should see more newly created Lab Instances made to lower the average CPU usage.

We also no longer need the Web Server 1 instance since it was only used for creating the AMI for the Auto Scaling group. We can terminate it by selecting Web Server 1 and pressing Instance State > Terminate Instance

With that you have successfully created a Target group, a load balancer and an auto scaler all of which are necessary components in cloud computing to ensure that there is high availability across the world to your service.