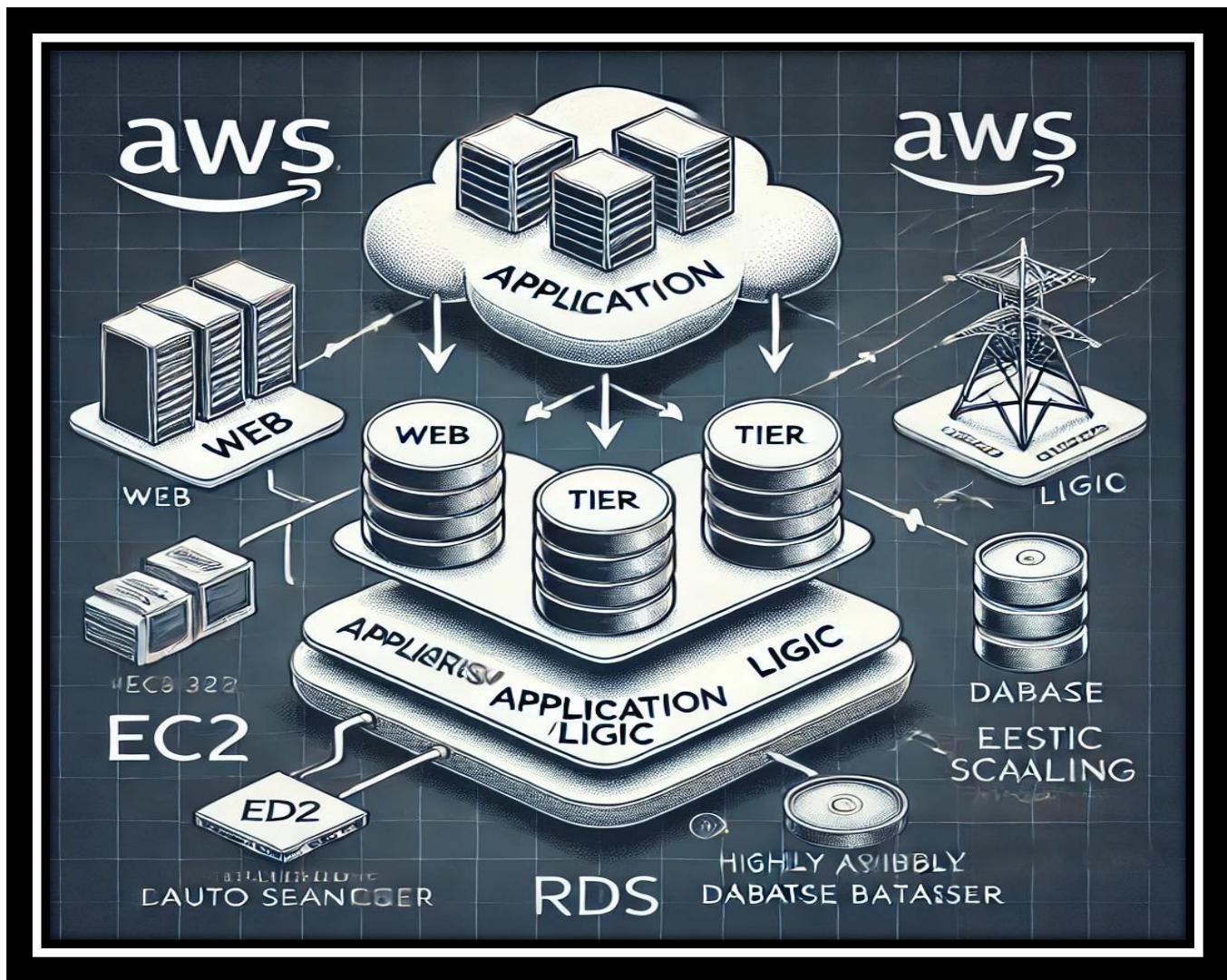
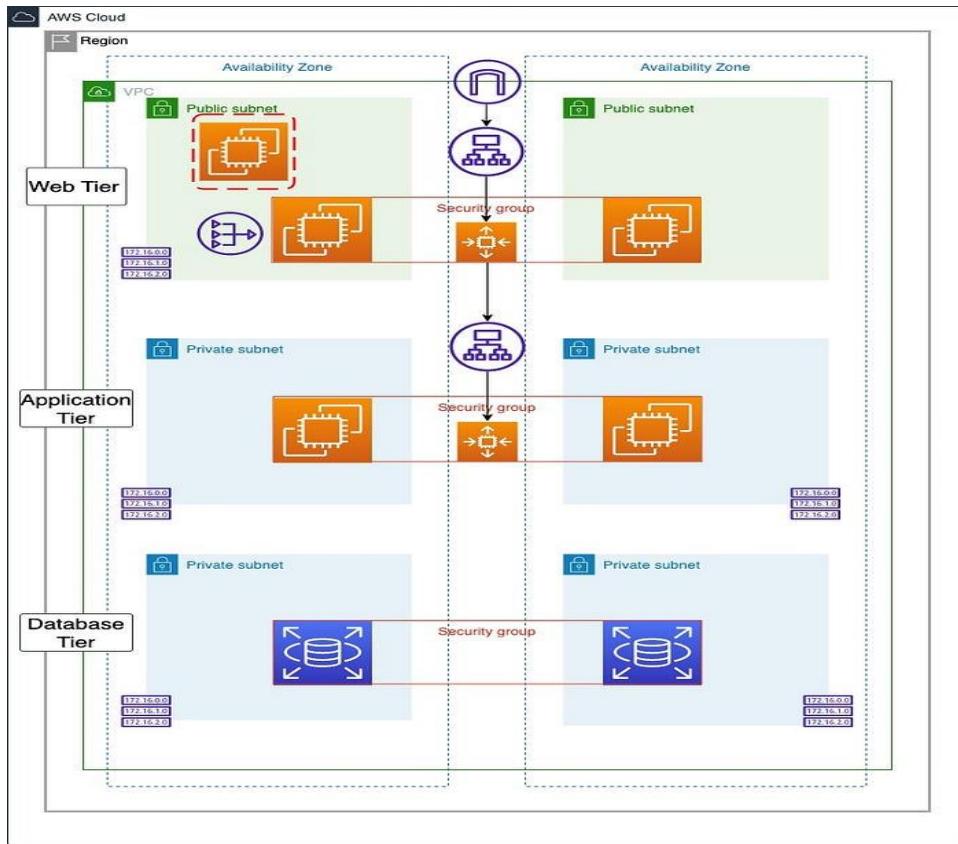


Creating a Three-Tier Architecture for a Café



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

- Scalability:** Each tier can be scaled independently to manage varying workloads.
- Reliability:** Isolating tiers enhances system stability; issues in one tier won't directly impact the others.
- Security:** By separating tiers onto different servers, sensitive data and operations are more secure.
- 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**.

The screenshot shows the 'Create VPC' wizard in the AWS VPC service. On the left, the 'VPC settings' section includes fields for 'Name tag auto-generation' (checked for 'Auto-generate' and 'project_Cafe'), 'IPv4 CIDR block' (set to '10.0.0.0/16'), and 'Tenancy' (set to 'Default'). On the right, the 'Preview' section shows a diagram of the VPC structure. It features a central 'VPC' node labeled 'Show details' and 'Your AWS virtual network'. Six 'Subnets' are listed under the heading 'Subnets (6) Subnets within this VPC': 'us-east-1a' contains 'project_Cafe-subnet-public1-us' and 'project_Cafe-subnet-private1-us'; 'us-east-1b' contains 'project_Cafe-subnet-public2-us' and 'project_Cafe-subnet-private2-us'; and 'us-east-1c' contains 'project_Cafe-subnet-public3-us' and 'project_Cafe-subnet-private3-us'. Five 'Route tables' are listed under 'Route tables (5) Route network traffic to resources': 'project_Cafe-rtb-public', 'project_Cafe-rtb-private1-us-e', 'project_Cafe-rtb-private2-us-e', 'project_Cafe-rtb-private3-us-e', and 'project_Cafe-rtb-private4-us-e'.

The screenshot shows the 'Configure VPC' page in the AWS VPC service. It includes several configuration sections:

- Number of Availability Zones (AZs)**: Set to 3.
- Number of public subnets**: Set to 2.
- Number of private subnets**: Set to 4.
- NAT gateways (\$)**: Set to 'In 1 AZ'.
- VPC endpoints**: Options include 'None' and 'S3 Gateway'.

“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.

The screenshot shows a "Success" status for a CloudFormation stack named "vpc-Of5550e8024fca793". The "Details" section lists 32 successful actions, including creating the VPC, subnets, route tables, and internet gateway, along with their respective sub-resources like elastic IPs and route entries.

The screenshot displays the AWS VPC Resource Map. The left navigation bar includes tabs for "Resource map", "CIDRs", "Flow logs", "Tags", and "Integrations". The main area shows the "Resource map" view with four main components: "VPC", "Subnets (6)", "Route tables (6)", and "Network connections (2)". The "VPC" section shows "project_Cafe-vpc". The "Subnets (6)" section shows subnets grouped by AZ: "us-east-1a" (public, private1, private2, private3) and "us-east-1b" (public, private4, private2). The "Route tables (6)" section lists route tables for each subnet. The "Network connections (2)" section lists the Internet Gateway (igw) and a NAT gateway (nat).

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**”.

Screenshots showing the AWS VPC Subnets list and the subnet configuration details.

Subnets (1/17) Info

Name	Subnet ID	State	VPC	Block Public...
Private-1B	subnet-0d428c8eee46b45e2	Available	vpc-013d189f49d62de84 MyV...	Off
project_Cafe-subnet-private1-us-east-1a	subnet-015a3fcfa0d98d62e	Available	vpc-0f5550e8024fca793 proje...	Off
project_Cafe-subnet-private2-us-east-1b	subnet-0bff7515e5ecb65fb	Available	vpc-0f5550e8024fca793 proje...	Off
project_Cafe-subnet-private3-us-east-1a	subnet-07612bfcfcf147010	Available	vpc-0f5550e8024fca793 proje...	Off
project_Cafe-subnet-private4-us-east-1b	subnet-03613414b1e914c05	Available	vpc-0f5550e8024fca793 proje...	Off
project_Cafe-subnet-public1-us-east-1a	subnet-0afc270d9833d68d8	Available	vpc-0f5550e8024fca793 proje...	Off
project_Cafe-subnet-public2-us-east-1b	subnet-0e1cbc3d30d0b4732	Available	vpc-0f5550e8024fca793 proje...	Off
Public-1A	subnet-0d48ec7061df8c7e9	Available	vpc-013d189f49d62de84 MyV...	Off
Public-1B	subnet-026g4a8475f380192	Available	vpc-013d189f49d62de84 MyV...	Off

subnet-0afc270d9833d68d8 / project_Cafe-subnet-public1-us-east-1a

Details Flow logs Route table Network ACL CIDR reservations Sharing Tags

Details

Subnet ID	subnet-0afc270d9833d68d8	Subnet ARN	arn:aws:ec2:us-east-1:905418204129:subnet/subnet-0afc270d9833d68d8	State	Available	Block Public Access	Off
IPv4 CIDR	10.0.0.0/20	Available IPv4 addresses	4090	IPv6 CIDR	-	IPv6 CIDR association ID	-
Network border group	MyVPC	VPC	MyVPC	Network border group	MyVPC	VPC	MyVPC

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

Screenshots showing the AWS VPC Edit subnet settings page.

Edit subnet settings

Subnet

Subnet ID	subnet-0e1cbc3d30d0b4732	Name	project_Cafe-subnet-public2-us-east-1b
-----------	--------------------------	------	--

Auto-assign IP settings

Enable AWS to automatically assign a public IPv4 or IPv6 address to a new primary network interface for an instance in this subnet.

- Enable auto-assign public IPv4 address
- Enable auto-assign customer-owned IPv4 address

Option disabled because no customer owned pools found.

Resource-based name (RBN) settings

Specify the hostname type for EC2 instances in this subnet and optional RBN DNS query settings.

- Enable resource name DNS A record on launch
- Enable resource name DNS AAAA record on launch

Hostname type

- Resource name
- IP name

DNS64 settings

Enable DNS64 to allow IPv6-only services in Amazon VPC to communicate with IPv4 endpoints and vice versa.

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:

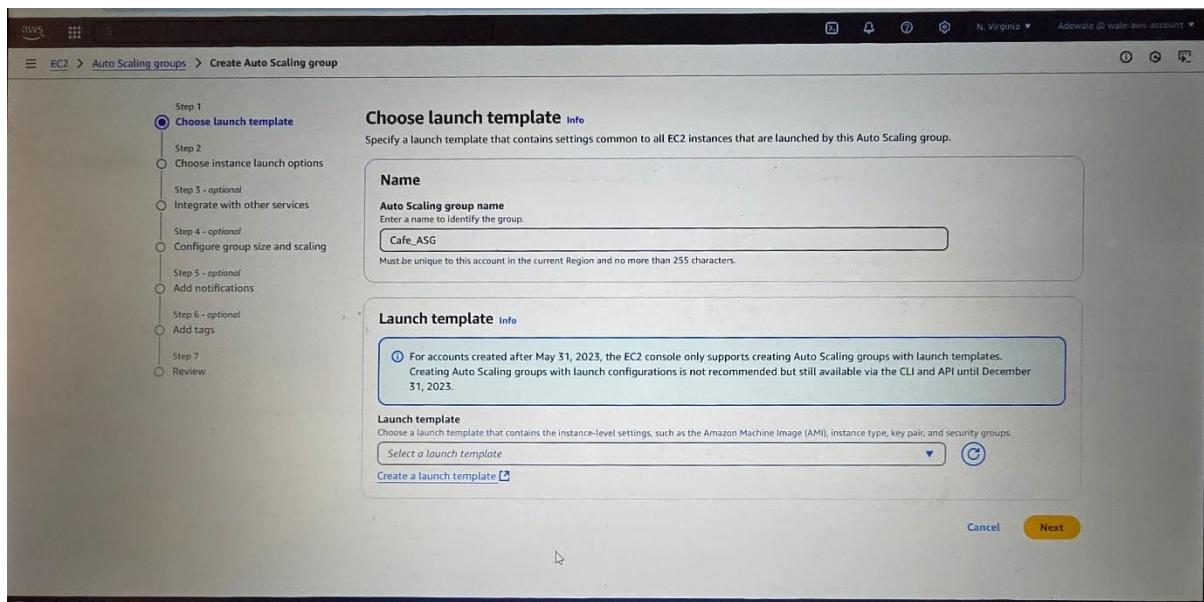
1. Launch EC2 instances in an Auto Scaling Group (ASG). Configure an Application Load Balancer (ALB) to distribute incoming traffic across instances.
2. Define inbound rules in the Security Group to allow HTTP, HTTPS, and SSH access.
3. Use a launch template to automate the configuration and include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



Screenshot of the AWS EC2 Launch Templates - Create launch template page.

Amazon Machine Image (AMI)

Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type
ami-0166fe664262f664c (64-bit (x86)) / ami-07bc5cc4add81dad9 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

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: 664c

Advanced

CloudShell Feedback Privacy Terms Cookie preferences

Screenshot of the AWS EC2 Launch Templates - Create launch template page.

Instance type: 664c

Instance type

t2.micro
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

Free tier eligible

All generations

Compare instance types

Additional costs apply for AMIs with pre-installed software

Key pair (login)

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

Subnet

CloudShell Feedback Privacy Terms Cookie preferences

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.

Network settings

Subnet: Don't include in launch template | Create new subnet

Firewall (security groups): Select existing security group (radio button) | Create security group (radio button selected)

Security group name - required: Cafe_SG

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and _-/.@#=;&:\$*

Description - required: Cafe SSH and HTTP Access

VPC: vpc-0f5550e8024fca793 (project_Cafe-vpc) | 10.0.0.0/16

Inbound Security Group Rules: No security group rules are currently included in this template. Add a new rule to include it in the launch template.

Add security group rule

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

Inbound Security Group Rules

- Security group rule 1 (TCP, 22, 0.0.0.0/0)**
 - Type: ssh | Protocol: TCP | Port range: 22
 - Source type: Anywhere | Source: 0.0.0.0/0
 - Description: e.g. SSH for admin desktop
- Security group rule 2 (TCP, 80, 0.0.0.0/0)**
 - Type: HTTP | Protocol: TCP | Port range: 80
 - Source type: Anywhere | Source: 0.0.0.0/0
 - Description: e.g. SSH for admin desktop
- Security group rule 3 (TCP, 443, 0.0.0.0/0)**
 - Type: HTTPS | Protocol: TCP | Port range: 443
 - Source type: Anywhere | Source: 0.0.0.0/0
 - Description: e.g. SSH for admin desktop

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”.

User data - optional | Info
Upload a file with your user data or enter it in the field.

```
#!/bin/bash

#Update all yum package repositories
yum update -y

#Install Apache Web Server
yum install -y httpd.x86_64

#Start and Enable Apache Web Server
systemctl start httpd.service
systemctl enable httpd.service

#Adds our custom webpage html code to "index.html" file.
echo "<html><body><h1>Welcome to MY CAFE!</h1></body></html>" > /var/www/html/index.html
```

User data has already been base64 encoded

▼ Summary

Software Image (AMI)
Amazon Linux 2 Kernel 5.10 AMI.. [read more](#)
ami-0166fe6e4262f664c

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

Name

Auto Scaling group name
Enter a name to identify the group.

Must be unique to this account in the current Region and no more than 255 characters.

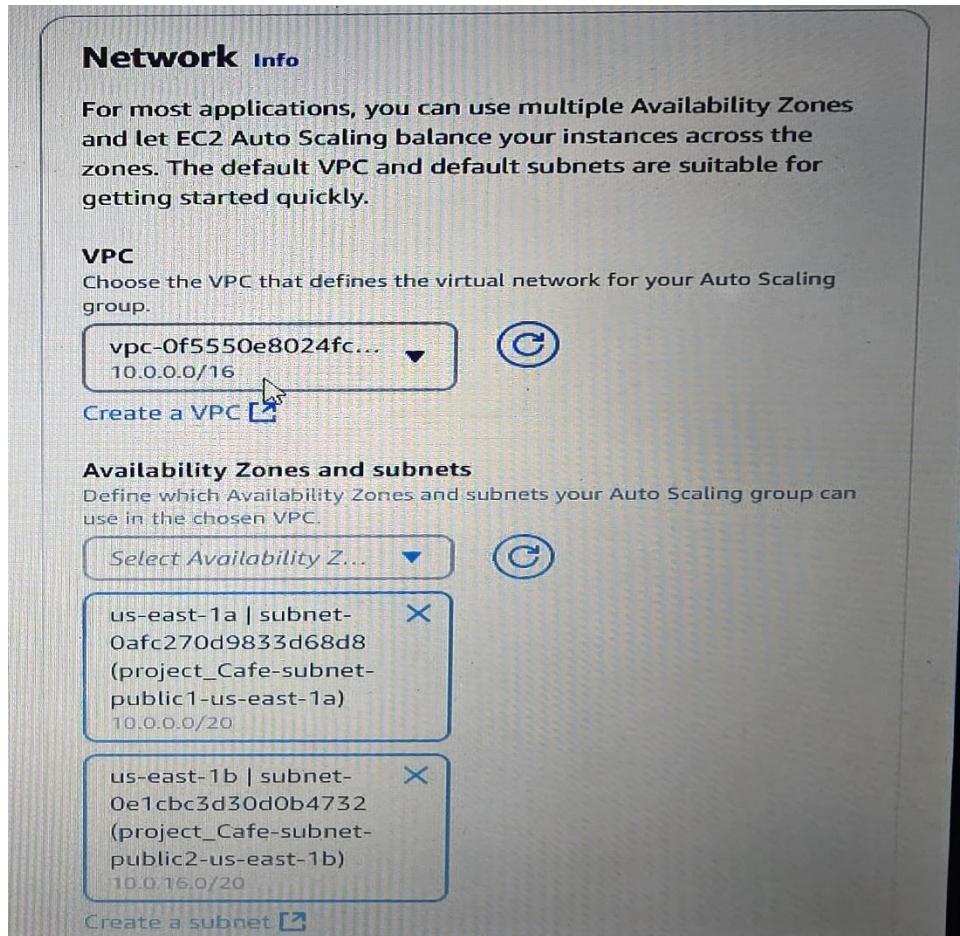
Launch template [Info](#)

For accounts created after May 31, 2023, the EC2 console only supports creating Auto Scaling groups with launch templates. Creating Auto Scaling groups with launch configurations is not recommended but still available via the CLI and API until December 31, 2023.

Launch template
Choose a launch-template that contains the instance-level settings, such as the Amazon Machine Image (AMI), Instance type, key pair, and security groups.
 [Create a launch template](#)

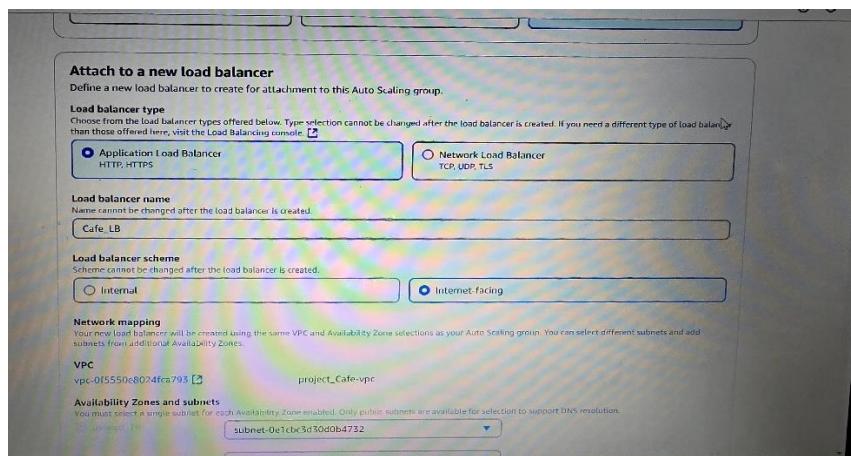
Version
 [Create a launch template version](#)

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.

Availability Zones and subnets
You must select a single subnet for each Availability Zone enabled. Only public subnets are available for selection to support DNS resolution.

us-east-1b subnet-0e1cbc3d30d0b4732
 us-east-1a subnet-0afc270d9833d68d8

Listeners and routing
If you require secure listeners, or multiple listeners, you can configure them from the Load Balancing console after your load balancer is created.

Protocol	Port	Default routing (forward to)
80		Create a target group

New target group name
An instance target group with default settings will be created.
Cafe_LB

Tags - optional
Consider adding tags to your load balancer. Tags enable you to categorize your AWS resources so you can more easily manage them.

Add tag
50 remaining

VPC Lattice integration options Info

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”

Desired capacity
Specify your group size.
2

Scaling Info
You can resize your Auto Scaling group manually or automatically to meet changes in demand.

Scaling limits
Set limits on how much your desired capacity can be increased or decreased.

Min desired capacity	Max desired capacity
2	5

Equal or less than desired capacity Equal or greater than desired capacity

Automatic scaling - optional
Choose whether to use a target tracking policy Info
You can set up other metric-based scaling policies and scheduled scaling after creating your Auto Scaling group.

No scaling policies
Your Auto Scaling group will remain at its initial size and will not dynamically resize to meet demand.

Target tracking scaling policy
Choose a CloudWatch metric and target value and let the scaling policy adjust the desired capacity in proportion to the metric's value.

Scaling policy name
Target Tracking Policy

Metric type Info
Monitored metric that determines if resource utilization is too low or high. If using EC2 metrics, consider enabling detailed monitoring for better scaling performance.

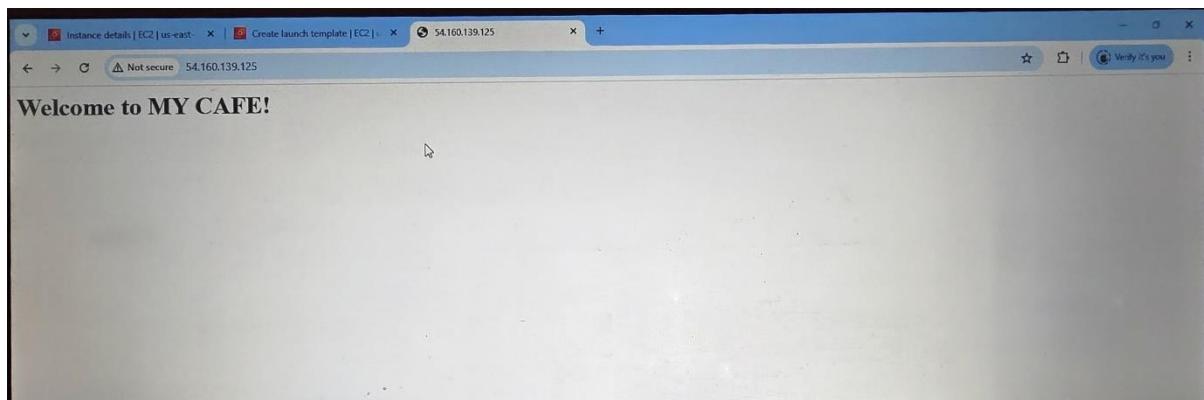
Average CPU utilization

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 in 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.

The screenshot shows the 'Network settings' configuration for a new security group. The 'Security group name - required' field is set to 'AppSG'. The 'Description - required' field contains 'Allow SSH Access and Ping from WebTier'. Under 'VPC', the selected VPC is 'vpc-0f5550e8024fca793 (project_Cafe-vpc)'. In the 'Inbound Security Group Rules' section, there is one rule: 'Security group rule 1 (TCP, 22, sg-0078724ff65a82c37)'. This rule allows traffic from the 'WebTier' security group on port 22.

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.

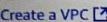
The screenshot shows the 'Choose launch template' step of the AWS EC2 Auto Scaling wizard. The 'Name' field is set to 'cafe_app'. A note at the bottom of the screen states: 'For accounts created after May 31, 2023, the EC2 console only supports creating Auto Scaling groups with launch templates. Creating Auto Scaling groups with launch configurations is not recommended but still available via the CLI and API until December 31, 2023.'

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

For most applications, you can use multiple Availability zones and let EC2 Auto Scaling balance your instances across the zones. The default subnets are suitable for getting started quickly.

VPC
Choose the VPC that defines the virtual network for your Auto Scaling group.

vpc-0f5550e8024fcfa793 (project_Cafe-vpc) ▾ 

Create a VPC 

Availability Zones and subnets
Define which Availability Zones and subnets your Auto Scaling group can use in the chosen VPC.

Select Availability Zones and subnets ▾ 

us-east-1a | subnet-015a3fcfa0d98d62e (project_Cafe-subnet-private1-us-east-1a)
10.0.128.0/20 

us-east-1b | subnet-0bff7515e5ecb65fb (project_Cafe-subnet-private2-us-east-1b)
10.0.144.0/20 

Create a subnet 

Availability Zone distribution - new
Auto Scaling automatically balances instances across Availability Zones. If launch failures occur in a zone, select a strategy.

Balanced best effort
If launches fail in one Availability Zone, Auto Scaling will attempt to launch in another healthy Availability Zone.

Balanced only
If launches fail in one Availability Zone, Auto Scaling will continue to attempt to launch in the unhealthy Availability Zone to preserve balanced distribution.

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.

3 - optional
Integrate with other services

4 - optional
Configure group size and scaling

5 - optional
Notifications

6 - optional
Tags

Load balancing 

Use the options below to attach your Auto Scaling group to an existing load balancer, or to a new load balancer that you define.

No load balancer
Traffic to your Auto Scaling group will not be fronted by a load balancer.

Attach to an existing load balancer
Choose from your existing load balancers.

Attach to a new load balancer
Quickly create a basic load balancer to attach to your Auto Scaling group.

Attach to a new load balancer
Define a new load balancer to create for attachment to this Auto Scaling group.

Load balancer type
Choose from the load balancer types offered below. Type selection cannot be changed after the load balancer is created. If you need a different type of load balancer than those offered here, visit the Load Balancing console.

Application Load Balancer
HTTP, HTTPS

Network Load Balancer
TCP, UDP, TLS

Load balancer name
Name cannot be changed after the load balancer is created.
cafeAppLB

Load balancer scheme
Scheme cannot be changed after the load balancer is created.

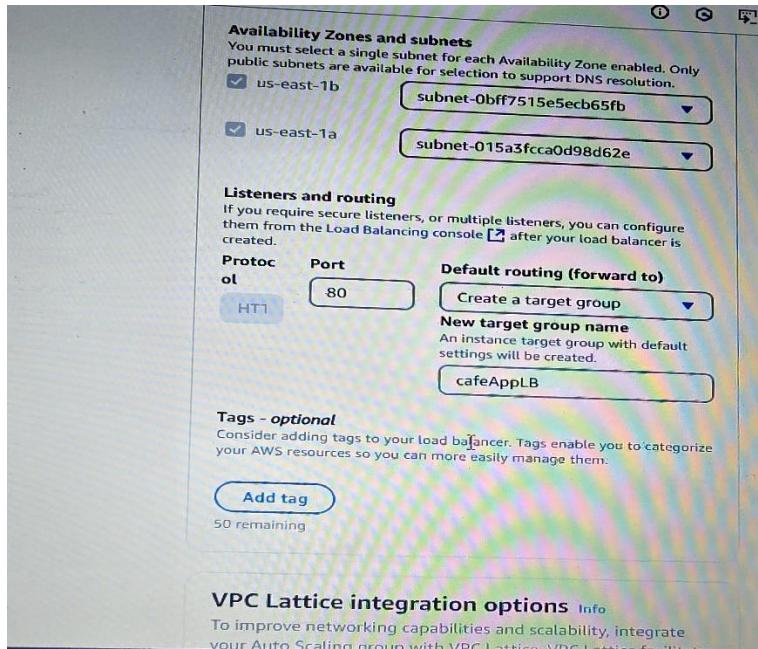
Internal

Internet-facing

Network mapping
Your new load balancer will be created using the same VPC and Availability Zone selections as your Auto Scaling group. You can select different subnets and add subnets from additional Availability Zones.

VPC
vpc-0f5550e8024fcfa793 

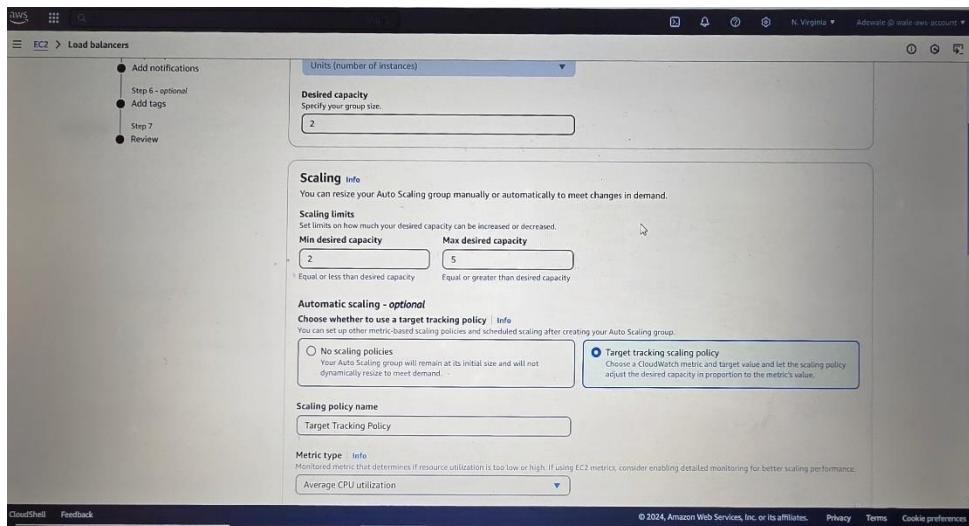
Make sure your VPC and both private subnets are selected. For “**Listeners and routing**”, select “**Create a target group**” and choose our Application Tier load balancer.



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 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.

The screenshot shows the AWS Auto Scaling Groups page. At the top, there is a success message: "cafe_App, 1 Scaling policy, 1 Load balancer, 1 Target group, 1 Listener created successfully. 1 new target group has been attached to ASG." Below this, the heading "Auto Scaling groups (2)" is followed by a table. The first row shows "cafe_App" with "Cafe_app_temp | Version Default" and 0 instances. The second row shows "Cafe_ASG" with "Cafe_Temp | Version Default" and 2 instances. A search bar and filter buttons for Name, Launch template/configuration, and Instances are visible above the table. A button labeled "Create Auto Scaling group" is located at the top left of the main content area.

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.

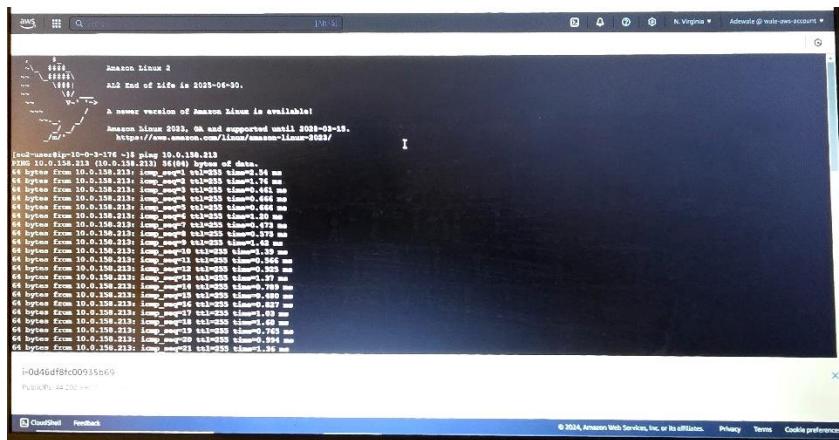
The screenshot shows the AWS Instances page. The heading "Instances (1/4)" is followed by a table. The table has columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4 DNS. There are four rows, each representing an EC2 instance. The first instance is checked and has the ID i-07360e37bd249ee6a, state Running, type t2.micro, status 2/2 checks passed, alarm status View alarms +, availability zone us-east-1b, and public IP ec2-54-160-139-125. The other three instances have IDs i-0978cd7e739b23206, i-0bbea951d8b536ae8, and i-0d46df8fc00935b69, respectively, with similar details. A search bar and filter buttons are at the top of the table.

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 your 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_ip>
```

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

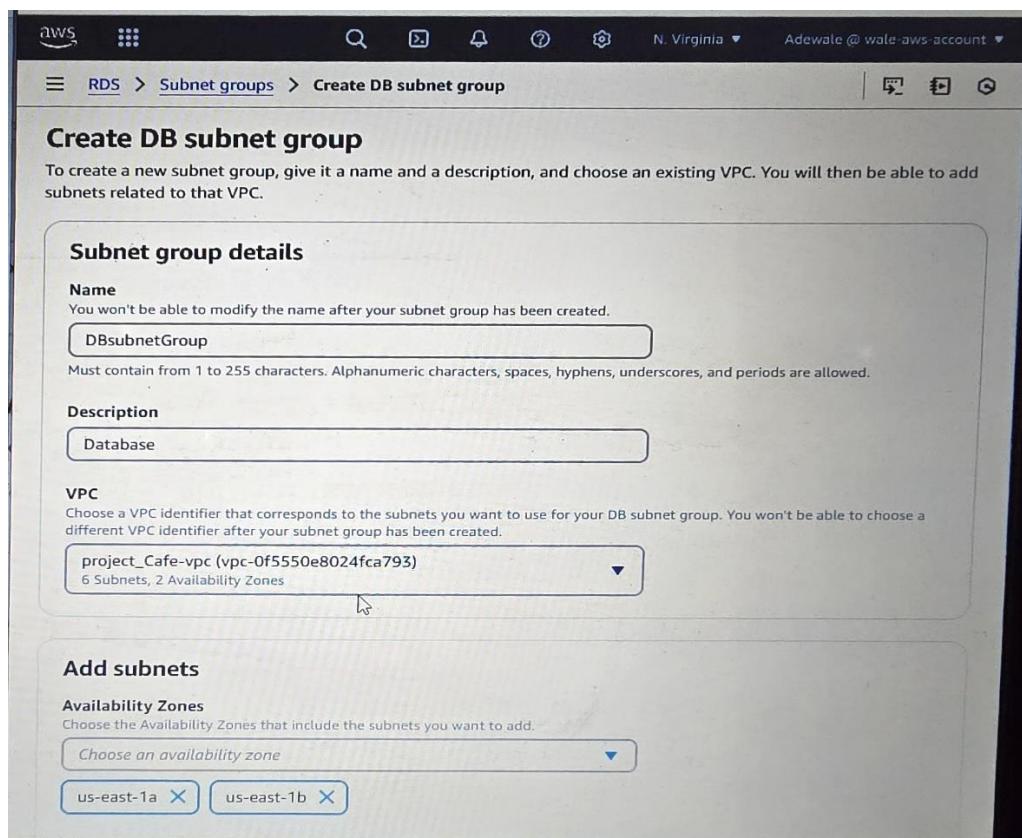
To store and process data:

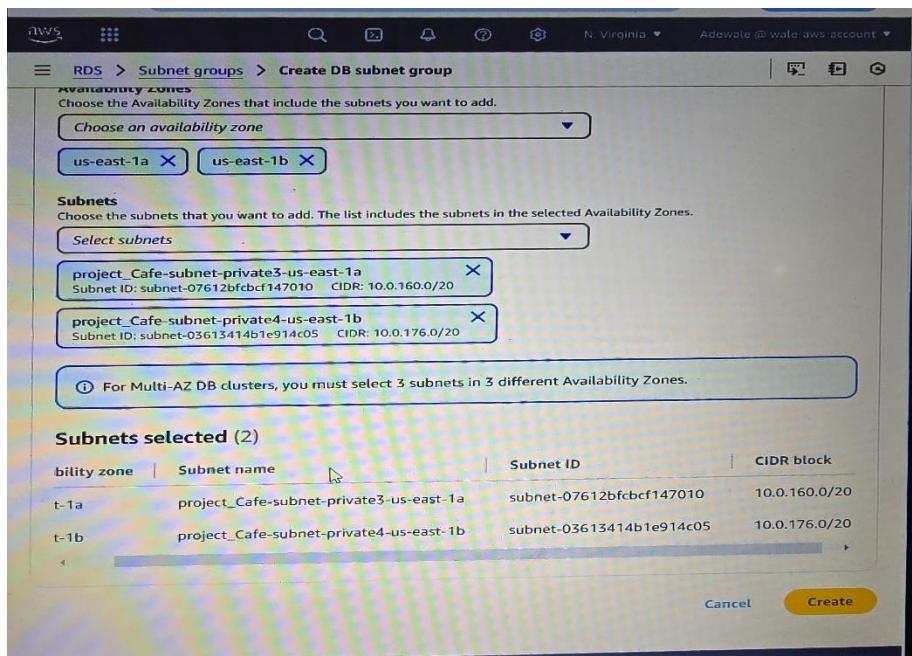
1. Create a MySQL database using Amazon RDS in private subnets with Multi-AZ deployment for high availability.
2. Configure a DB subnet group and adjust inbound rules to permit access only from the Application Tier.

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.

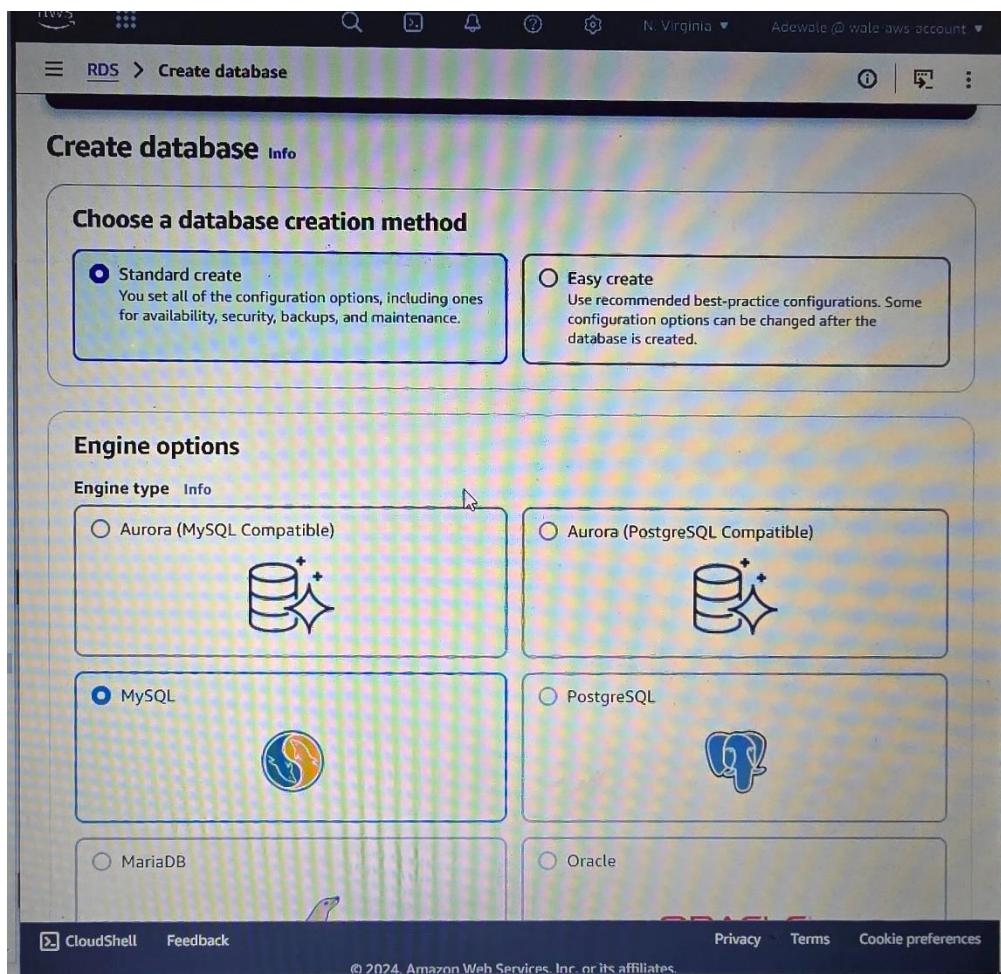
For higher availability and data redundancy, I will choose a Multi-AZ DB Instance deployment. To achieve this, choose your two AZs, then make sure to select the remaining two private subnets that has not been used to host our DB Instances, the create the group.



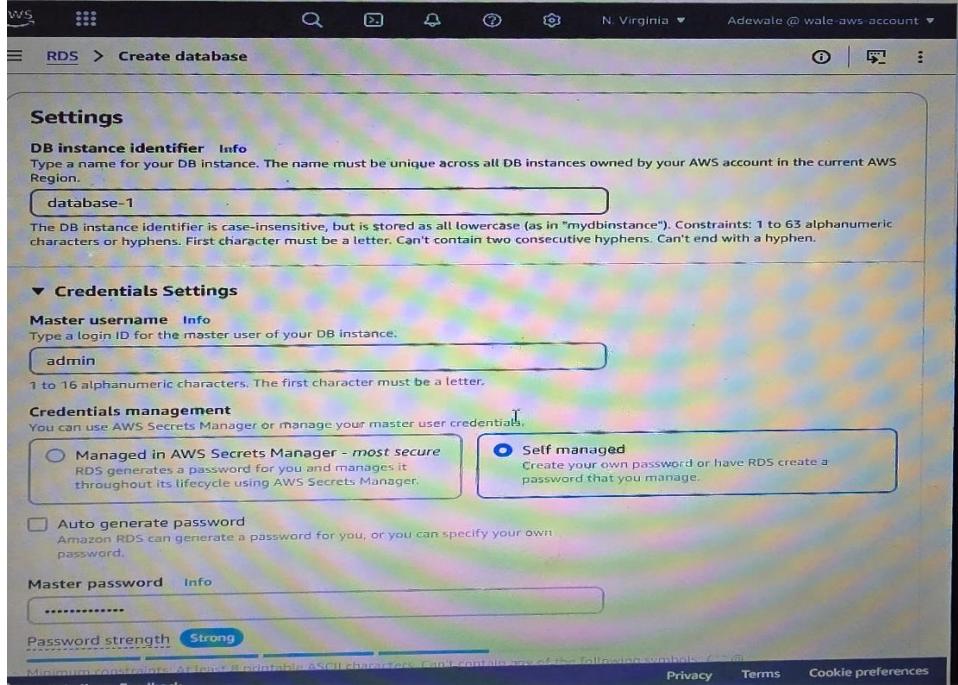


Navigate back to the RDS dashboard, click “**Databases**”, then “**Create database**”.

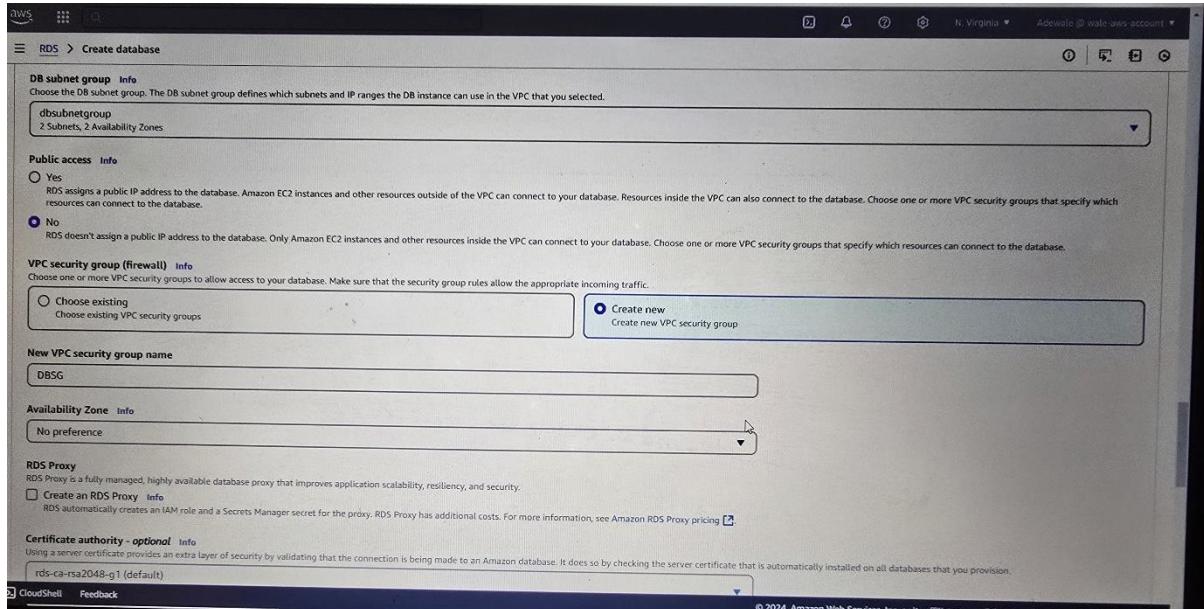
Select “**Standard create**” and “**MySQL**”.



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.

The screenshot shows the AWS RDS console with the path: RDS > Databases > database-1. The "Connectivity & security" tab is selected. The interface is divided into three main sections: Endpoint & port, Networking, and Security.

- Endpoint & port:** Shows the endpoint and port details.
- Networking:** Shows the VPC, Subnet group, Subnets, and Network type (IPv4).
- Security:** Shows the VPC security groups (DBSG), which is active, and the publicly accessible status (No). It also displays the certificate authority information.

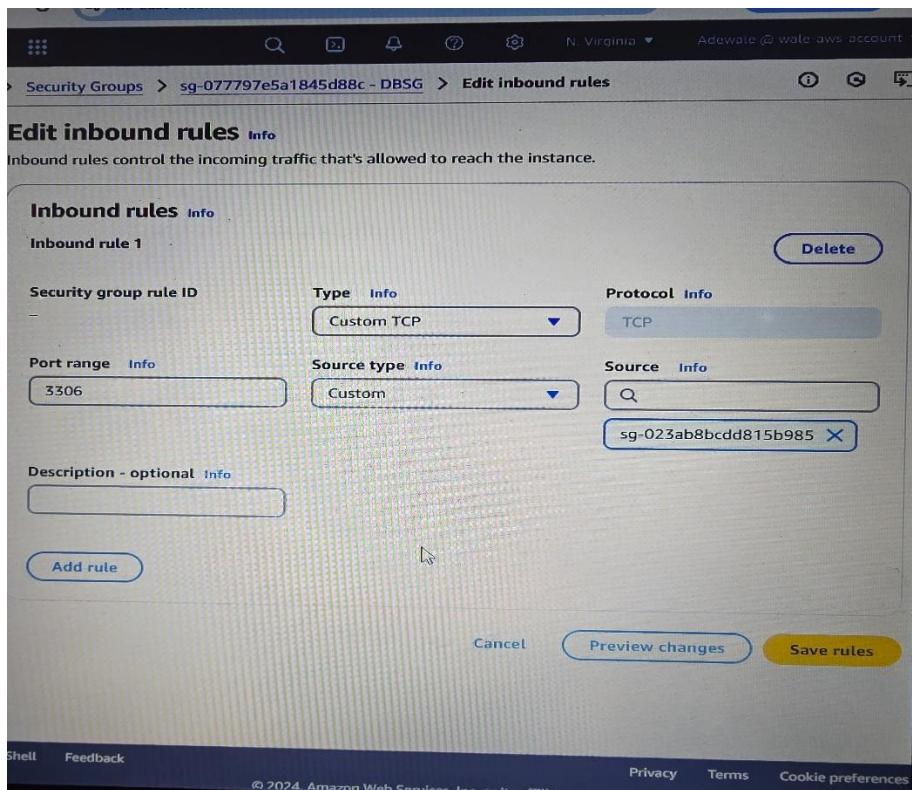
At the bottom, there is a section for "Connected compute resources" with a note about automatically created connections.

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 range — 3306, then add the Application Tier security group. You can now, delete the old rule.

The screenshot shows the AWS Security Groups list with one item: "sg-077797e5a1845d88c". The table has columns for Name, Security group ID, and Security group name. The "Actions" button is visible at the top left of the list.

Name	Security group ID	Security group name
sg-077797e5a1845d88c	sg-077797e5a1845d88c	DBSG



Success!

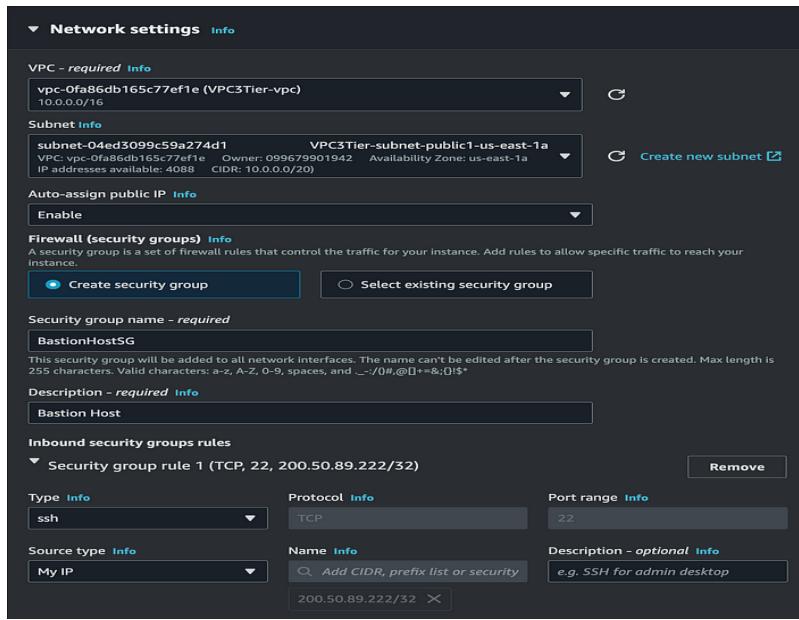
We've successfully built the Database Tier! Let's **proceed to Step 5: Creating a Bastion Host**.

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.micro 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 SHA256:5k5tHvsgP3RuuJPcCO7ntv/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:Azvaktv1QtwsLU+LJ4qBPXPeGu4zr7hIayUZLtt4Nc.
ECDSA key fingerprint is MD5:3d:a9:5c:4a:al: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