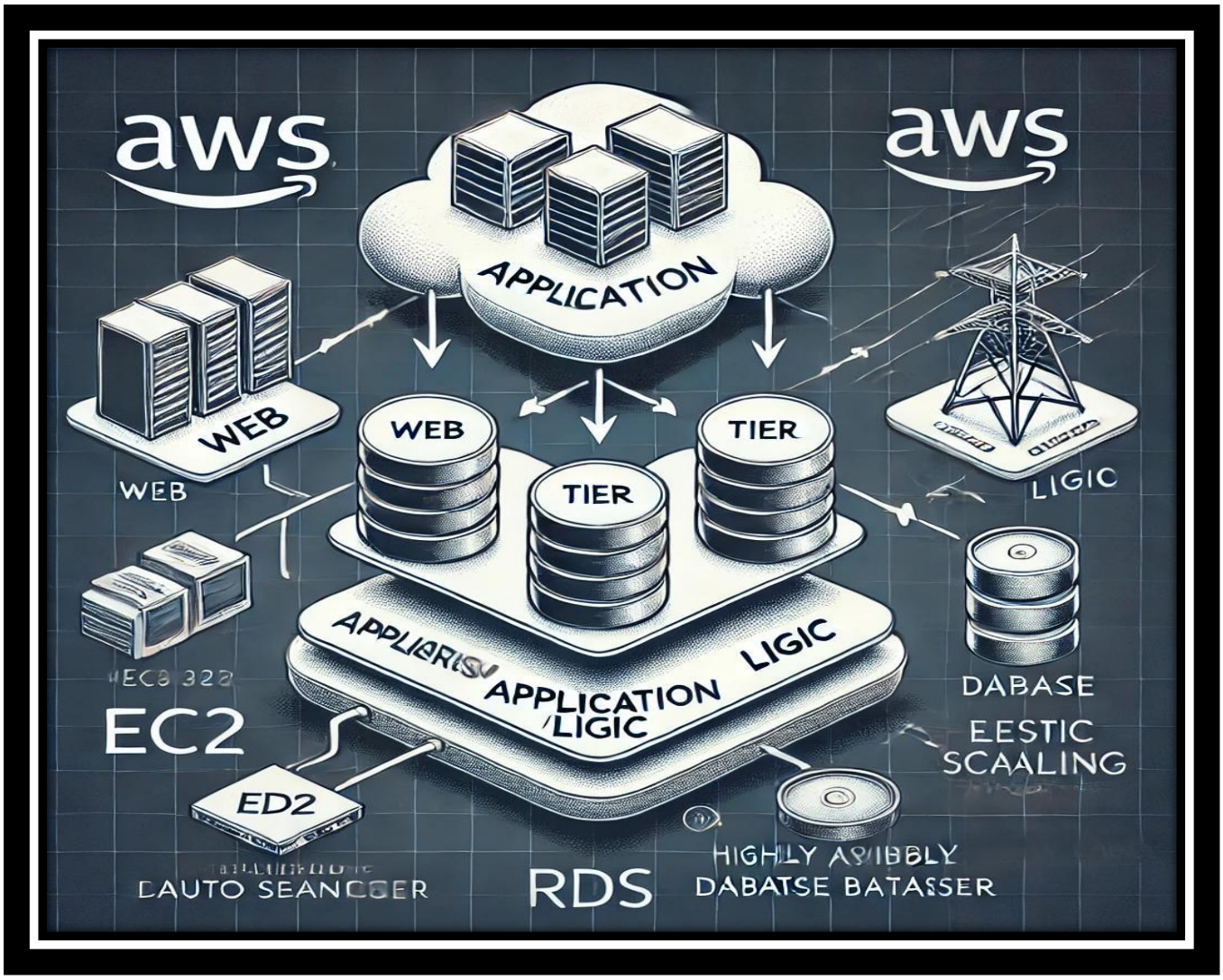
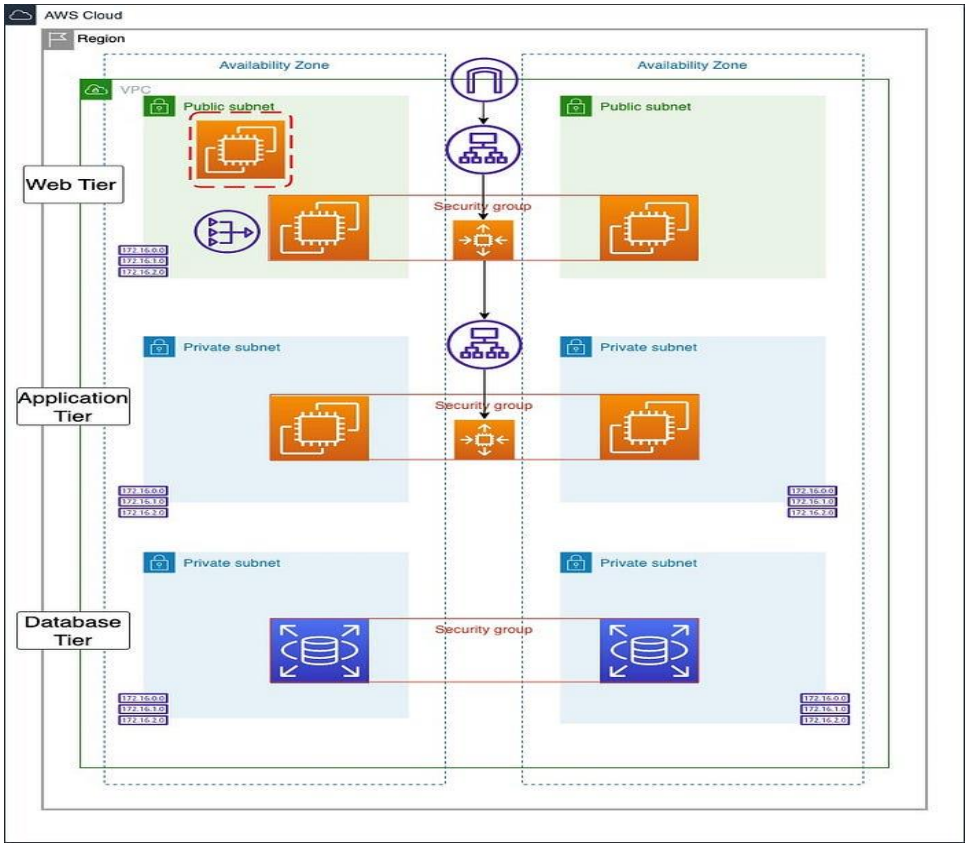
Creating a Three-Tier Architecture for a Café



This guide walks through setting up a three-tier architecture on Amazon Web Services (AWS), a common approach for designing scalable, reliable, and secure applications. This model separates an application into three logical or physical layers:

- **Presentation Tier:** The user interface for interacting with the application, typically accessed on personal devices via web browsers using HTML, CSS, and JavaScript.
- **Application Tier:** This layer processes logic and handles communication between the user interface and the database. It executes the core application functions and is developed using programming languages like Python, Java, or PHP.
- **Database Tier:** This layer stores and manages data, using a database management system for retrieval and processing.



Benefits of a Three-Tier Architecture

1. **Scalability:** Each tier can be scaled independently to manage varying workloads.
2. **Reliability:** Isolating tiers enhances system stability; issues in one tier won't directly impact the others.
3. **Security:** By separating tiers onto different servers, sensitive data and operations are more secure.
4. **Cost Efficiency:** Resources can be allocated based on the needs of each tier, optimizing costs.

Prerequisites

- An active AWS account.
- Command Line Interface (CLI) access.

Use Case:
As a cloud engineer for CLOUD_CREATE Inc., you are tasked with designing a high-availability, three-tier architecture for a web application. This architecture will handle customer requests, process application logic, and store product and customer information in the database.

Step 1: Build our VPC

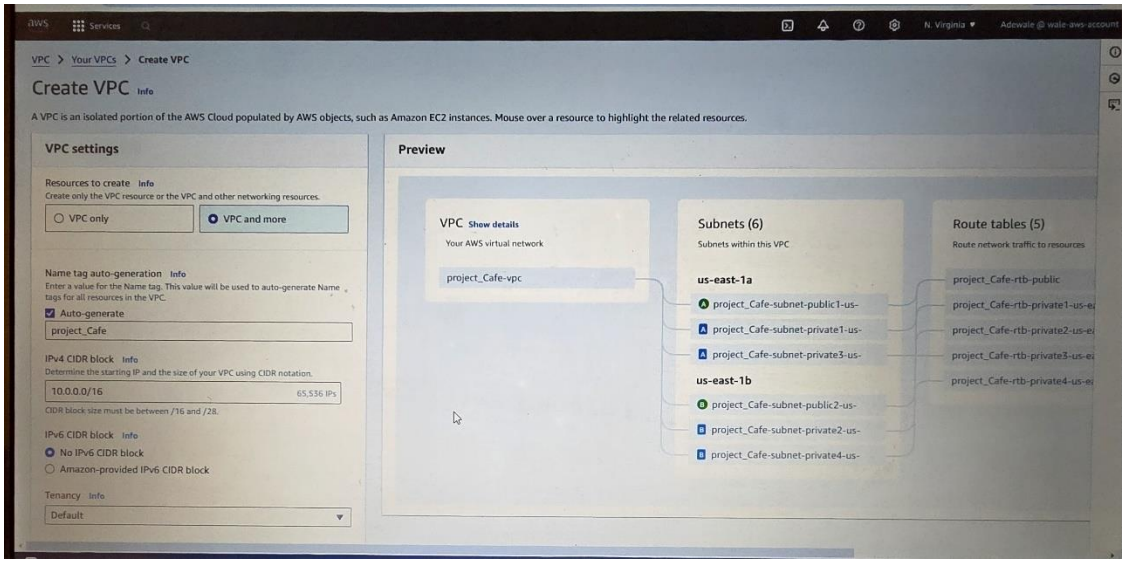
To begin, configure a Virtual Private Cloud (VPC) and its networking components:

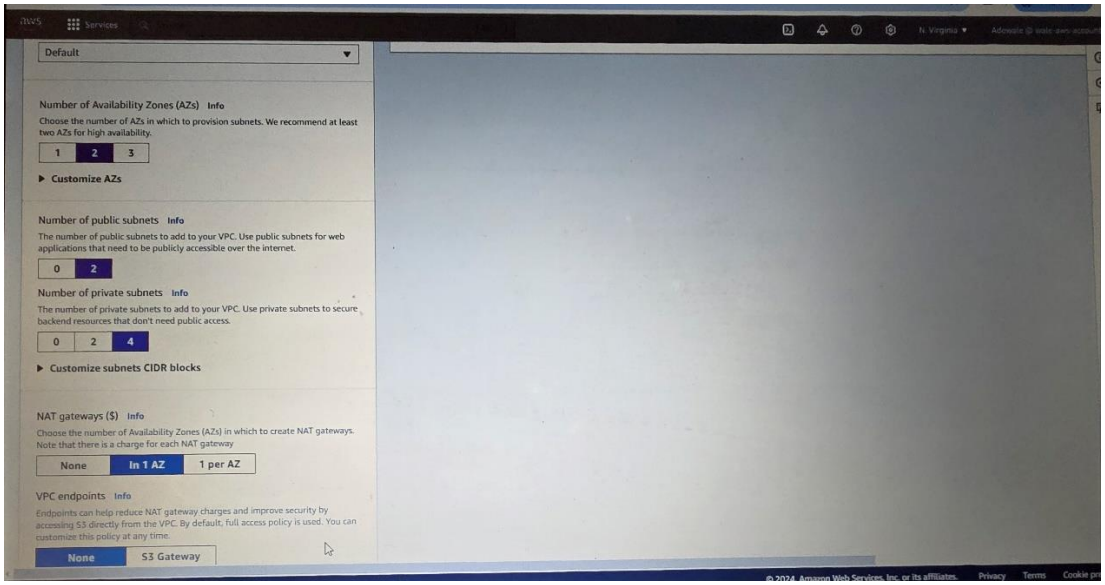
1. Navigate to the VPC Dashboard in AWS and create a VPC using the "VPC and more" option. Assign the IPv4 CIDR block 10.0.0.0/16.
2. Verify that six subnets (three public and three private) are created.
3. Enable automatic public IPv4 addressing for the public subnets to allow external access.

HOW?

Configure VPC and other network services

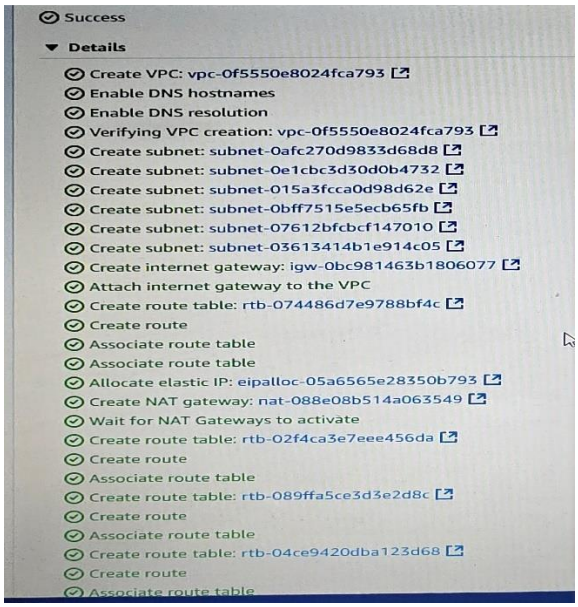
Navigate to the VPC dashboard and click "**Create VPC**". Select the "**VPC and more**" option to configure your VPC, subnets and other networking services at the same time. Name your VPC, then add the IPv4 CIDR block. We will be using **10.0.0.0/16.**

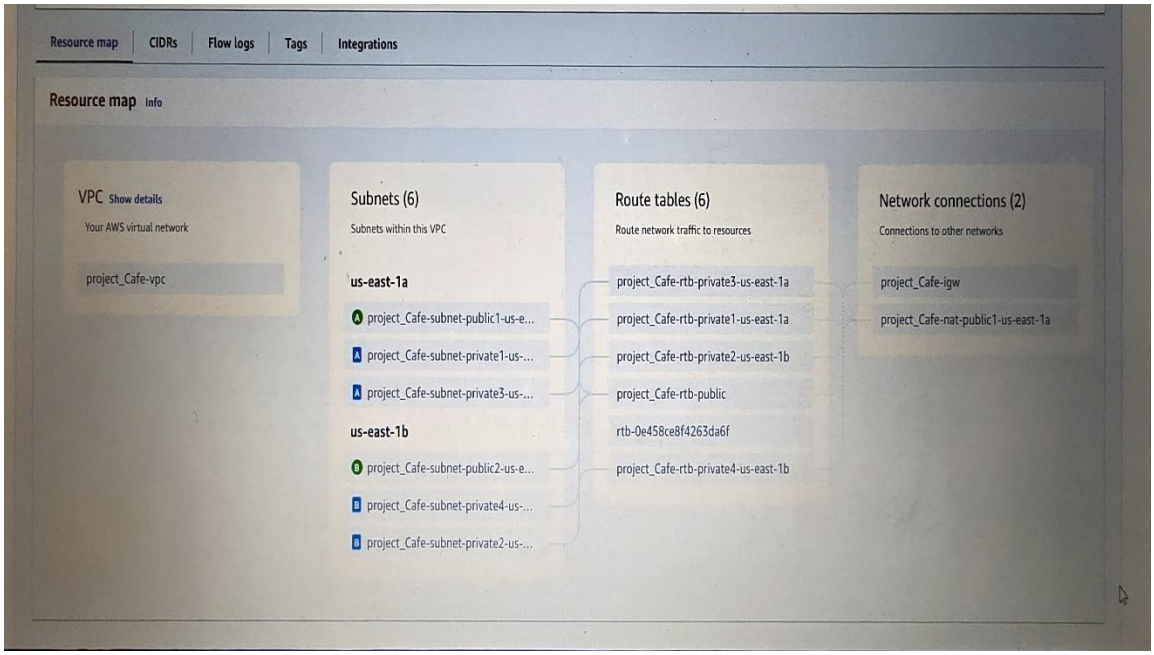




“Create VPC”, then wait until all the resources are created, as shown below.

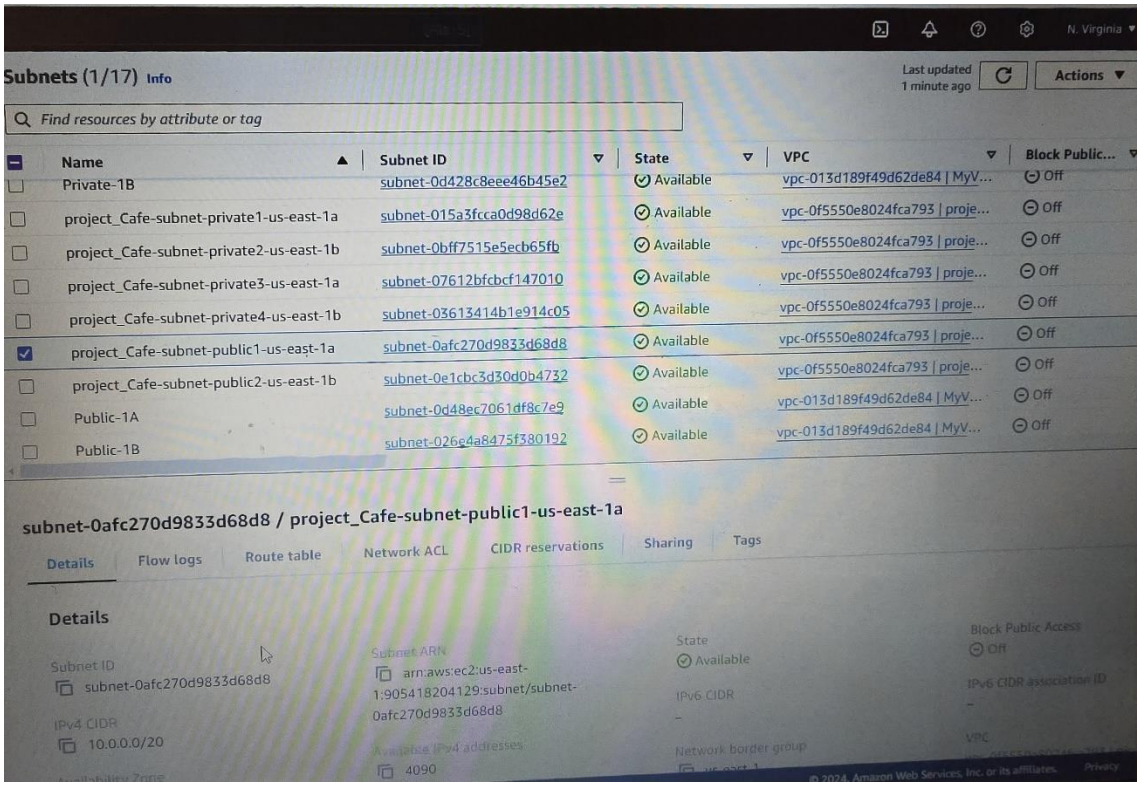
Notice that many network resources are created to configure correctly our VPC according to our specific configurations. It’s fortunate that AWS provides us with this feature, but we can also build most of these resources individually by our ourselves.



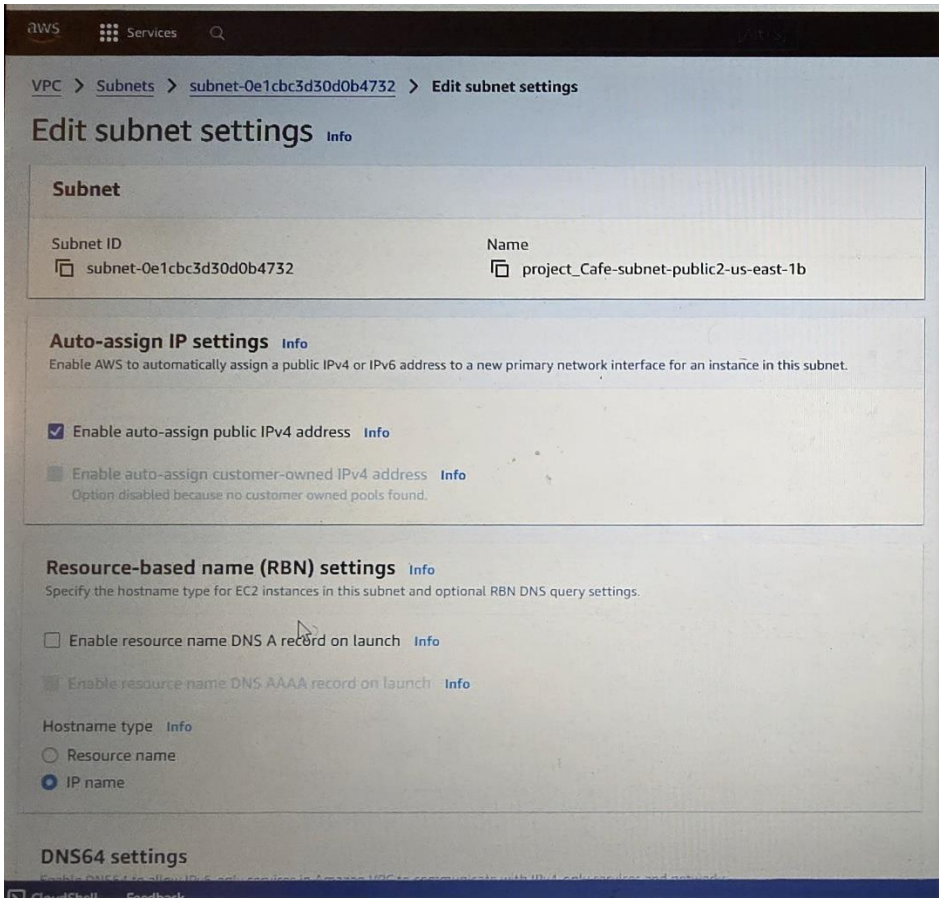


Navigate to “**Subnets**” in the left pane of the VPC dashboard and verify that all 6 subnets have been created. We now need to configure each public subnet one at a time to auto-assign public IPv4 address so they can be publicly accessed through the internet.

To configure this, select one of the public subnets, click “**Actions**” on the top right, then click “**Edit subnet settings**”.



Select “**Enable auto-assign public IPV4 address**”, the click “**Save**”.



Remember to make sure you repeat the process again for the second public subnet.

Now that we've configured our VPC, subnets and other network services, we can proceed to **Step 2: Building out Web Tier!**

Step 2: Build the Web Tier

Set up the frontend layer:

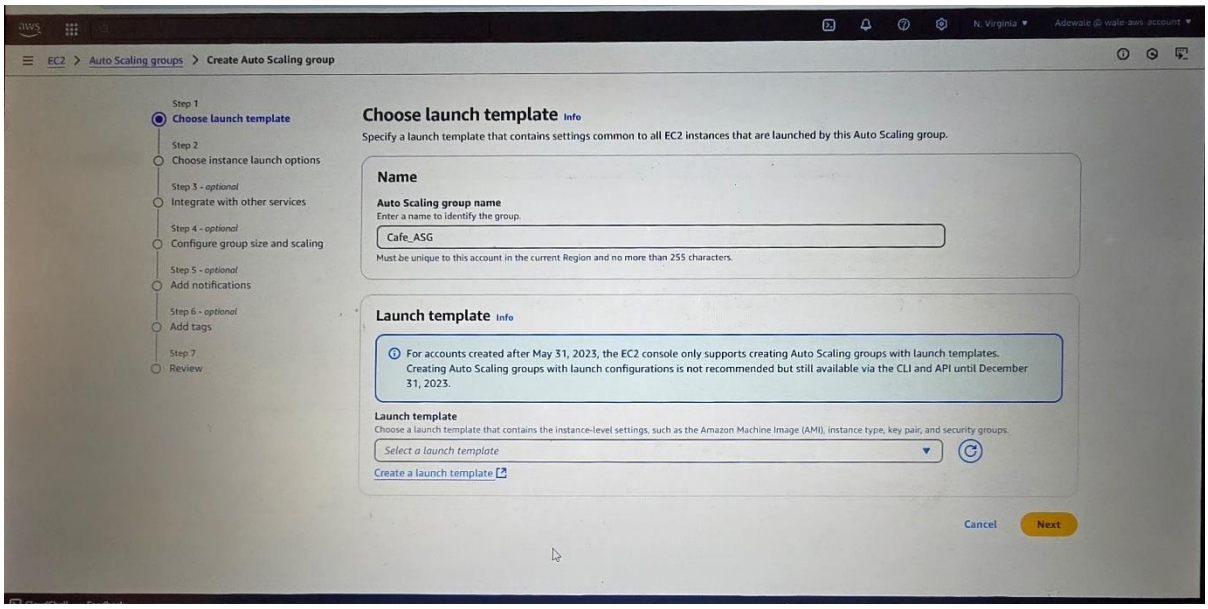
1. Launch EC2 instances in an Auto Scaling Group (ASG). Configure an Application Load Balancer (ALB) to distribute incoming traffic across instances.
2. Define inbound rules in the Security Group to allow HTTP, HTTPS, and SSH access.
3. Use a launch template to automate the configuration and include a script to set up the web server.

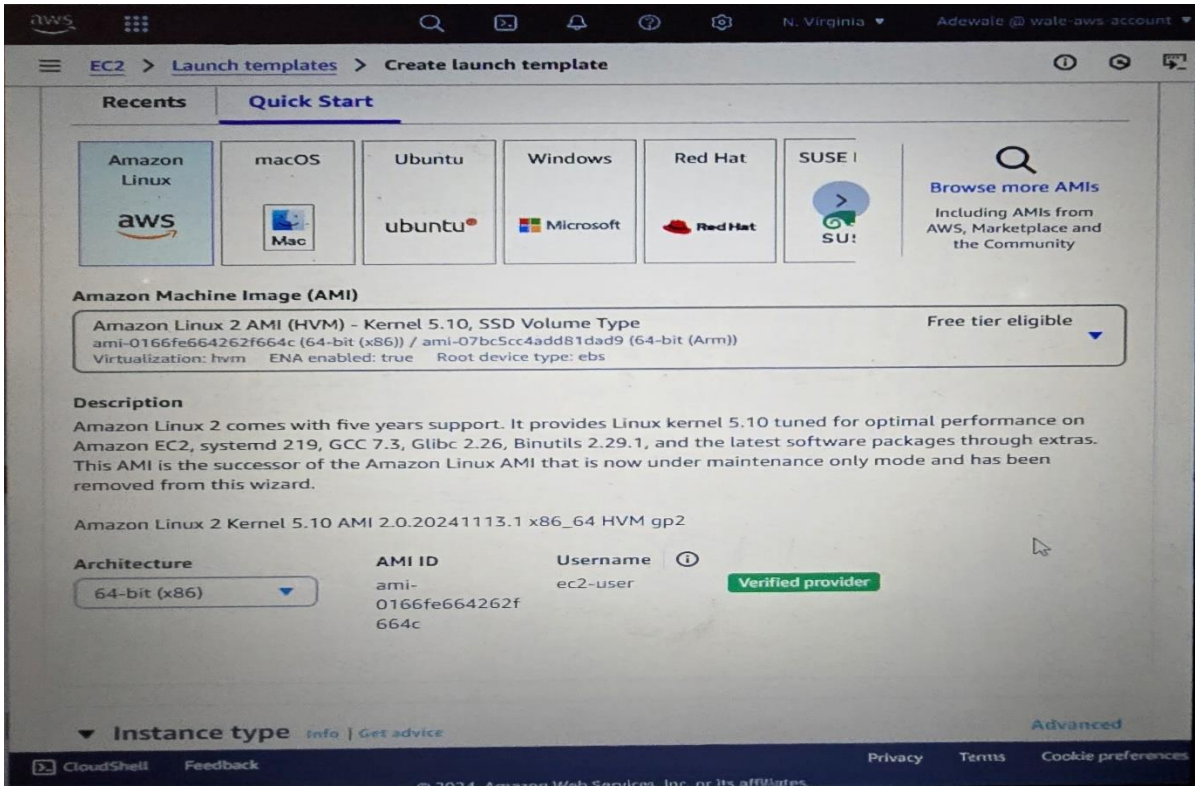
HOW?

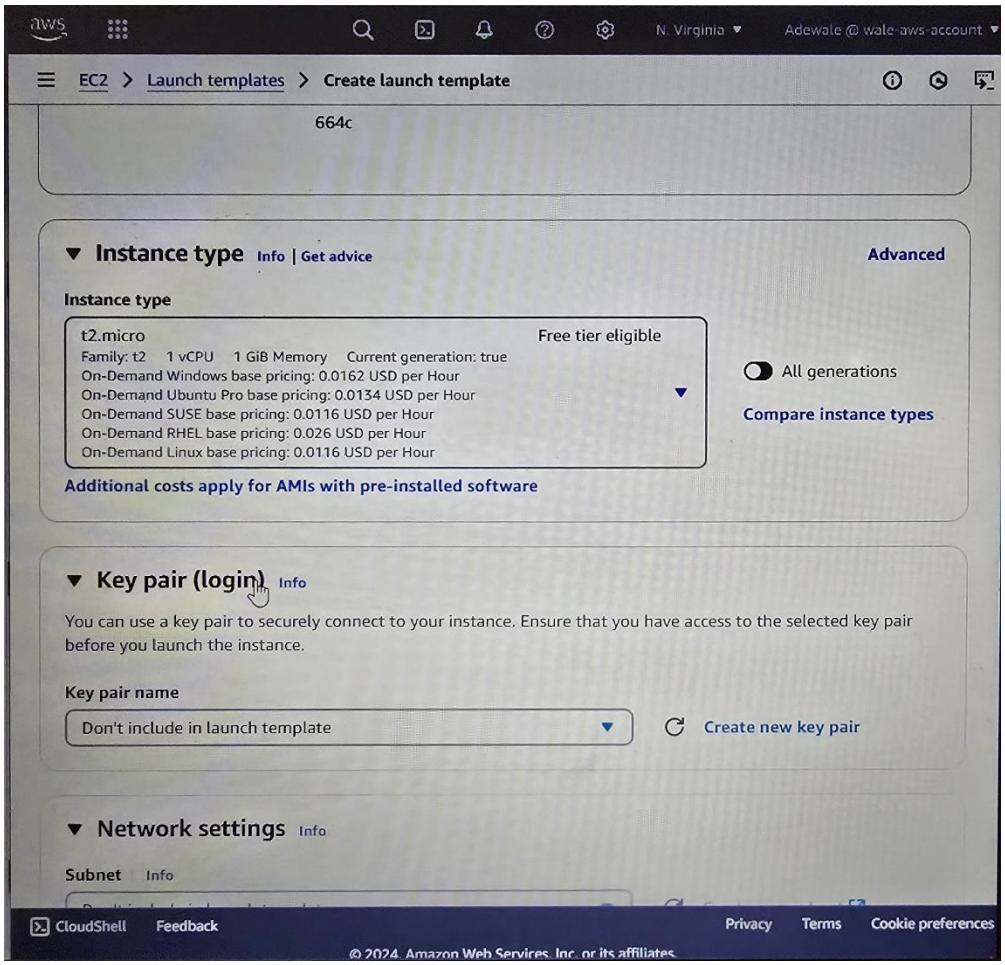
Launch EC2s in Auto Scaling Group (ASG)

Navigate to your EC2 dashboard, scroll down, on the left side, click "**Auto Scaling Groups**", then "**Create Auto Scaling group**".

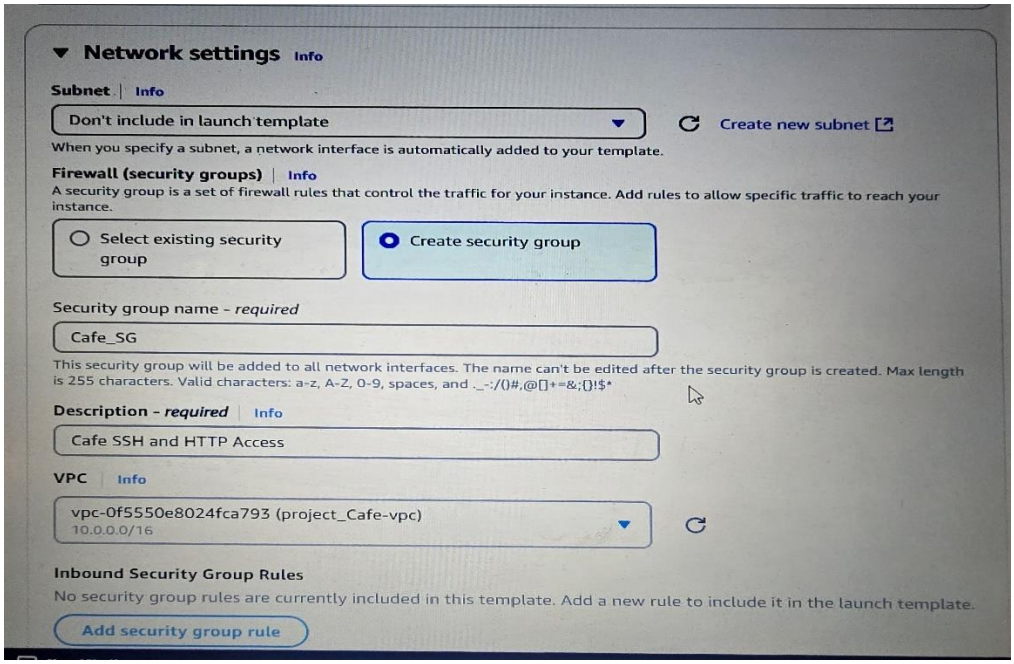
Name your ASG, then click "**Create a launch template**". This template will contain pre-configurations to launch our public facing EC2 Instance from the ASG.



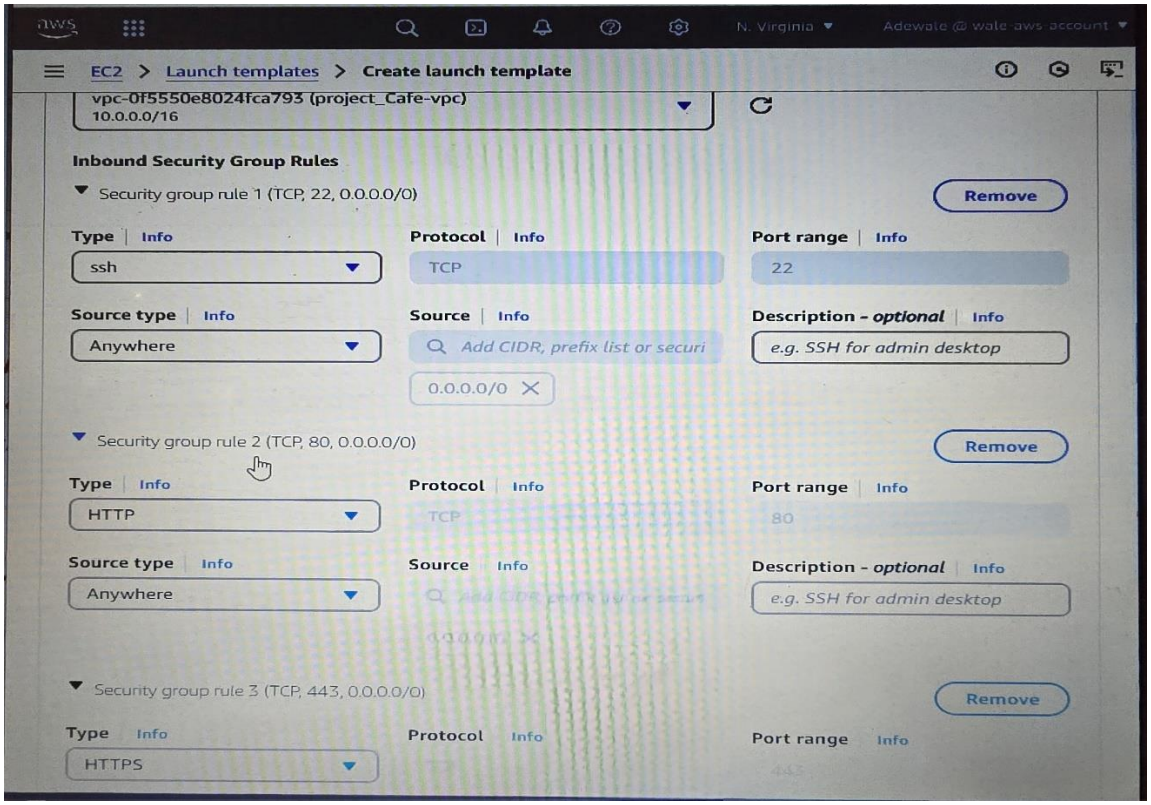

N. Virginia
Adewale @ wale-aws-account
EC2 > Launch templates > Create launch template
Recents
Quick Start
Amazon Linux
macOS
Ubuntu
Windows
Red Hat
SUSE
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community
Amazon Machine Image (AMI)
Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type
ami-0166fe664262f664c (64-bit (x86)) / ami-07bcScc4add81dad9 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs
Free tier eligible
Description
Amazon Linux 2 comes with five years support. It provides Linux kernel 5.10 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is now under maintenance only mode and has been removed from this wizard.
Amazon Linux 2 Kernel 5.10 AMI 2.0.20241113.1 x86_64 HVM gp2
Architecture
64-bit (x86)
AMI ID
ami-0166fe664262f664c
Username
ec2-user
Verified provider
Instance type Info | Get advice
Advanced
CloudShell
Feedback
Privacy
Terms
Cookie preferences


N. Virginia
Adewale @ wale-aws-account
EC2 > Launch templates > Create launch template
664c
Instance type Info | Get advice
Advanced
Instance type
t2.micro
Free tier eligible
Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Windows base pricing: 0.0162 USD per Hour
On-Demand Ubuntu Pro base pricing: 0.0134 USD per Hour
On-Demand SUSE base pricing: 0.0116 USD per Hour
On-Demand RHEL base pricing: 0.026 USD per Hour
On-Demand Linux base pricing: 0.0116 USD per Hour
All generations
Compare instance types
Additional costs apply for AMIs with pre-installed software
Key pair (login) Info
You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.
Key pair name
Don't include in launch template
Create new key pair
Network settings Info
Subnet Info
CloudShell
Feedback
Privacy
Terms
Cookie preferences
© 2024, Amazon Web Services, Inc. or its affiliates.

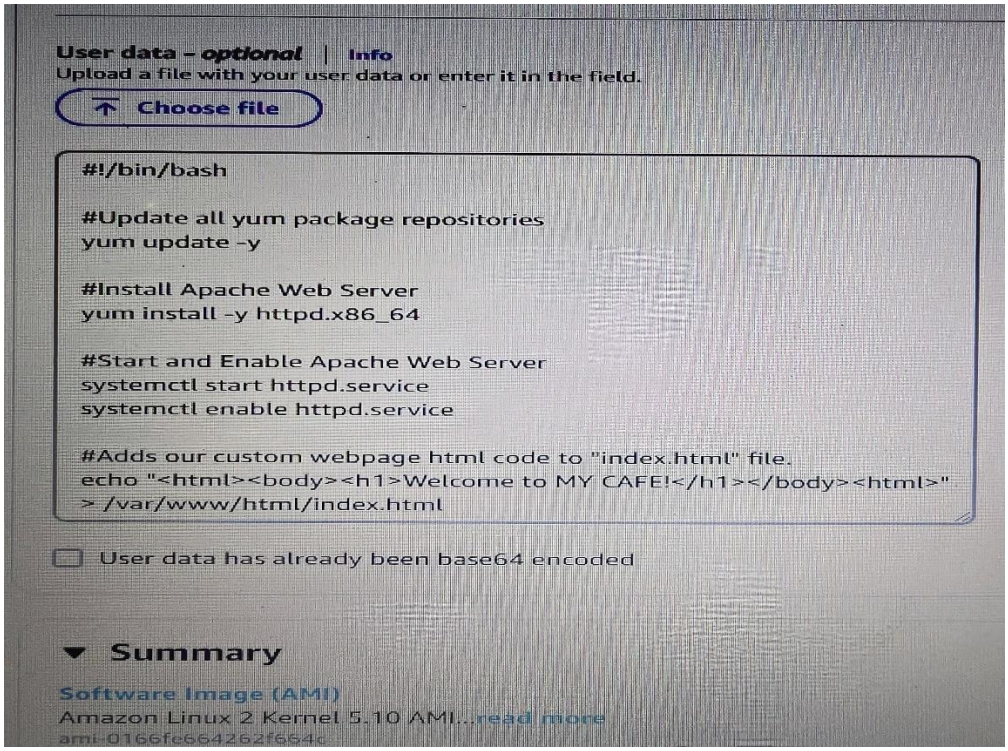
Continue by choosing an existing key pair or creating a new a key pair, then create a new Security Group. Name your security group and make sure to select your previously created VPC.



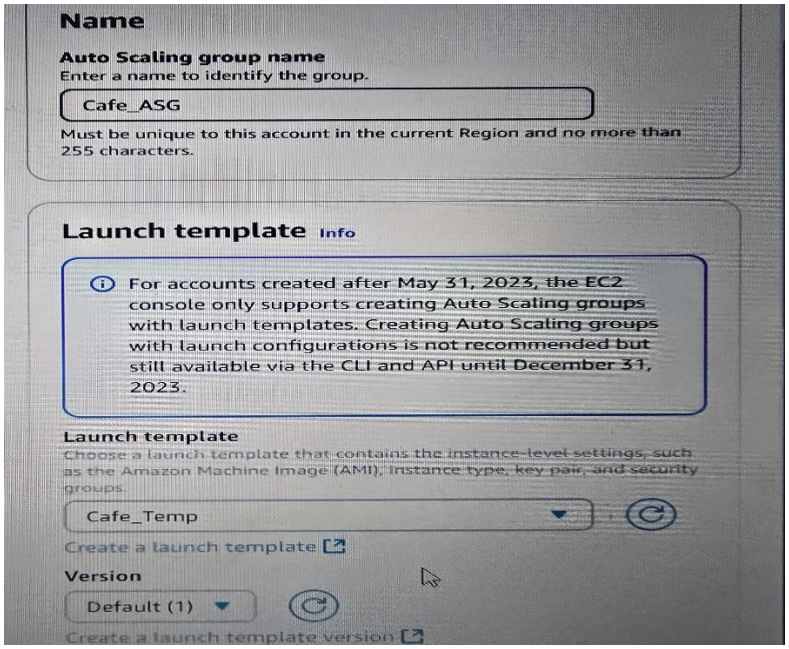
Configure your inbound rules to allow SSH, HTTP, and HTTPS access to our public facing EC2 Instances from anywhere



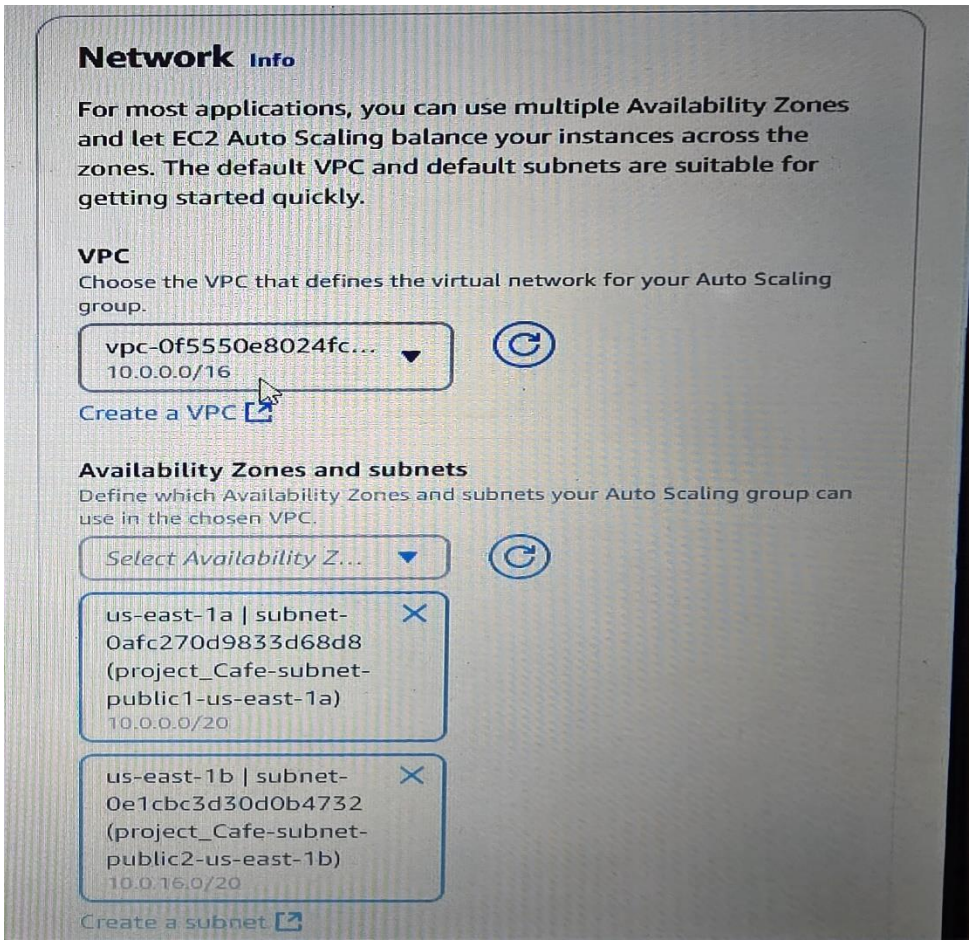
We will leave the rest of the options as default and scroll down to the the “**Advanced details**”. Scroll down and paste the following script in the “**User data**” field, then click “**Create launch template**”.



After the launch template has been successfully created, head back to the ASG window, select the launch template just created, then click “**Next**”.

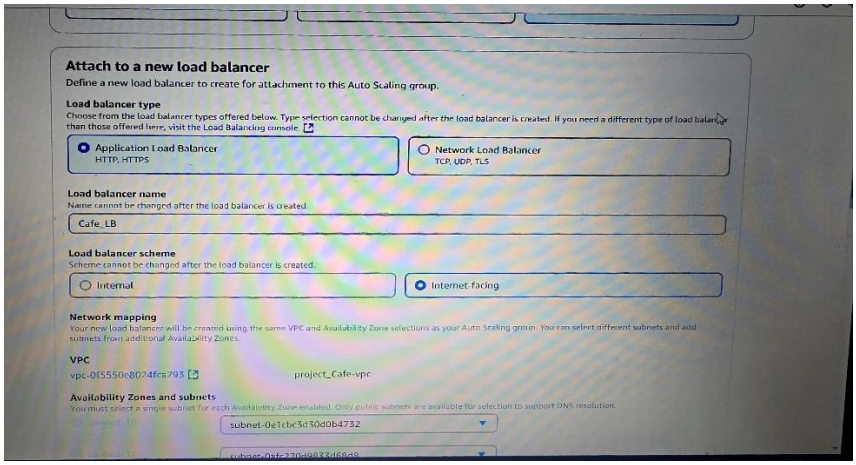


For the “**Network**” settings, choose your VPC, select the 2 public AZs in your network, then click “**Next**”.

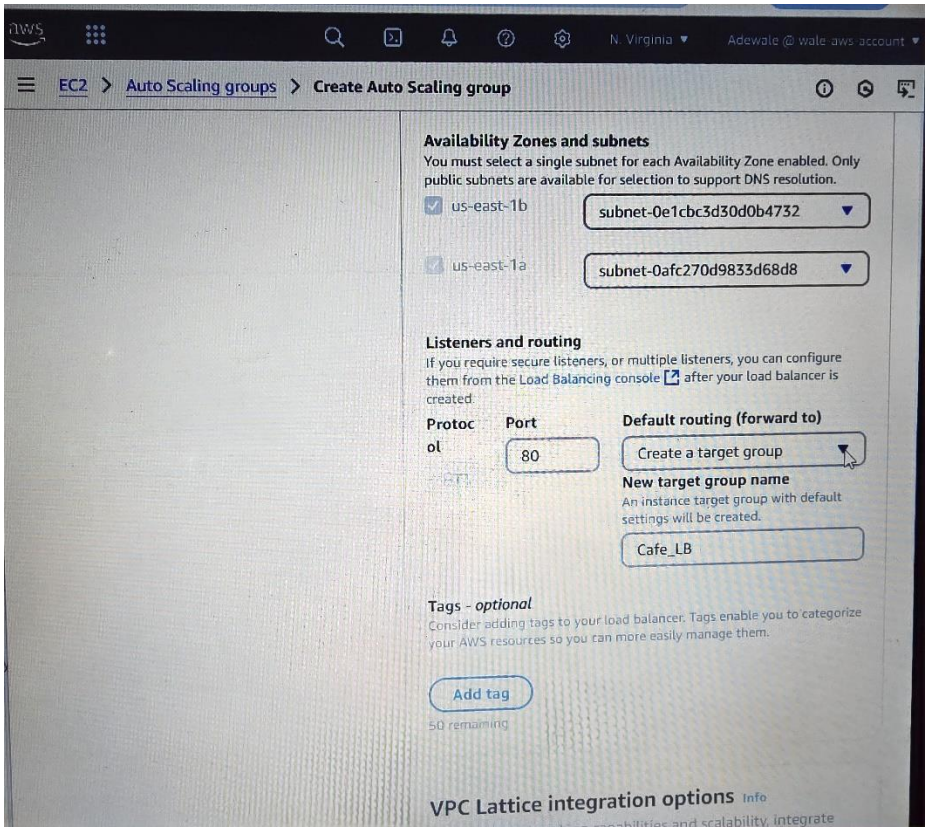


Now, we will configure a load balancer that will be used to distribute incoming Web Tier traffic across our EC2 instances to increase availability.

Select “**Attach to a new load balancer**”, then select “**Application Load Balancer**”. Name your load balancer, then select “**Internet-facing**”. Make sure your VPC and both public subnets are selected.



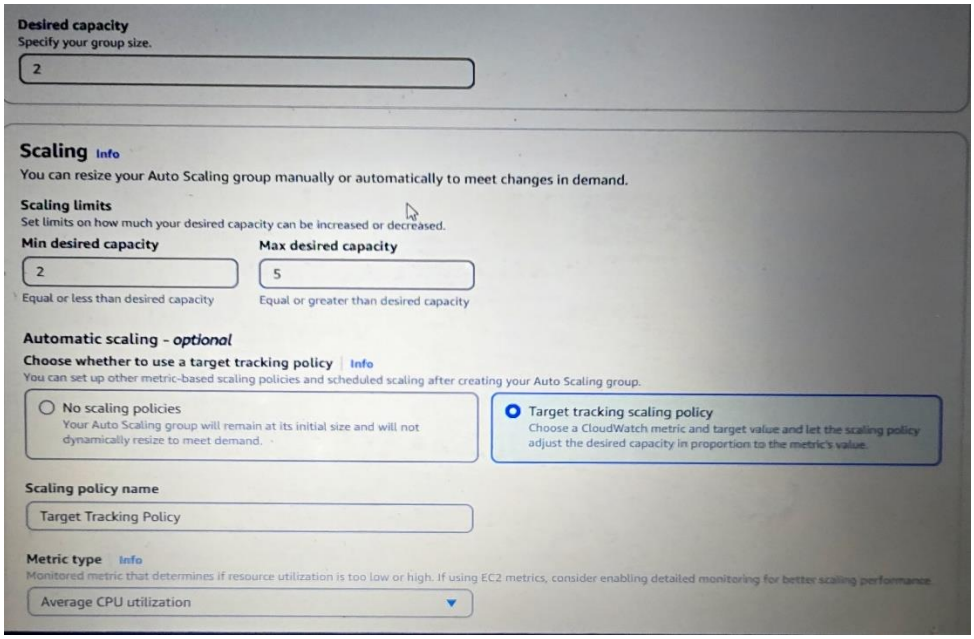
For “**Listeners and routing**”, select “**Create a target group**”, then select our new load balancer.



Scroll down, select "**Enable group metrics collection within CloudWatch**" to allow the collection of information of our ASG, then click "**Next**".

For the "**Configure group size and scaling policies**" section, chose 2 for desired capacity, 2 for minimum capacity and 5 for maximum capacity.

For "**Scaling policies**", select "**Target scaling policy**" and make sure the Metric type is "**Average CPU utilization**" and "**Target value**" is set to 50, then click "**Next**"

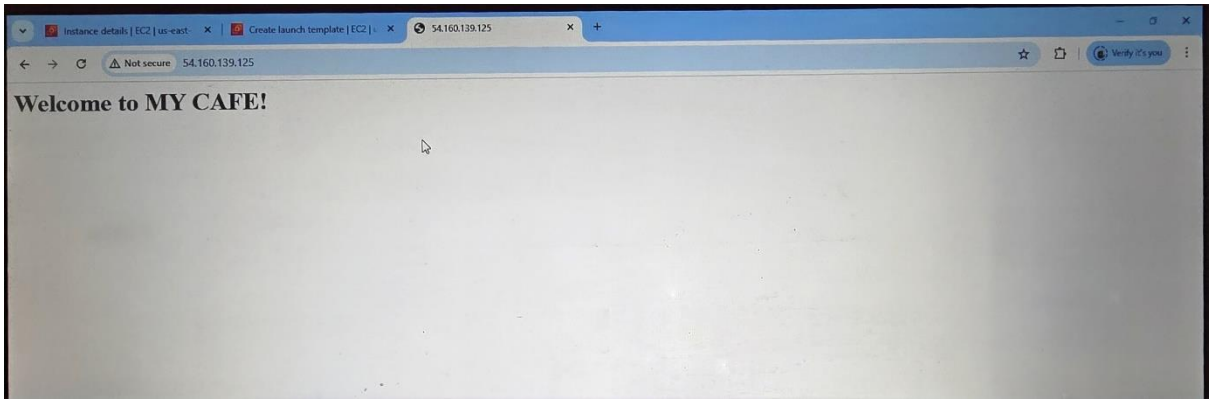


Continue to click "**Next**" until you arrive at the "**Review**" page, scroll down, then click "**Create Auto Scaling group**".

Your Web Tier's Auto Scaling group should be created. Wait a few minutes for it to update the capacity and create the new EC2 Instances.

Now, if you navigate to your EC2 Instances, you should see two EC2 Instances launched. After the Instances state has changed to "**Running**", copy the public IPv4 address of one, the paste it in your browser.

You should be able to view your Website, as shown below.



Additionally, if you navigate to your Application Load Balancer, copy the DNS name and paste it your browser, you should also be able to view your website.

We've successfully created built the Web Tier; we can **proceed to Step 3: Building the Application Tier!**

Step 3: Build the Application Tier

This layer manages the application logic:

1. Create an ASG for EC2 instances in private subnets.
2. Configure an internal ALB to route traffic from the Web Tier to the Application Tier.
3. Define Security Group rules to restrict access to only the Web Tier.

HOW?

For the Application Tier, we will create another ASG to launch EC2 Instances in 2 private subnets that will allow inbound traffic access from the Web Tier.

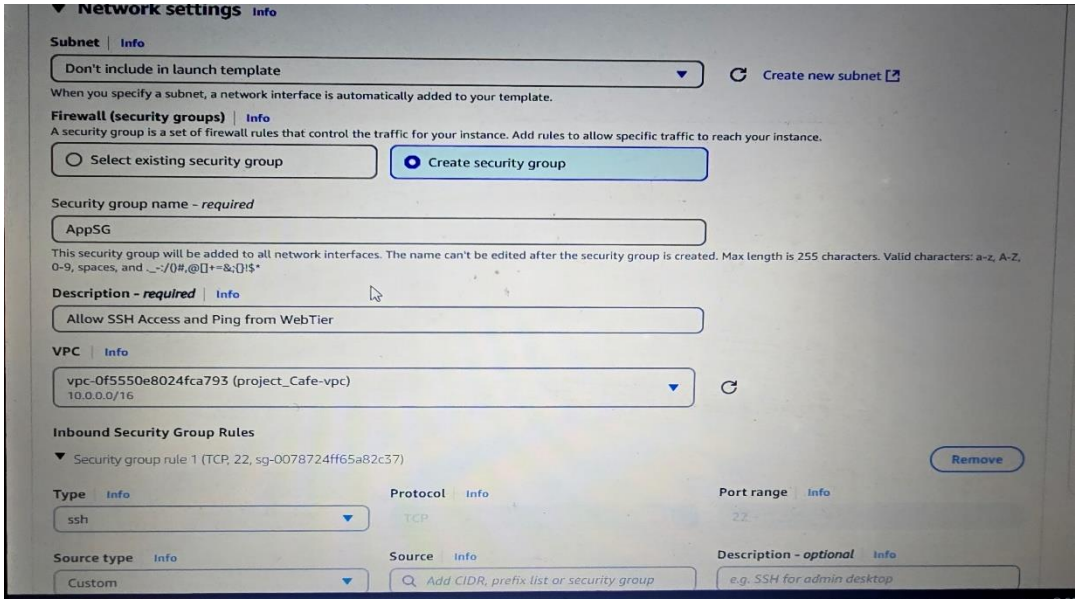
Note — This is not a true application tier as we don't have any provided code to run on the EC2 instances.

Launch EC2s in Auto Scaling Group (ASG)

As we did in the Web Tier, navigate to your EC2 dashboard, scroll down and on the left, click "**Auto Scaling Groups**", then "**Create Auto Scaling group**".

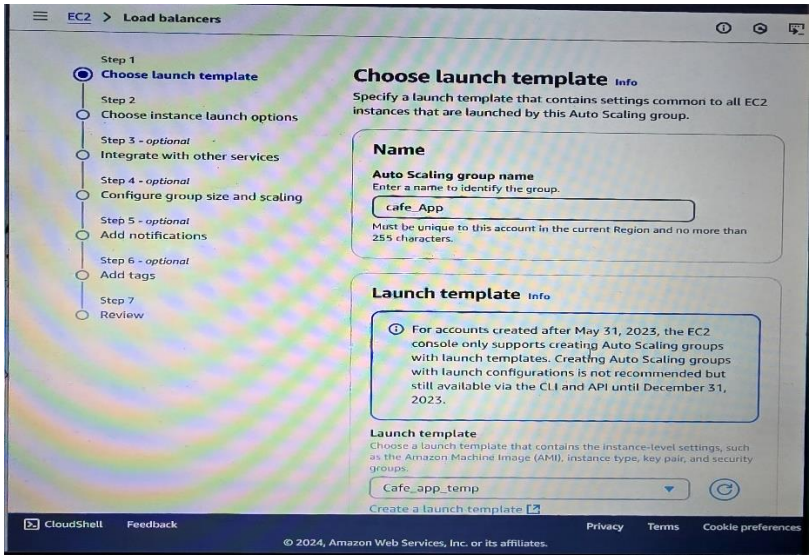
Name your ASG, then click "**Create a launch template**". Choose the same AMI (Amazon Linux 2) and instance type (t2.micro) as we did in the previous launch template, then choose a key pair.

Proceed to the "**Network setting**" and create a new security group. Choose your VPC, then configure the security group rules to allow SSH access from the Web Tier security group and ICMP — IPV4 to allow us to ping our EC2s from the Web Tier to verify connectivity.

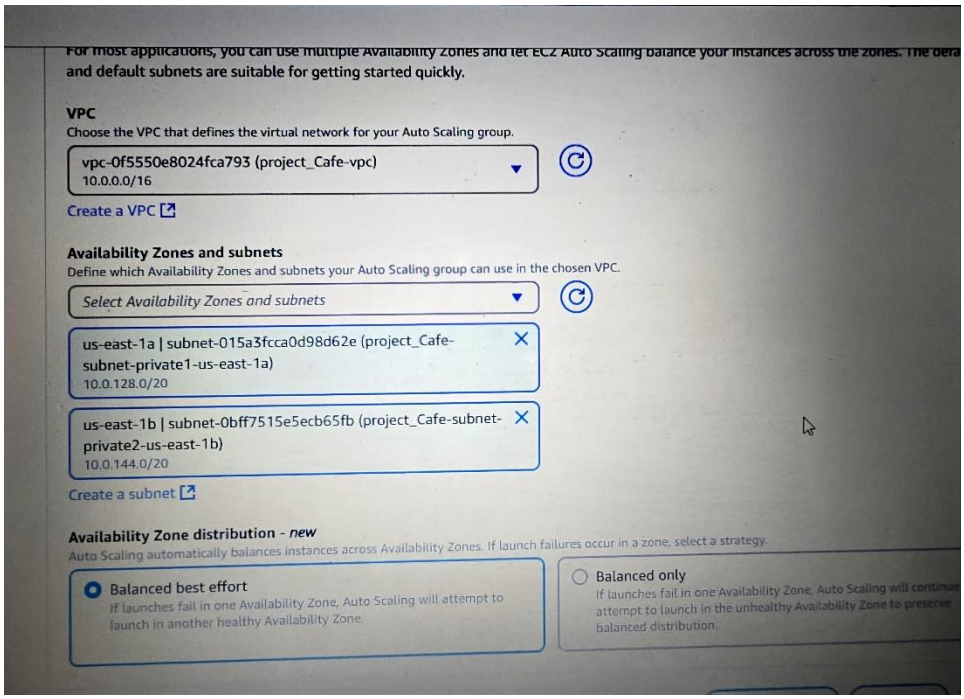


Now that we have configured our Application tier launch template, we can now click, "**Create launch template**".

Navigate back to the Application Tier ASG configuration and select the newly created launch template.

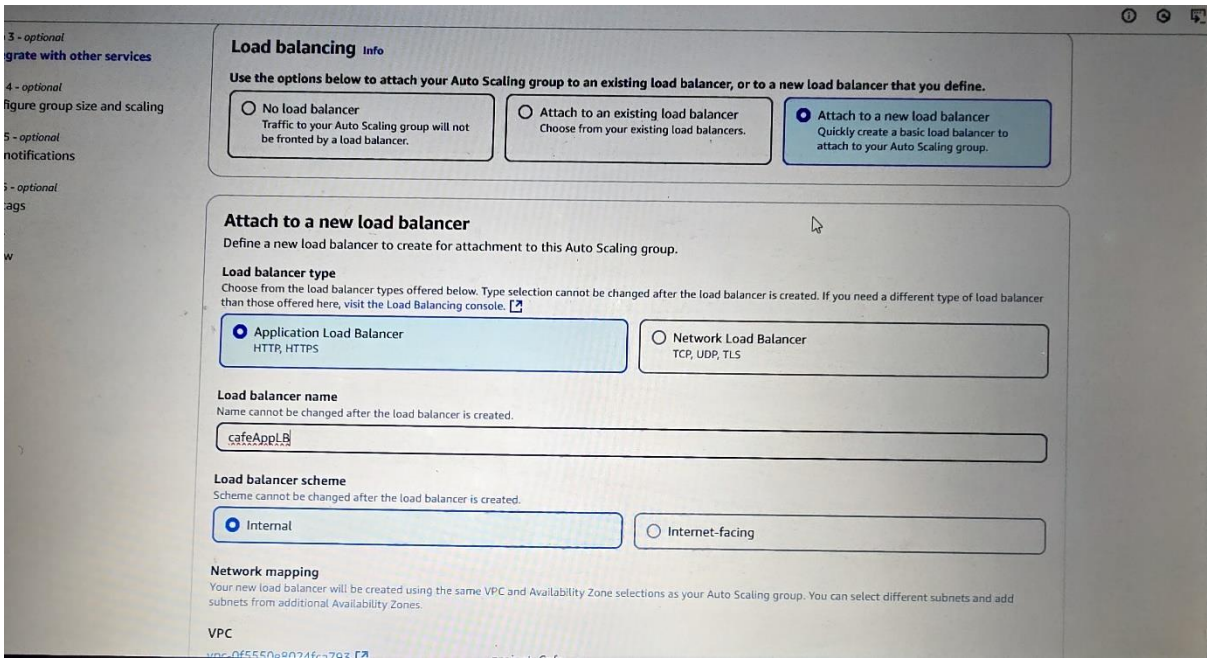


Choose your VPC, select the 2 private AZs in your network which will host our EC2 Instances for the Application Tier, then click “**Next**”.

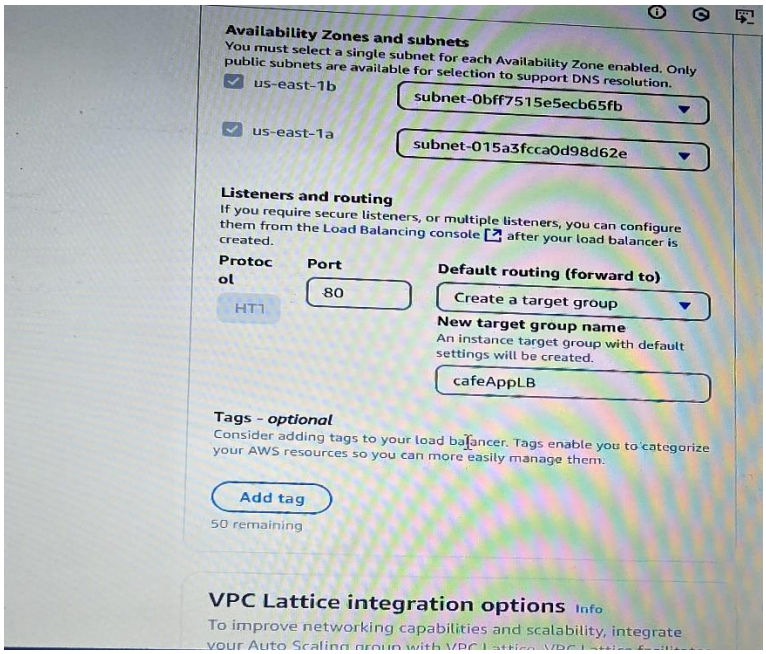


We will also create an Application Load Balancer to distribute traffic from the Web Tier to the Auto Scaling Group of EC2’s in the Application Tier.

Select “**Attach to a new load balancer**”, then “**Application Load Balancer**”. Give it a name, then select “**Internal**” as this is for private traffic only from our Web Tier and from the public internet.



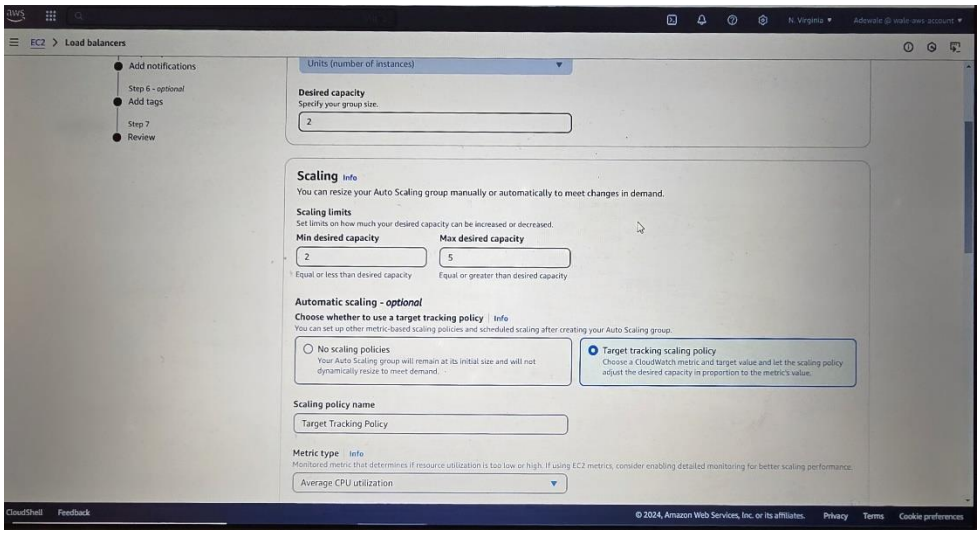
Make sure your VPC and both private subnets are selected. For “**Listeners and routing**”, select “**Create a target group**” and choose our Application Tier load balancer.



Scroll down, select “**Enable group metrics collection within CloudWatch**” to allow the collecting of information of our ASG, then click “**Next**”.

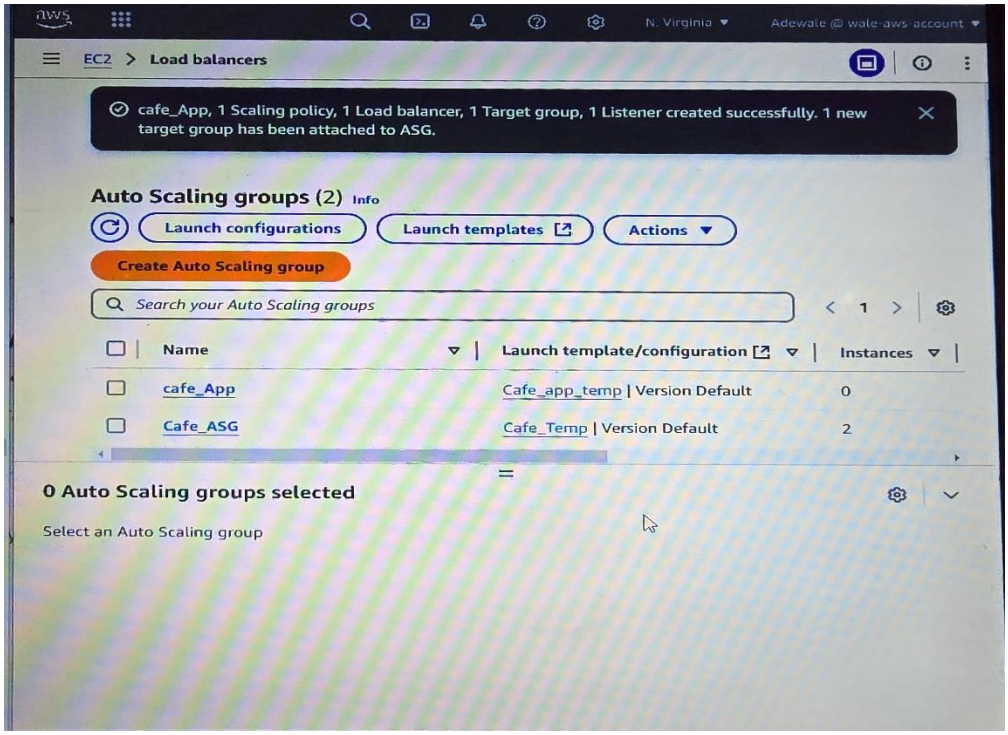
For the “**Configure group size and scaling policies**” section, we will the choose the same group settings as with the Web Tier, 2 for desired capacity, 2 for minimum capacity and 5 for maximum capacity.

Also, select “**Target scaling policy**” and make sure the metric type is set to “**Average CPU utilization**” and “**Target value**” is set to 50, then click “**Next**”.

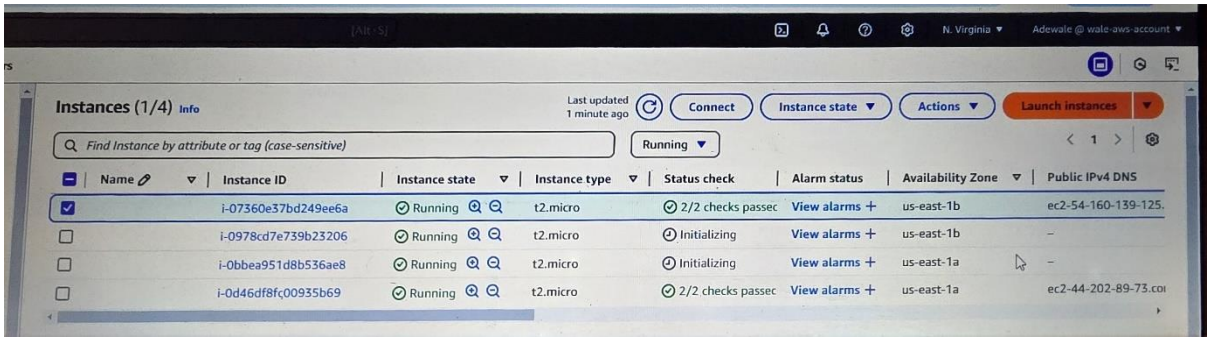


Click “**Next**”, until you arrive at the “**Review**” page, scroll down, then click “**Create Auto Scaling group**”.

Our new Application Tier ASG should be created.



Wait a few minutes for it to update the capacity and create the new EC2 Instances. We should now see 4 EC2 Instances. 2 from our Web Tier and 2 from our Application Tier, all launched from separate ASG’s each fronted with an Application Load Balancer.

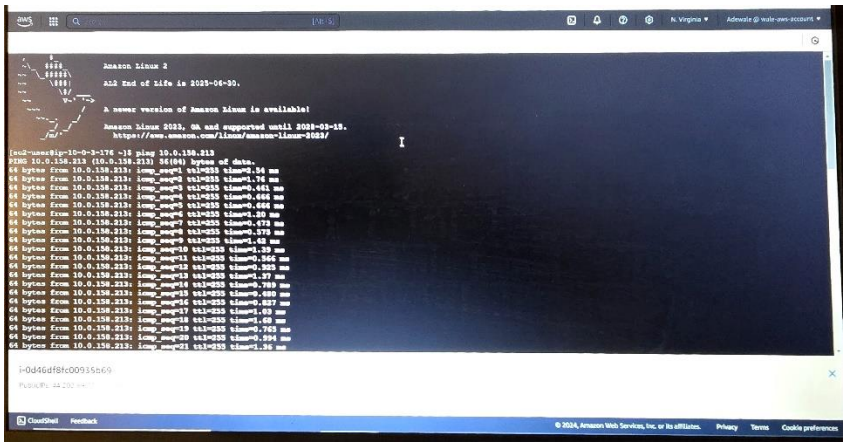


We now need to verify connectivity between the two tier’s by pinging the Application Tier from the Web Tier. We can achieve this by connecting through ssh to an EC2 Instance in the Web Tier using the public IPv4 address, then running to “**ping**” command to the private IPv4 address of an EC2 Instance in the Application Tier.

Now that we are connected to our Web Tier EC2 Instance, we can ping our Application Tier EC2 Instance by first obtaining the private IPv4 address of one of the EC2 Instances in the private subnet (Application Tier), then running the following command —

ping <app_tier_ec2_private_ip_address>

You should start to receive responses, as seen below.



Success!

This verifies connectivity between the two tiers!

Now that we've determined that we can reach our private IP, let's ssh into it from our Web Tier EC2. To accomplish this, first exit out of you EC2 connection by running the following command —

exit

Connect back into your Web Tier EC2 Instance via ssh by running the following command to add the ssh forwarding agent—

ssh -A ec2-user@<web_tier_ec2_public_ipv4_address>

After connecting to your EC2 Instance, run the following command to ssh into your Application Tier EC2 instance in a private subnet —

ssh -A ec2-user@<app_tier_ec2_private_ipv4_address>

You should now be connected to the Application Tier's EC2 Instance.

```
(base) ifeanyiro@Ifeanyichukwus-MacBook-Air Keys % ssh -A ec2-user@54.175.134.67
Last login: Wed Jan 11 04:32:21 2023 from 200.50.89.222

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-0-4-118 ~]$ ssh -A ec2-user@10.0.147.189
Last login: Wed Jan 11 04:33:20 2023 from ip-10-0-4-118.ec2.internal

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-0-147-189 ~]$
```

Success!

This verifies ssh connectivity from the Web Tier's EC2 Instance to the Application Tier's EC2 Instance!

Now that we've successfully built the Application Tier and verified connectivity from the Web Tier, we can **proceed to Step 4: Building the Database Tier!**

Step 4: Build the Database Tier

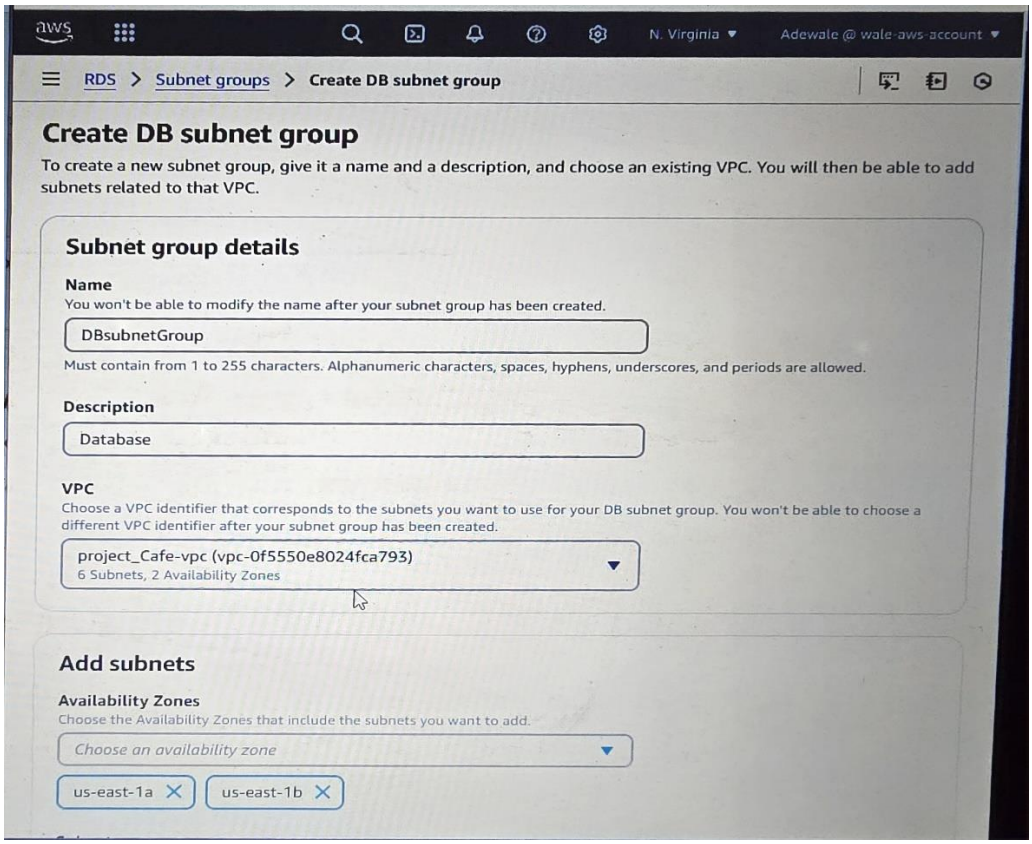
To store and process data:

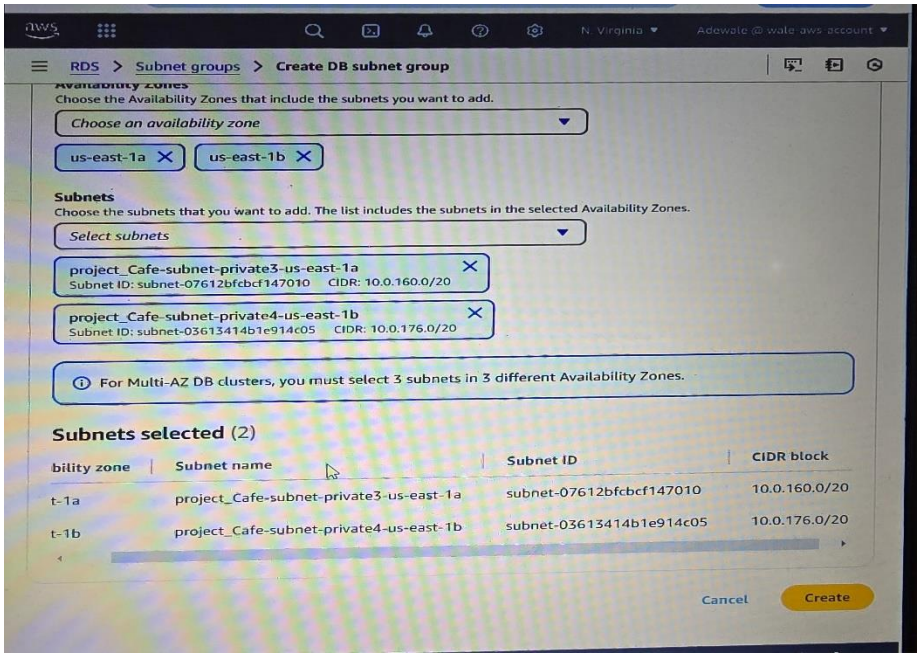
1. Create a MySQL database using Amazon RDS in private subnets with Multi-AZ deployment for high availability.
2. Configure a DB subnet group and adjust inbound rules to permit access only from the Application Tier.

HOW?

Navigate to your Amazon RDS dashboard, click "**Subnet groups**", then "**Create DB subnet group**". Name and describe your subnet group, then choose your VPC.

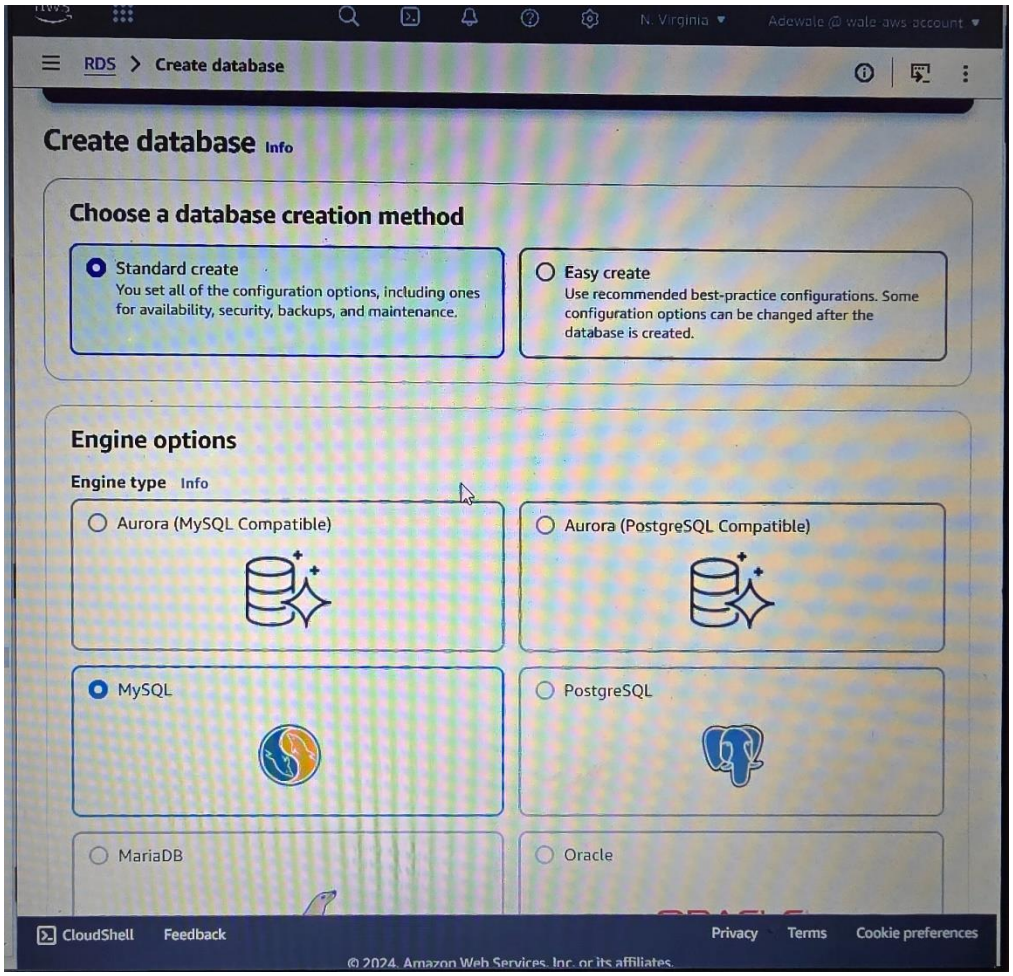
For higher availability and data redundancy, I will choose a Multi-AZ DB Instance deployment. To achieve this, choose your two AZs, then make sure to select the remaining two private subnets that has not been used to host our DB Instances, the create the group.



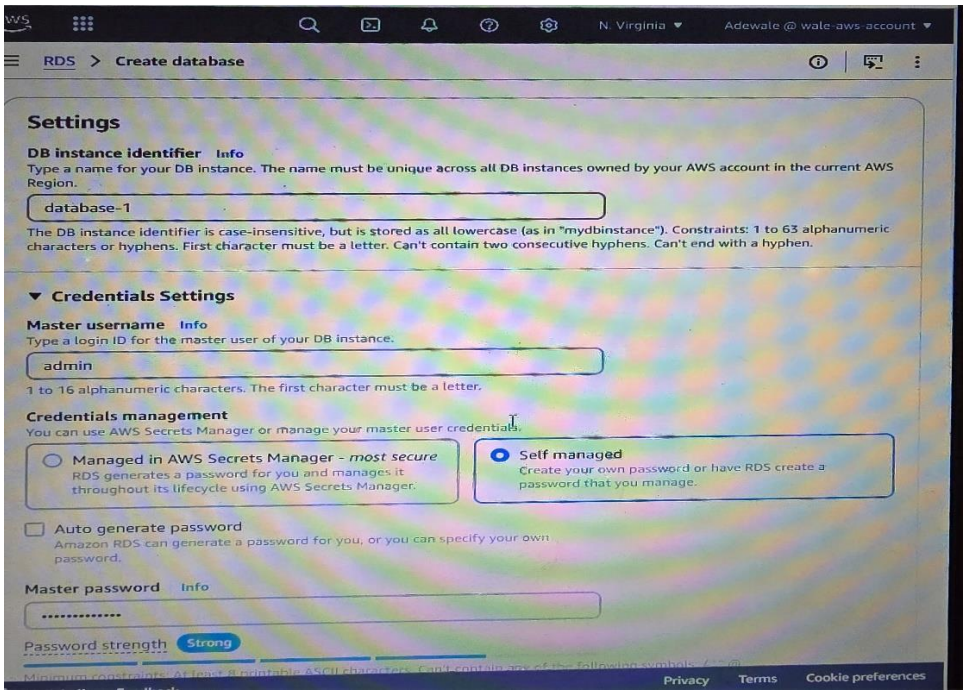


Navigate back to the RDS dashboard, click “**Databases**”, then “**Create database**”.

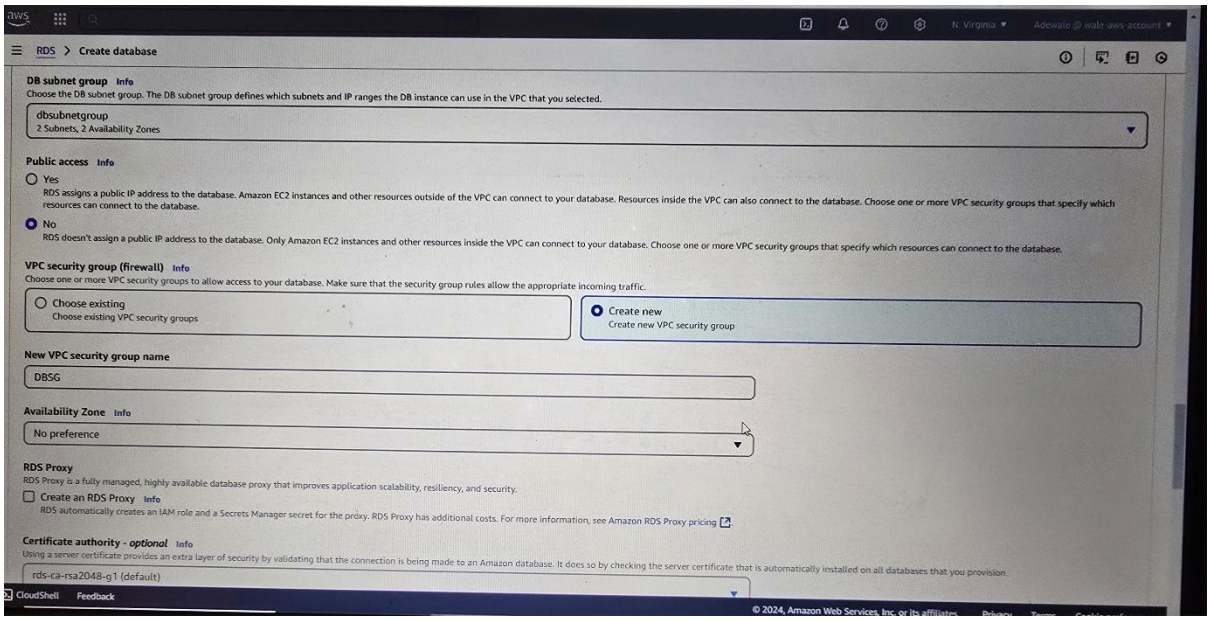
Select “**Standard create**” and “**MySQL**”.



For the “**Templates**” I will choose “**Dev/Test**” option to select Multi-AZ DB Instance, but you can choose the “**Free tier**”. For “**Settings**”, set a master password for the admin.

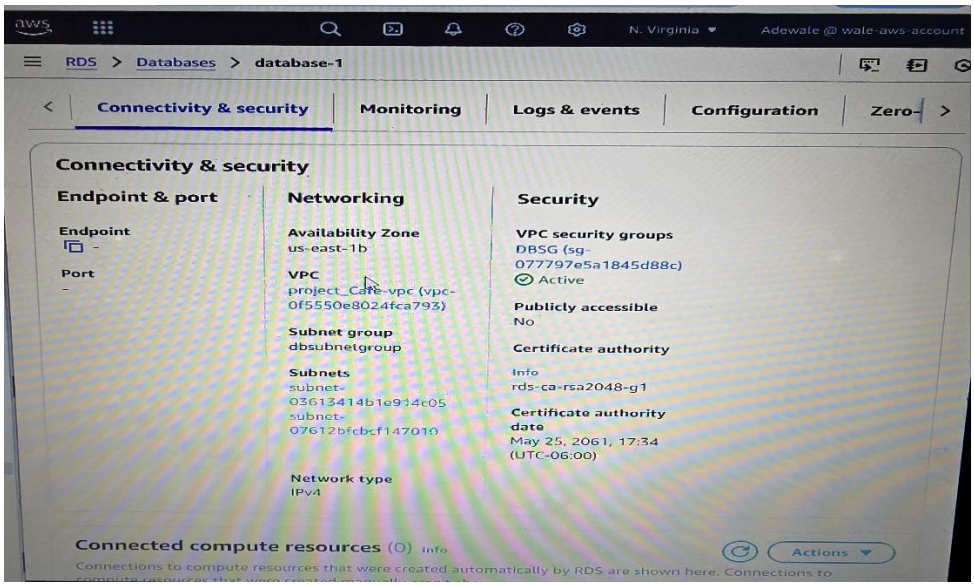


Continue to “**Connectivity**” and select your VPC and DB subnet group previously created. Keep “**Public access**” set to “**No”,** as this will only be access privately through the Application Tier. Select “**Create new**” security group, then give it a name.



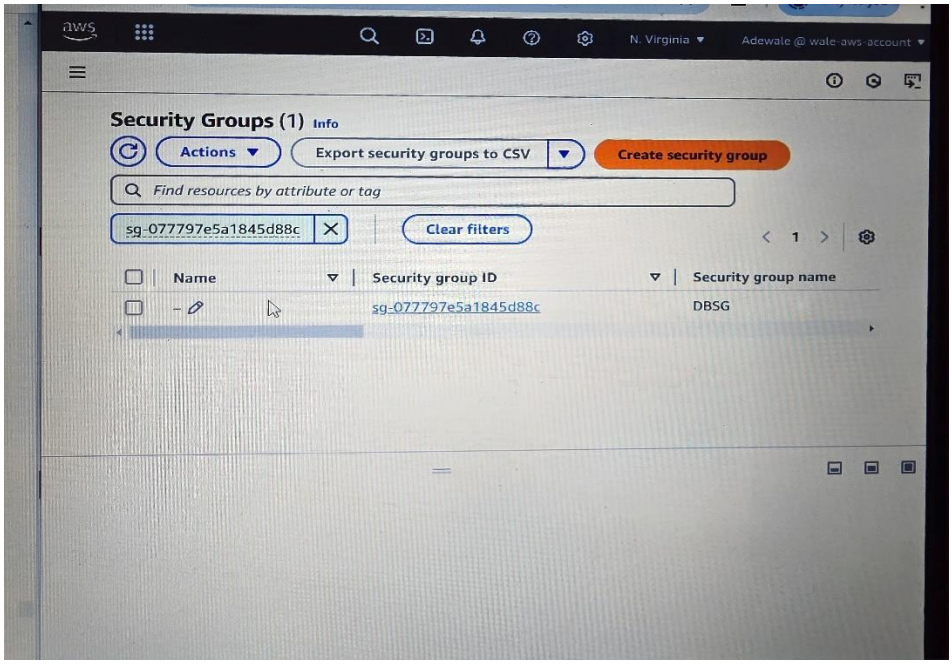
Scroll down and “**Create Database**”

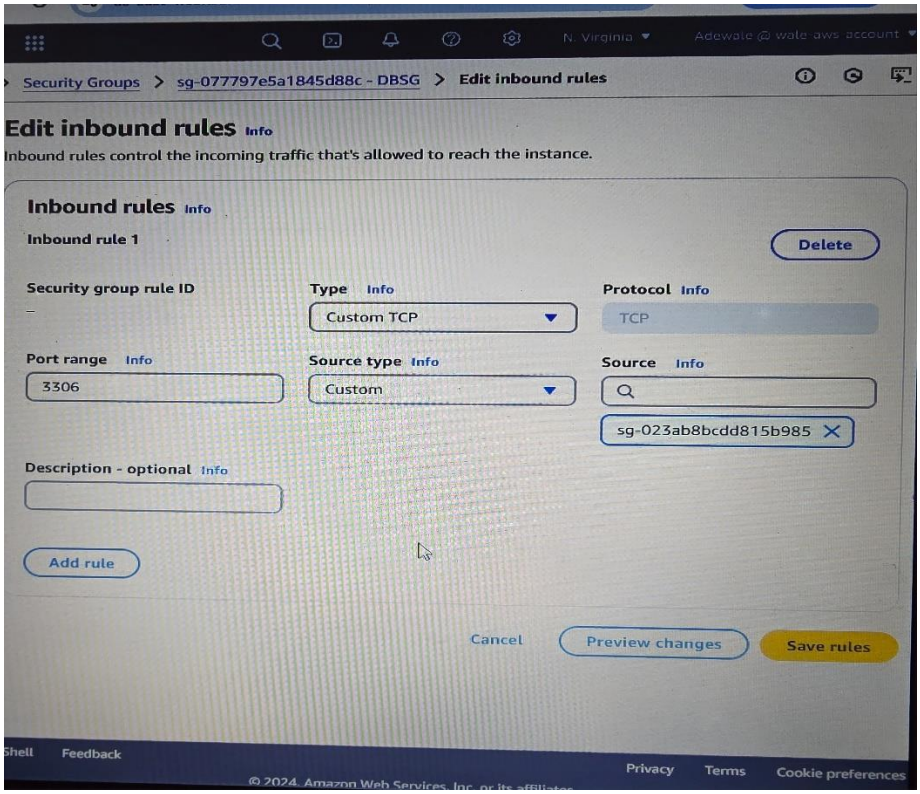
Select your new Database, then in the “**Connectivity & security tab**”, click the security group to edit it.



Click on “**Edit inbound rules**” on the bottom right, then change the source of the inbound rule to the security group of your Application Tier.

Note: if you encounter an error, create a new rule with the same Type — MySQL/Aurora, Protocol — TCP and Port rang — 3306, then add the Application Tier security group. You can now, delete the old rule.





Success!

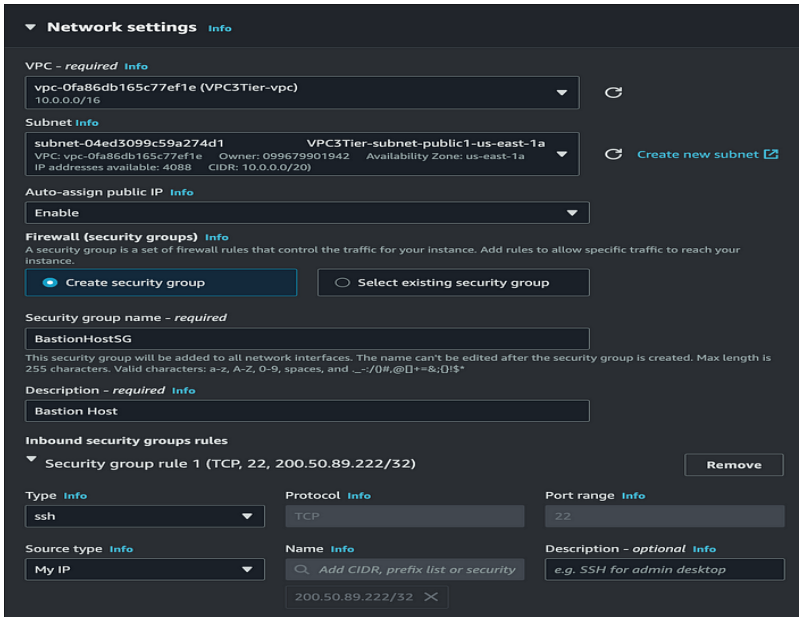
We've successfully built the Database Tier! Let's **proceed to Step 5: Creating a Bastion Host.**

Step 5: Understanding and creating a Bastion Host

A Bastion Host, also known as a jump server, is a remote access server used to securely access a private network from a public network, such as the internet. We will create a dedicated EC2 Instance in the Web Tier to be a Bastion Host which will be configured to act as a secure gateway for us to connect into our private subnet hosting our Application Tier EC2 Instances, instead of using our ASG EC2 Instances.

Proceed by navigating to the EC2 dashboard and clicking "**Launch as instance**". Name the Instance, select Amazon Linux AMI and t2.mirco Instance type, then choose a key pair.

In the "**Network settings**", choose your VPC, select a public subnet, enable auto-assign public IP, then create a new security group to only allow ssh from your IP address. After this, you can "**Launch Instance**".



Remember, to edit the rules in the security groups of our EC2 Instances in the Application Tier's private subnets to allow ssh from our Bastion Host security group and remove the rule to allow ssh from our Web Tier security group.

We can now ssh into our Bastion Host by running the following command again but with the public IPV4 of the Bastion Host —

ssh -A ec2-user@<bastion_host_public_ipv4_address>

After connecting to your EC2 Instance, run the following command to ssh into one of your Application Tier EC2 Instance in a private subnet —

ssh -A ec2-user@<application_tier_ec2_private_ipv4_address>

Success!

You should now be connected to the Application Tier's EC2 Instance from your Bastion Host.

```
(base) ifeanyiro@Ifeanyichukwus-MacBook-Air Keys % ssh -A ec2-user@3.80.48.181
The authenticity of host '3.80.48.181 (3.80.48.181)' can't be established.
ED25519 key fingerprint is SHA256:5k5tHBvsgP3RuuJPcCO7ntv/F6RCZ5lxRufKUiStBQs.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '3.80.48.181' (ED25519) to the list of known hosts.

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-0-14-208 ~]$ ssh -A ec2-user@10.0.138.125
The authenticity of host '10.0.138.125 (10.0.138.125)' can't be established.
ECDSA key fingerprint is SHA256:AzvaktVlQtWSlU+LJ4qBPXPeGu4Zr7hIayUZLt+t4Nc.
ECDSA key fingerprint is MD5:3d:a9:5c:4a:a1:2d:d3:cf:ff:ca:6d:2a:08:0b:98:b4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.138.125' (ECDSA) to the list of known hosts.

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-0-138-125 ~]$
```

created by

Adewale Salami