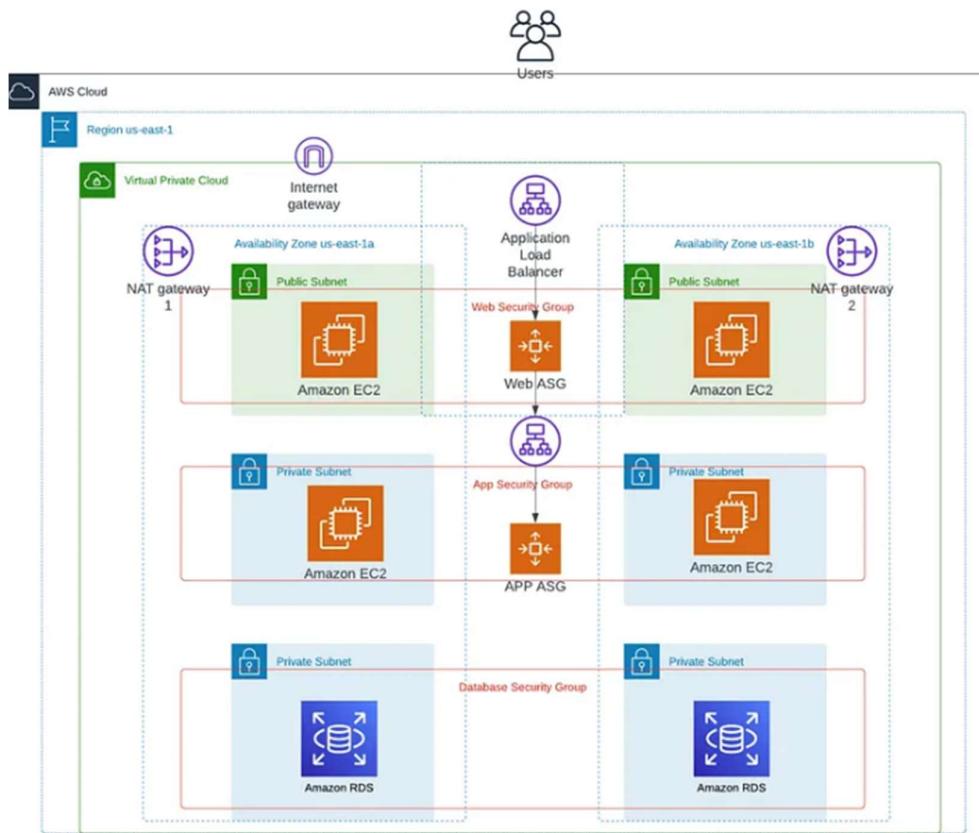


# AWS 3-Tier Architecture: Scalable and Highly Available Web Application Deployment.

(by - Nikhil Mahesh Shebannavar)



## Introduction

Deploying web applications with high availability, scalability, and security is a critical requirement in modern cloud infrastructure. This project demonstrates the implementation of a 3-Tier Architecture on Amazon Web Services (AWS) to host a web application in a production-ready environment.

The 3-Tier model is a widely adopted architectural pattern that separates application components into three distinct layers:

- Presentation Tier – the frontend, handling user interaction.
- Application Tier – the business logic or backend processing.
- Database Tier – the data layer that stores and manages application data.

This architecture was deployed across multiple Availability Zones (AZs) in AWS to ensure high availability, with components such as Elastic Load Balancers (ELBs), Auto Scaling Groups (ASGs), RDS (Relational Database Service), and secure networking using VPCs.

By implementing this solution, the project aims to simulate real-world cloud infrastructure practices suitable for hosting modern, scalable, and fault-tolerant web applications.

## **Why 3 tier architecture**

The 3-tier architecture is a widely adopted design pattern in web application development and cloud deployments due to its modularity, scalability, and security benefits. It divides the application into three separate layers, each responsible for specific functions:

### **1. Presentation Tier (Frontend)**

This is the user interface layer — typically a web browser or client app that interacts with users. Hosting this tier separately allows independent scaling based on user traffic.

### **2. Application Tier (Logic Layer)**

This tier handles the core functionality and business logic. It processes user inputs, manages sessions, and communicates between the frontend and database. Isolating logic makes the application easier to maintain and scale.

### **3. Database Tier (Data Layer)**

Responsible for storing, retrieving, and managing data. It ensures data integrity, security, and availability. Keeping the database separate improves security and performance.

## **Task 1: Creating a VPC and Subnets**

Using the architecture diagram as a reference, I have created new VPC with 2 public subnets and 4 private subnets.

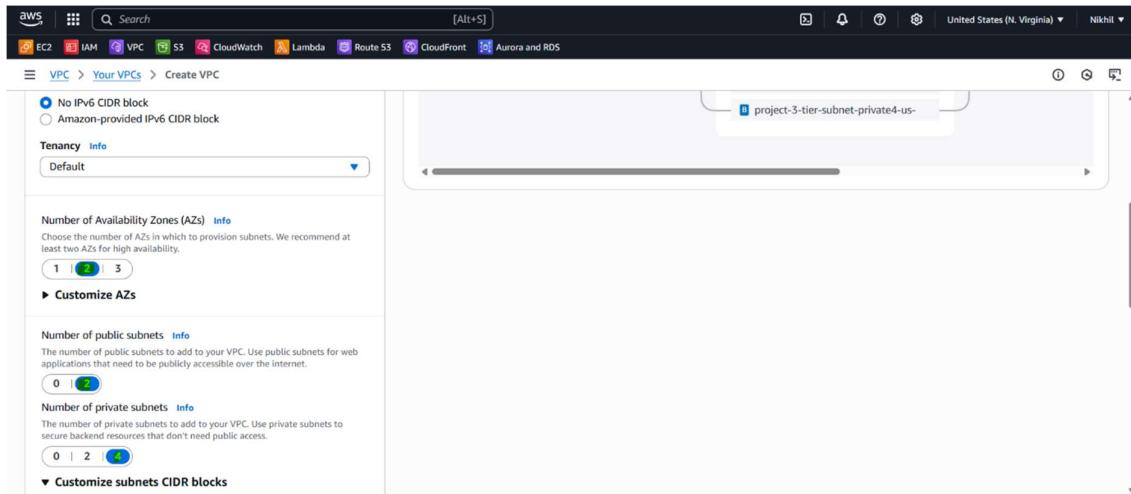
In VPC dashboard click on Create VPC.

The screenshot shows the AWS VPC Dashboard. On the left, there's a sidebar for 'Virtual private cloud' with options like 'Your VPCs', 'Subnets', 'Route tables', etc. The main area displays 'Resources by Region' for N. Virginia, including 'VPCs' (1), 'NAT Gateways' (0), 'Subnets' (6), 'VPC Peering Connections' (0), 'Route Tables' (1), 'Network ACLs' (1), 'Internet Gateways' (1), and 'Security Groups' (1). On the right, there are sections for 'Service Health', 'Settings' (with 'Block Public Access', 'Zones', 'Console Experiments'), and 'Additional Information' (with links to 'VPC Documentation', 'All VPC Resources', 'Forums', and 'Report an Issue').

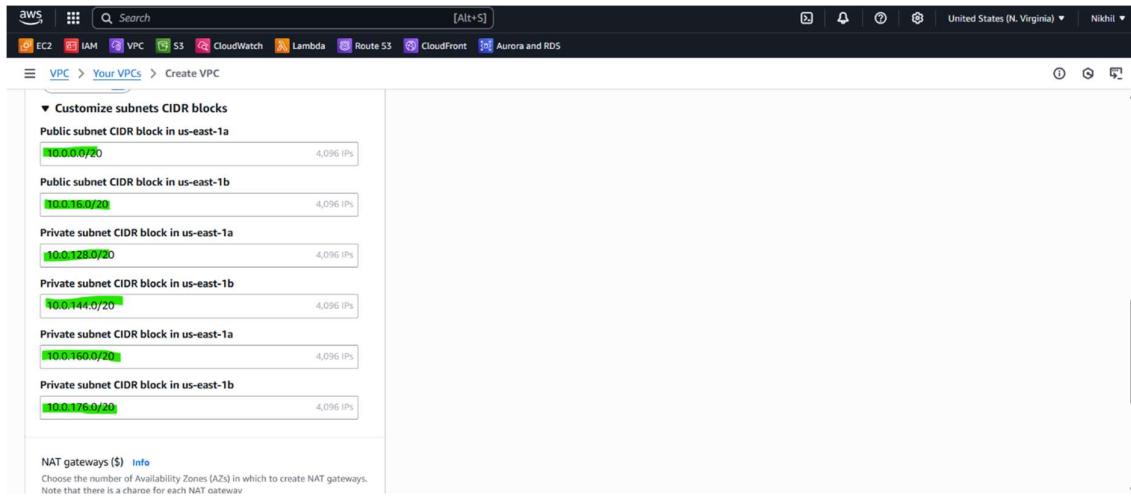
Select VPC and more, name the VPC, IPV4 CIDR block I have taken starting IP and size as 10.0.0.0/16 and IPV6 is not needed.

The screenshot shows the 'Create VPC' wizard. In the 'VPC settings' section, under 'Resources to create', 'VPC only' is selected. Under 'Name tag auto-generation', 'Auto-generate' is checked, and the name 'project-3-tier' is entered. Under 'IPv4 CIDR block', the starting IP is '10.0.0.0/16' and the size is '65,536 IPs'. Under 'IPv6 CIDR block', 'No IPv6 CIDR block' is selected. Under 'Tenancy', 'Default' is chosen. In the 'Preview' section, it shows a single VPC named 'project-3-tier-vpc' with 6 subnets across two AZs ('us-east-1a' and 'us-east-1b'). Each AZ has three subnets: 'project-3-tier-subnet-public1-us-', 'project-3-tier-subnet-private1-us-', and 'project-3-tier-subnet-private3-us-' in 'us-east-1a', and 'project-3-tier-subnet-public2-us-', 'project-3-tier-subnet-private2-us-', and 'project-3-tier-subnet-private4-us-' in 'us-east-1b'. Each subnet is associated with a specific route table.

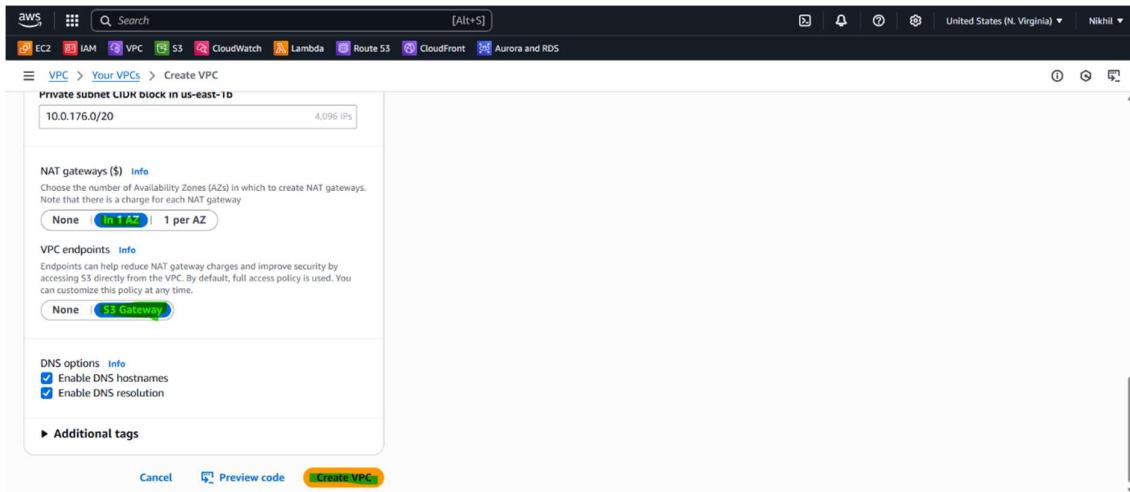
Select 2 AZs, Number of public subnets as 2 and Number of private subnets as 4



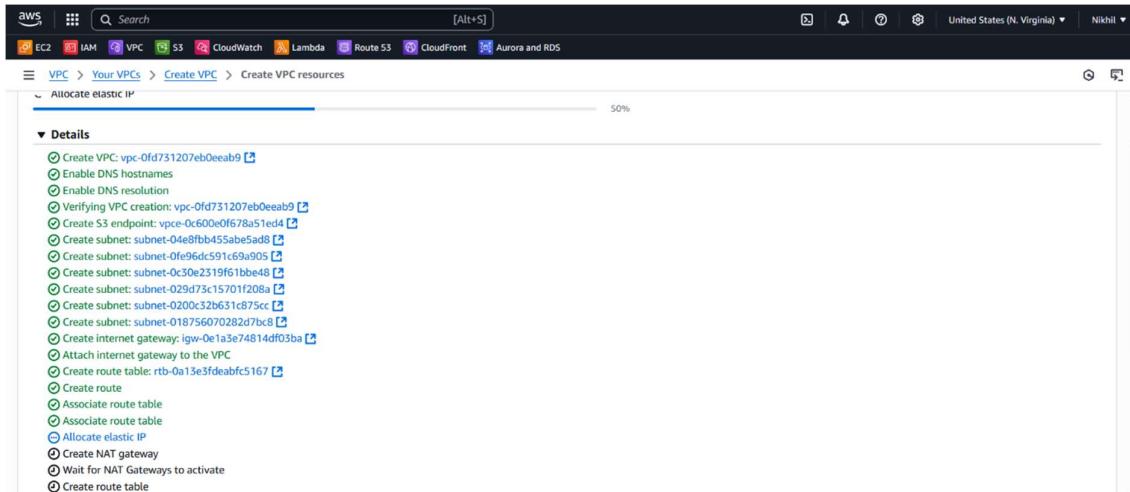
In customize subnet I have given 6 different IP for all subnets.



NAT gateway in 1 AZ. Click on create VPC



VPC is creating.



VPC with 2 public and 4 private subnets are created, also we can check resource map.

VPC dashboard < VPC

**vpc-0fd731207eb0eeab9 / project-3-tier-vpc**

**Details** **info**

VPC ID: vpc-0fd731207eb0eeab9 | State: Available | DNS resolution: Enabled | Main network ACL: acl-095e8279e11870f7 | IPv6 CIDR (Network border group): -

Tenancy: default | Default VPC: No | Network Address Usage metrics: Disabled | Block Public Access: Off | DHCP option set: dosp-095e8279e11870f7c | IPv4 CIDR: 10.0.0.0/16 | Route 53 Resolver DNS Firewall rule groups: -

DNS hostnames: Enabled | Main route table: rtb-0c50819081606690 | IPv6 pool: - | Owner ID: 285013783190

**Resource map** **info**

**VPC** Show details Your AWS virtual network project-3-tier-vpc

**Subnets (6)** Subnets within this VPC

- us-east-1a
  - project-3-tier-subnet-public1-us-east-1a
  - project-3-tier-subnet-private1-us-east-1a
  - project-3-tier-subnet-private3-us-east-1a
  - project-3-tier-subnet-private4-us-east-1b
  - project-3-tier-subnet-public1-us-east-1b
  - project-3-tier-subnet-private2-us-east-1b
- us-east-1b
  - project-3-tier-subnet-public2-us-east-1a
  - project-3-tier-subnet-private2-us-east-1b
  - project-3-tier-subnet-private3-us-east-1a

**Routes (6)** Route network traffic to resources

- rtb-0c50819081606690
  - project-3-tier-rtb-public
  - project-3-tier-rtb-private2-us-east-1b
  - project-3-tier-rtb-private4-us-east-1b
  - project-3-tier-rtb-private1-us-east-1a
  - project-3-tier-rtb-private3-us-east-1a
  - project-3-tier-rtb-private2-us-east-1a

**Network connections (3)** Connections to other networks

- project-3-tier-ligu
- project-3-tier-nat-public1-us-east-1a
- project-3-tier-vpc-s3

Now I changed the subnet setting to enable auto generate IPs, to do click on subnets in VPC dashboard, select one subnet, click on actions and click edit subnet settings

VPC dashboard < Subnets

**Subnets (1/6) Info**

Name	Subnet ID	State	VPC
project-3-tier-subnet-private1-us-east-1a	subnet-0c30e2319f61bbe48	Available	vpc-0fd731207eb
project-3-tier-subnet-private2-us-east-1b	subnet-029d73c15701f208a	Available	vpc-0fd731207eb
project-3-tier-subnet-private3-us-east-1a	subnet-0200c32b631c875cc	Available	vpc-0fd731207eb
project-3-tier-subnet-private4-us-east-1b	subnet-01875607028d7bc8	Available	vpc-0fd731207eb
project-3-tier-subnet-public1-us-east-1a	subnet-04e8fb4455abe5ad8	Available	vpc-0fd731207eb
project-3-tier-subnet-public2-us-east-1b	subnet-0fe96dc591c69a905	Available	vpc-0fd731207eb

**subnet-0c30e2319f61bbe48 / project-3-tier-subnet-private1-us-east-1a**

**Details** Flow logs Route table Network ACL CIDR reservations Sharing Tags

**Details**

Last updated 2 minutes ago | Actions | Create subnet | View details | Create flow log | Edit IPv4 CIDR | Edit IPv6 CIDRs | Edit network ACL association | Edit route table association | Edit CIDR reservations | Share subnet | Manage tags | Delete subnet

IPv4 CIDR: 10.0.128.0/20, 10.0.144.0/20, 10.0.160.0/20, 10.0.176.0/20, 10.0.0.0/20, 10.0.16.0/20

Now, Enable auto assign public IPv4 address and save. I have done this to all 6 subnets.

**Subnet**

Subnet ID: subnet-0c30e2319f61bbe48

Name: project-3-tier-subnet-private1-us-east-1a

**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

Check the route table if it is associated with correct subnets.

**Route tables (1/7)**

Name	Route table ID	Explicit subnet associations	Main	VPC
-	rtb-012cccc6173cf7ec9	-	Yes	vpc-06faf9226a78d2341
-	rtb-0c60d819081606690	-	Yes	vpc-0fd731207eb0eeab91f
<input checked="" type="checkbox"/> project-3-tier-rtb-public	rtb-0a13e3fdeabfc5167	2 subnets	No	vpc-0fd731207eb0eeab91f
project-3-tier-rtb-private2-us-east-1b	rtb-026b30f1f7f7014f4	subnet-029d73c15701f2...	No	vpc-0fd731207eb0eeab91f

**rtb-0a13e3fdeabfc5167 / project-3-tier-rtb-public**

**Routes (2)**

Destination	Target	Status	Propagated
0.0.0.0/0	local	Active	No
10.0.0.0/16	local	Active	No

## Task 2: Creating a Web Server Tier

Next, I have created EC2 Instance for first tier i.e web tier(front end), then I have created an auto scaling group of EC2 instance that will host webpage. Starting with creating EC2 instance.

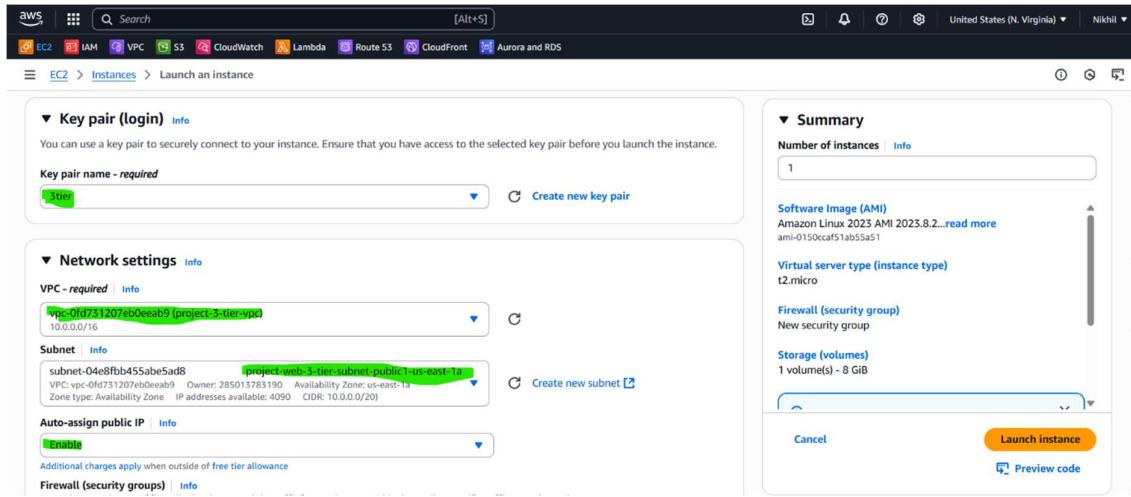
Go to EC2 dashboard, click on Launch Instance.

The screenshot shows the AWS EC2 dashboard with the 'Launch instance' section highlighted. The 'Resources' sidebar on the left lists various EC2-related services. The main area shows a summary of current resources and a 'Launch instance' button. To the right, there's a 'Service health' section and a 'Offer usage (monthly)' section displaying metrics like EBS Snapshot Usage at 100% and Storage space on EBS at 53%.

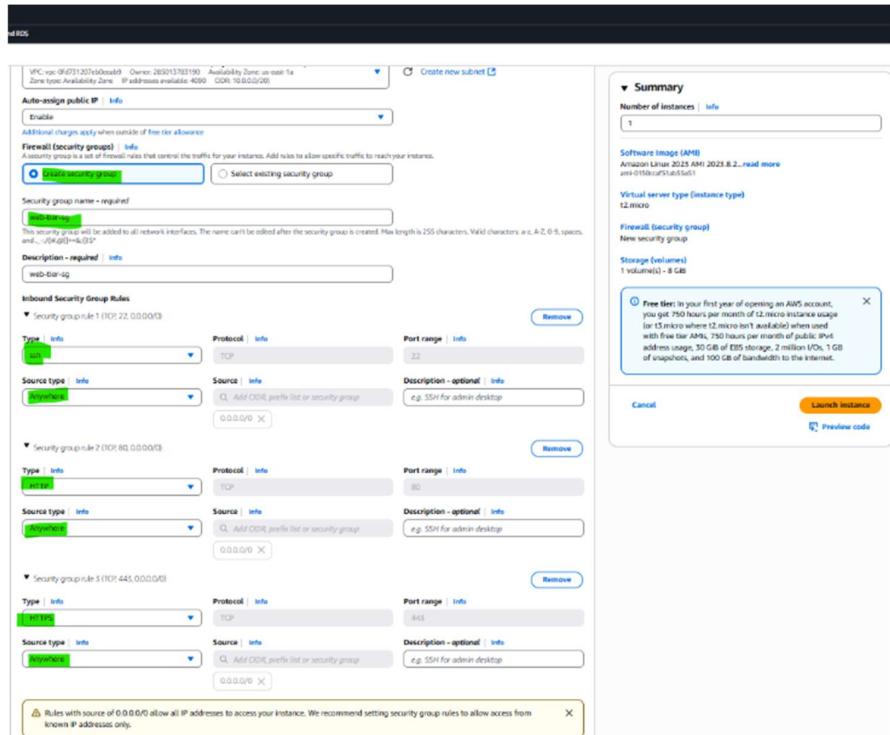
Named my instance, selected an AMI(I have selected Amazon Linux updated version) and instance type as T2.micro.

This screenshot captures the 'Launch an instance' wizard. It starts with the 'Name and tags' step, where 't2.micro' is chosen. The next step, 'Application and OS Images (Amazon Machine Image)', shows the selection of 'Amazon Linux 2023.8.20230707.0 x86\_64 HVM kernel-6.1'. The 'Quick Start' section details the instance configuration, including architecture (64-bit x86), boot mode (UEFI preferred), AMI ID, publish date, and a note about verified packages. The 'Summary' step shows one instance selected. A tooltip for the instance type provides information about free tier usage, stating: 'Free tier in your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMI. 750 hours per month of public IPv4 address, 100 GB of storage, 2000 requests per second, 1000 snapshots, and 100 GB of bandwidth to the internet.'

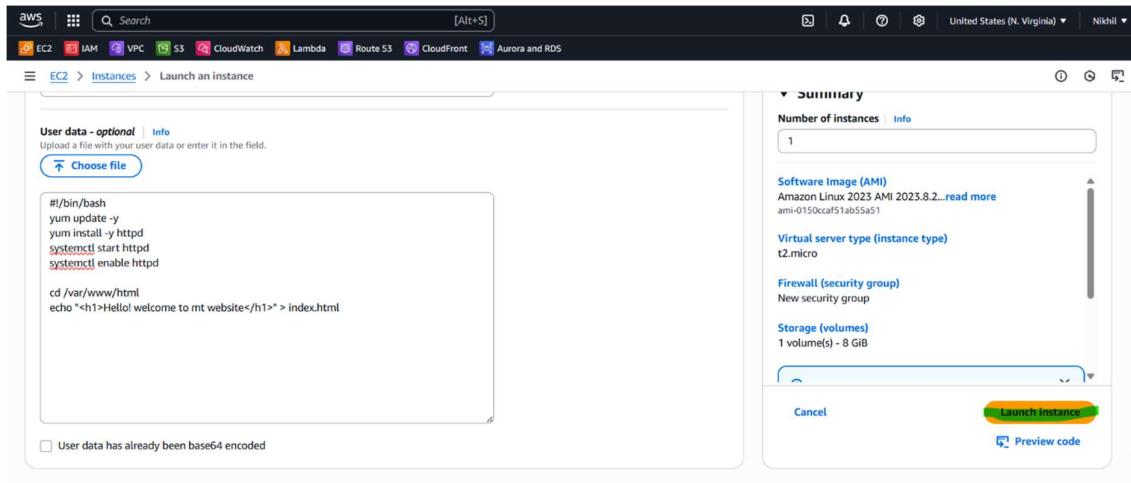
Select the key pair or create new one, select the new VPC and the correct subnet(public) and enable auto assign IP.



Create a new security group. For inbound security groups rules, add rules for ssh, HTTP, HTTPS from everywhere(not safe but for demo project it is fine).



I have not changed anything for storage settings, Now, In the advanced detailed, at bottom, I ran the script to launch an Apache web server when instance starts. Click on launch instance

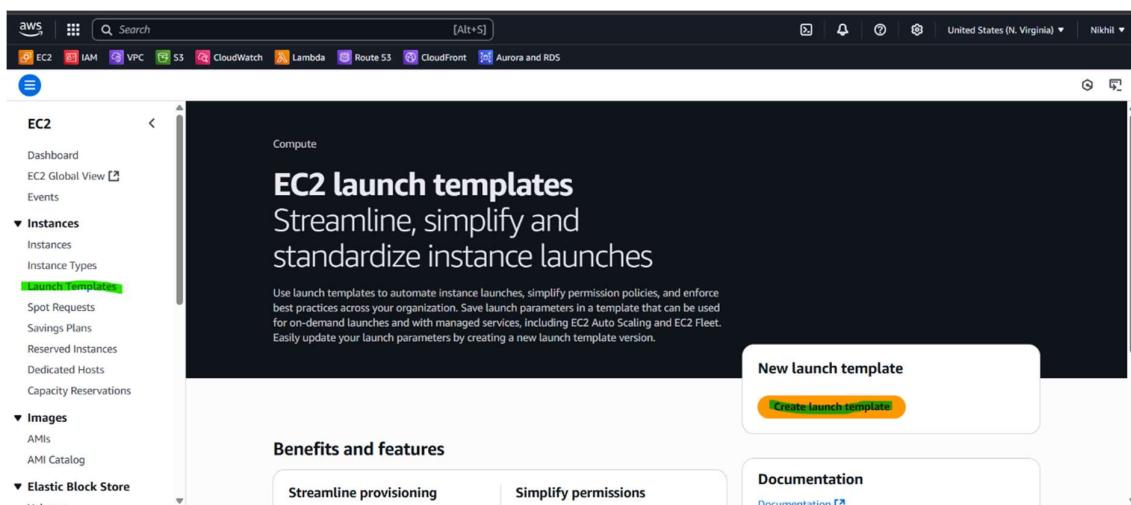


Once I launched the EC2 instance, copied the Public IP and Pasted on web server.

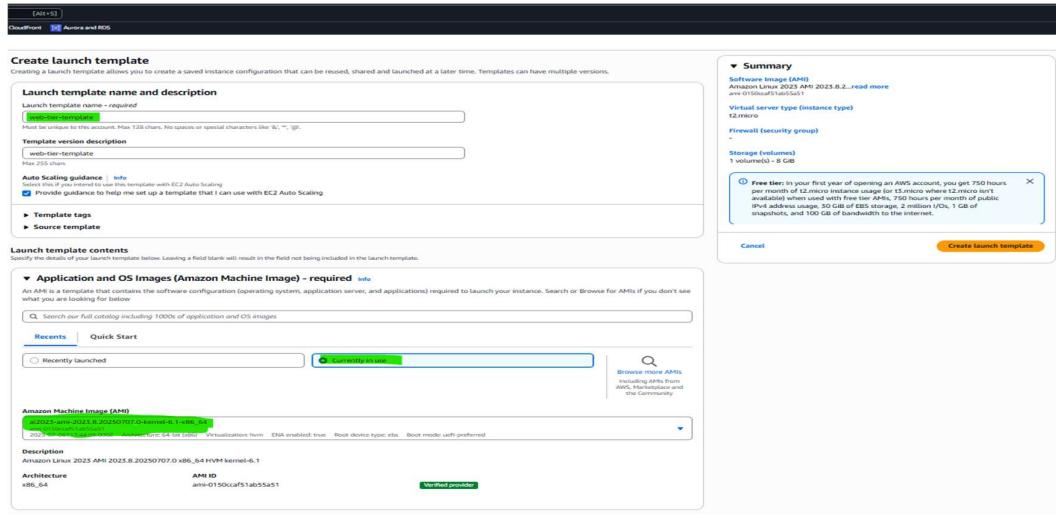


I can see that website is created in Apache server and It's working fine.

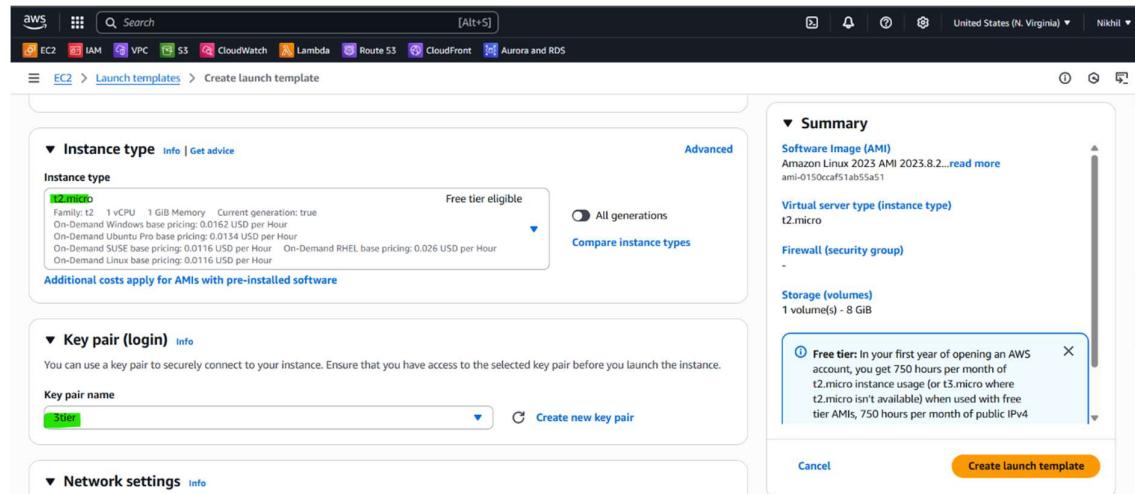
This project needs auto scaling group that I will attach to EC2 instance. This will increase reliability and availability. Next, I have created a launch template. Under EC2 dashboard, select launch templates, and click create launch template.



Name the template, and check the box to provide guidance.



Used recently launched AMI T2.micro instance type and selected key pair.



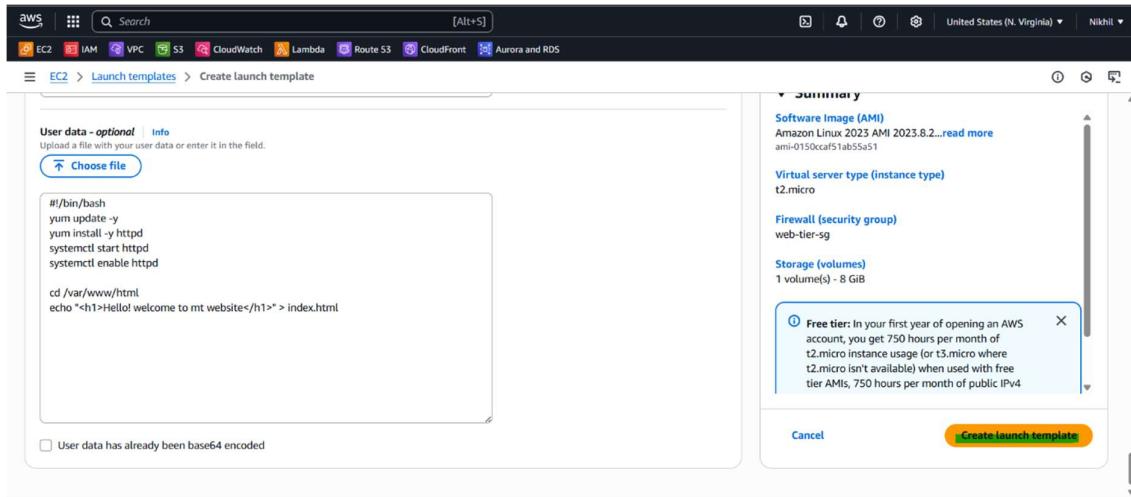
## Selected existing security group (web tier sg).

The screenshot shows the AWS EC2 Launch Templates interface. In the 'Network settings' section, under 'Security groups', the 'Select security groups' dropdown is open, showing 'web-tier-sg sg-07fe093eaf27cd1b'. This group is highlighted with a blue border. Below this, the 'Advanced network configuration' section is partially visible.

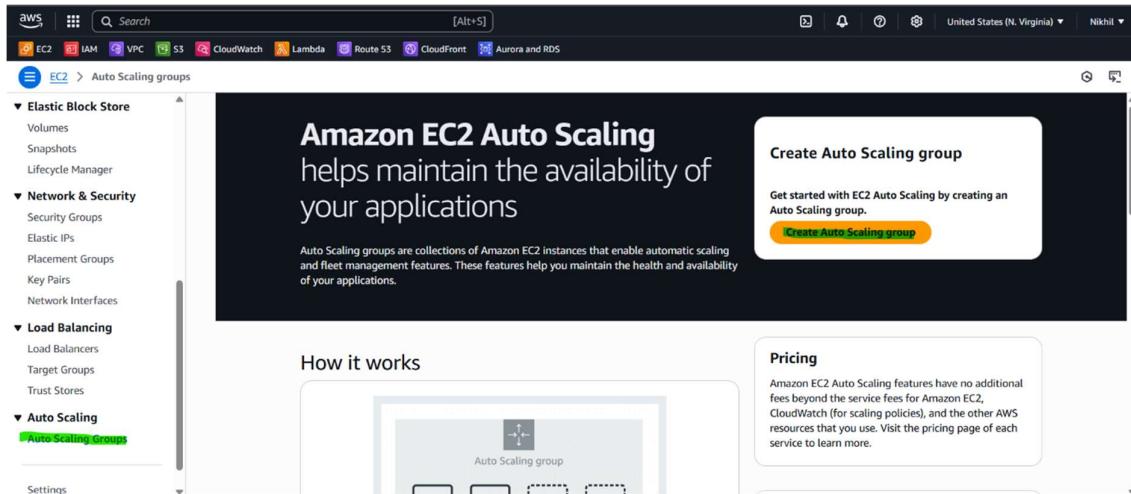
Under advanced network configuration, enable auto assign public IP.

The screenshot shows the 'Advanced network configuration' section of the launch template editor. Under 'Network interface 1', the 'Auto-assign public IP' checkbox is checked and highlighted with a green border. A tooltip on this field states: 'Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMIs, 750 hours per month of public IPv4 address usage, 30 GiB of EBS storage, 2 million I/Os, 1 GiB of snapshots, and 100 GiB of bandwidth to the internet.' Other configuration options like 'Primary IP', 'IPv4 Prefixes', and 'ENAs' are also visible but not selected.

Click on the advanced details, at bottom enter the same script as I did earlier for EC2 instance, then click create launch template.



Now, I created auto scaling group. Click create auto scaling group,



Name the ASG, choose the launch template that created just now, then click next.

Under Network, select VPC that created earlier, under availability zones and subnets select the public subnets that created , select Balanced best effort click next.

Now, click on attach a new load balancer, select Application load balancer, name the load balancer, selected internet facing, VPC and subnets are already be selected. Under the listeners and routing section select create a target group which should be on port 80 for HTTP traffic.

**Step 1**  
Choose launch template  
**Step 2**  
Choose instance launch options  
**Step 3 - optional**  
**Integrate with other services**  
**Step 4 - optional**  
Configure group size and scaling  
**Step 5 - optional**  
Add notifications  
**Step 6 - optional**  
Add tags  
**Step 7**  
Review

**Integrate with other services - optional** Info

Use a load balancer to distribute network traffic across multiple servers. Enable service-to-service communications with VPC Lattice. Shift resources away from impaired Availability Zones with zonal shift. You can also customize health check replacements and monitoring.

**Load balancing** Info

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  
Create a new Application 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. (+) View details

Application Load Balancer  
HTTP, HTTPS  
 Network Load Balancer  
TCP, UDP, TLS

**Load balancer name**  
Name cannot be changed after the load balancer is created.  
web-tier-asg-lb

**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-0fd731207eb0eefab9 (+) View details project-3-tier-vpc

**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.

use1-az2 (us-east-1b)	subnet-0f6e0d4991e69a005
use1-az1 (us-east-1a)	subnet-0246fb0435a1e000

**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)
HTTP	80	<input checked="" type="radio"/> Create target group New target group name An instance target group with default settings will be created. web-tier-asg-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.

Leave no VPC lattice service checked.

Click to turn on elastic load balancing health checks.

I have entered health check grace period as 50.

Click next.

Select VPC Lattice service to attach

No VPC Lattice service  
VPC Lattice will not manage your Auto Scaling group's network access and connectivity with other services.

Attach to VPC Lattice service  
Incoming requests associated with specified VPC Lattice target groups will be routed to your Auto Scaling group.

Create new VPC Lattice service [\[+\]](#)

**Application Recovery Controller (ARC) zonal shift - new** [Info](#)  
During an Availability Zone impairment, target instance launches towards other healthy Availability Zones.

Enable zonal shift  
New instance launches will be retargeted towards healthy Availability Zones until the zonal shift is canceled.

**Health checks**  
Health checks increase availability by replacing unhealthy instances. When you use multiple health checks, all are evaluated, and if at least one fails, instance replacement occurs.

**EC2 health checks**  
 Always enabled

**Additional health check types - optional** [Info](#)  
 Turn on Elastic Load Balancing health checks [\[?\]](#) **Recommended**  
Elastic Load Balancing monitors whether instances are available to handle requests. When it reports an unhealthy instance, EC2 Auto Scaling can replace it on its next periodic check.

**EC2 Auto Scaling will start to detect and act on health checks performed by Elastic Load Balancing. To avoid unexpected terminations, first verify the settings of these health checks** [\[x\]](#)  
in the [Load Balancer console](#) [\[+\]](#)

Turn on VPC Lattice health checks  
VPC Lattice can monitor whether instances are available to handle requests. If it considers a target as failed a health check, EC2 Auto Scaling replaces it after its next periodic check.

Turn on Amazon EBS health checks  
EBS monitors whether an instance's root volume or attached volume stalls. When it reports an unhealthy volume, EC2 Auto Scaling can replace the instance on its next periodic health check.

**Health check grace period** [Info](#)  
This time period delays the first health check until your instances finish initializing. It doesn't prevent an instance from terminating when placed into a non-running state.  
 seconds

[Cancel](#) [Skip to review](#) [Previous](#) [Next](#)

In configure group size and scaling, I have selected desired capacity as 2, Minimum capacity as 2, Maximum capacity as 3, In automatic scaling selected Target tracking scaling policy, Metric type as Average CPU utilization, Target value as 50, Instances need as 50.

Step 1  Choose launch template  
Step 2  Choose instance launch options  
Step 3 - optional  Integrate with other services  
Step 4 - optional  Configure group size and scaling  
Step 5 - optional  Add notifications  
Step 6 - optional  Add tags  
Step 7  Review

**Configure group size and scaling - optional** [Info](#)  
Define your group's desired capacity and scaling limits. You can optionally add automatic scaling to adjust the size of your group.

**Group size** [Info](#)  
Set the initial size of the Auto Scaling group. After creating the group, you can change its size to meet demand, either manually or by using automatic scaling.

**Desired capacity type**  
Choose the unit of measurement for the desired capacity value. vCPUs and Memory(GiB) are only supported for mixed instances groups configured with a set of instance attributes.

▼

**Desired capacity**  
Specify your group size.

**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**   
Equal or less than desired capacity

**Max 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 Metrics metric as target value and let the scaling policy adjust the desired capacity in proportion to the metric's value.

**Scaling policy name**

**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 [\[?\]](#)

**Target value**

**Instance warmup** [Info](#)  
 seconds

Disable scale in to create only a scale-out policy

Selected no policy, and checked enable group metrics collection within cloudwatch. Click next.

**Choose a replacement behavior depending on your availability requirements**

- Mixed behavior** (selected)
  - No policy**: For rebalancing events, new instances will launch before terminating others. For all other events, instances terminate and launch at the same time.
- Prioritize availability**
  - Launch before terminating**: Launch new instances and wait for them to be ready before terminating others. This allows you to go above your desired capacity by a given percentage and may temporarily increase costs.
- Control costs**
  - Terminate and launch**: Terminate and launch instances at the same time. This allows you to go below your desired capacity by a given percentage and may temporarily reduce availability.
- Flexible**
  - Custom behavior**: Set custom values for the minimum and maximum amount of available capacity. This gives you greater flexibility in setting how far below and over your desired capacity EC2 Auto Scaling goes when replacing instances.

**Additional capacity settings**

**Capacity Reservation preference** | [Info](#)  
Select whether you want Auto Scaling to launch instances into an existing Capacity Reservation or Capacity Reservation resource group.

- Default**: Auto Scaling uses the Capacity Reservation preference from your launch template.
- None**: Instances will not be launched into a Capacity Reservation.
- Capacity Reservations only**: Instances will only be launched into a Capacity Reservation. If capacity isn't available, the instances fail to launch.
- Capacity Reservations first**: Instances will attempt to launch into a Capacity Reservation first. If capacity isn't available, instances will run in On-Demand capacity.

**Additional settings**

**Instance scale-in protection**  
If protect from scale in is enabled, newly launched instances will be protected from scale in by default.  
 Enable instance scale-in protection

**Monitoring** | [Info](#)  
 Enable group metrics collection within CloudWatch

**Default instance warmup** | [Info](#)  
The amount of time that CloudWatch metrics for new instances do not contribute to the group's aggregated instance metrics, as their usage data is not reliable yet.  
 Enable default instance warmup

[Cancel](#) [Skip to review](#) [Previous](#) [Next](#)

Click next, On next screen I have not added notifications and also tags, but I have skipped these for now. Click next.

**Step 1** Choose launch template  
**Step 2** Choose instance launch options  
**Step 3 - optional** Integrate with other services  
**Step 4 - optional** Configure group size and scaling  
**Step 5 - optional** **Add notifications** (selected)  
**Step 6 - optional** Add tags  
**Step 7** Review

**Add notifications - optional** | [Info](#)  
Send notifications to SNS topics whenever Amazon EC2 Auto Scaling launches or terminates the EC2 instances in your Auto Scaling group.

[Add notification](#)

[Cancel](#) [Skip to review](#) [Previous](#) [Next](#)

**Step 1** Choose launch template  
**Step 2** Choose instance launch options  
**Step 3 - optional** Integrate with other services  
**Step 4 - optional** Configure group size and scaling  
**Step 5 - optional** Add notifications  
**Step 6 - optional** **Add tags** (selected)  
**Step 7** Review

**Add tags - optional** | [Info](#)  
Add tags to help you search, filter, and track your Auto Scaling group across AWS. You can also choose to automatically add these tags to instances when they are launched.

(i) You can optionally choose to add tags to instances (and their attached EBS volumes) by specifying tags in your launch template. We recommend caution, however, because the tag values for instances from your launch template will be overridden if there are any duplicate keys specified for the Auto Scaling group.

**Tags (0)**  
[Add tag](#)  
50 remaining

[Cancel](#) [Previous](#) [Next](#)

ASG is created.

The screenshot shows the AWS Auto Scaling Groups page. At the top, there are navigation links for EC2, IAM, VPC, S3, CloudWatch, Lambda, Route 53, CloudFront, and Aurora and RDS. The main heading is "Auto Scaling groups (1) Info". Below it is a search bar with placeholder text "Search your Auto Scaling groups". A status bar indicates "Last updated less than a minute ago". There are tabs for "Launch configurations", "Launch templates", "Actions", and a prominent orange "Create Auto Scaling group" button. A table lists the single Auto Scaling group: "Name" is "web-tier-template", "Launch template/configuration" is "Version Default", "Instances" is "2", "Status" is "-", "Desired capacity" is "2", "Min" is "2", "Max" is "3", and "Availability Zones" are "2 Availability Zones". At the bottom left, it says "0 Auto Scaling groups selected".

As I can see ASG is doing its job.

The screenshot shows the AWS Instances page. At the top, there are tabs for "Info", "Connect", "Instance state", "Actions", and a yellow "Launch instances" button. A search bar has placeholder text "Find Instance by attribute or tag (case-sensitive)" and a dropdown set to "All states". A "Clear filters" button is also present. The main table lists three instances: "Name" (i-Off0aa72a4cc69210, i-06b319cd55ad0007, 3-tier-web), "Instance ID" (i-0ff0aa72a4cc69210, i-06b319cd55ad0007, i-08124152a7a0a34c5), "Instance state" (Running, Running, Running), "Instance type" (t2.micro, t2.micro, t2.micro), "Status check" (Initializing, Initializing, 2/2 checks passed), "Alarm status" (View alarms +, View alarms +, View alarms +), "Availability Zone" (us-east-1b, us-east-1a, us-east-1a), and "Public IP" (ec2-3-92-..., ec2-3-21..., ec2-13-2...).

I have copied and pasted public IP of newly created Instance in web server, I can see the webpage, Its working.



## Task 3: Creating Application Tier.

Go to EC2 dashboard, select launch template and click create launch template.

The screenshot shows the AWS EC2 Launch Templates page. On the left, there's a navigation sidebar with options like Dashboard, EC2 Global View, Instances, Images, and Elastic Block Store. Under Instances, 'Launch Templates' is selected and highlighted in green. The main content area is titled 'Launch Templates (1) Info'. It shows a table with one row: 'Launch Template ID' (lt-0b6ae1d8043828efc), 'Launch Template Name' (web-tier-template), 'Default Version' (1), 'Latest Version' (1), 'Create Time' (2025-07-14T12:58:30.000Z), and 'Created By' (arn:aws:iam::2850). Below the table, a button labeled 'Create launch template' is highlighted with a yellow box. A modal window titled 'Select a launch template' is open, showing the same information as the main table.

Name the template, enable guidance, Select currently in use AMI.

The screenshot shows the 'Create launch template' wizard. The first step is 'Launch template name and description'. It includes fields for 'Launch template name - required' (containing 'app-tier-template'), 'Template version description' (containing 'app-tier-template'), and 'Auto Scaling guidance' (with a checkbox for 'Provide guidance to help me set up a template that I can use with EC2 Auto Scaling'). Below this is a section for 'Launch template contents' with a note about specifying details. The second step is 'Application and OS Images (Amazon Machine Image) - required'. It shows a search bar ('Search our full catalog including 1000s of application and OS images') and a results list. One result is selected: 'Amazon Linux 2023 AMI 2023.8.2...'. The right side of the screen displays a summary with sections for 'Software Image (AMI)', 'Virtual server type (instance type)', 'Firewall (security group)', and 'Storage (volumes)'. A callout box for the 'Free tier' is shown, stating: 'In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMIs, 750 hours per month of public IPv4 addresses, 30 GB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet.' At the bottom right is a 'Create launch template' button.

Selected t2.micro instance type, add key pair, create security group, name the sg, Select the VPC.

The screenshot shows the AWS Launch Wizard interface for creating a new Amazon EC2 instance. The configuration steps are as follows:

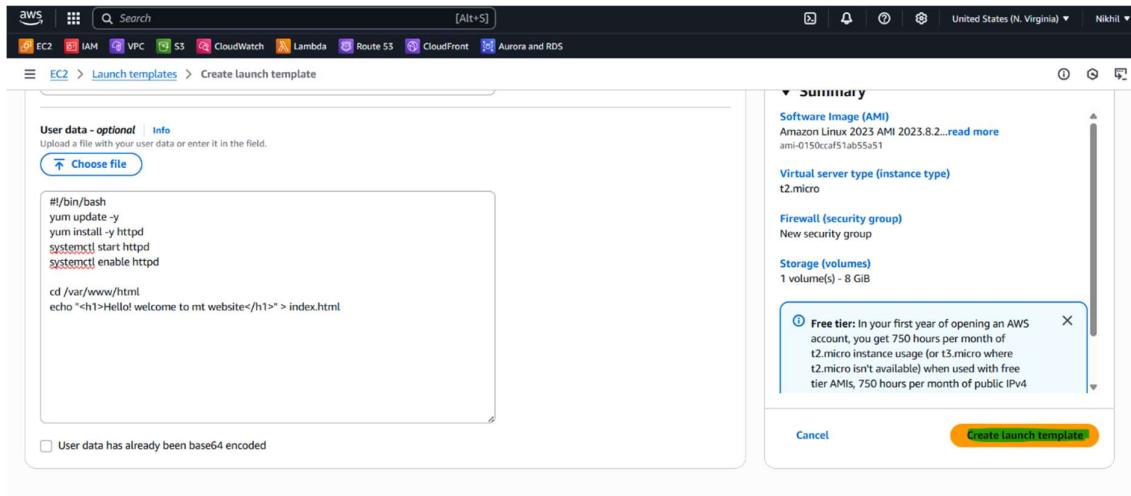
- Instance type:** t2.micro (selected)
- Key pair (login):** testkey (selected)
- Network settings:**
  - Subnet: Don't include in launch template
  - Availability Zone: Not applicable for EC2 Auto Scaling
  - Firewall (security group): sg-07fef093eaf27c418 (selected)
  - Description - required: allow ssh access to application tier
  - VPC: vpc-0f971207 (selected)
- Summary:**
  - Software Image (AMI): Amazon Linux 2023.8.2... (selected)
  - Virtual server type (instance type): t2.micro
  - Firewall (security group): New security group
  - Storage (volumes): 1 volume(s) - 8 GB
  - Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t5.micro where t2.micro isn't available) when used with free tier AMIs, 750 hours per month of public IP4 address usage, 30 GiB of EBS storage, 2 million I/Os, 1 GiB of snapshots, and 100 GiB of bandwidth to the internet.

For inbound SG rules, first I added ssh from anywhere, second I added Custom TCP from custom id sg of web tire, Third I added ALL ICMP – Ipv4 from anywhere.

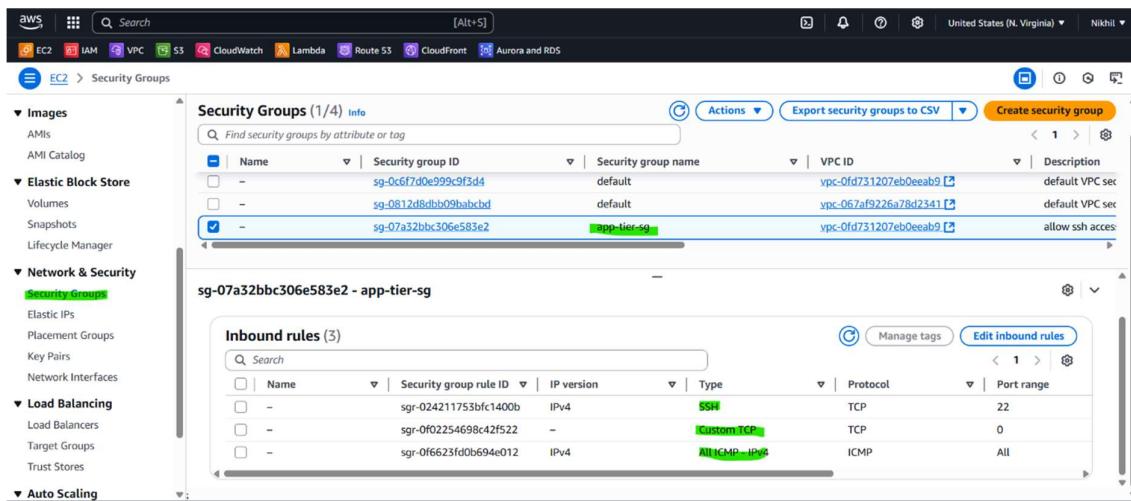
The screenshot shows the Inbound Security Group Rules configuration screen. It displays three rules:

- Security group rule 1 (TCP, 22, 0.0.0.0/0):**
  - Type: ssh
  - Protocol: TCP
  - Port range: 22
  - Source type: Anywhere
  - Description - optional: e.g. SSH for admin desktop
- Security group rule 2 (TCP, 0, sg-07fef093eaf27c418):**
  - Type: Custom (TCP)
  - Protocol: TCP
  - Port range: 0
  - Source type: Custom
  - Source: sg-07fef093eaf27c418
  - Description - optional: e.g. SSH for admin desktop
- Security group rule 3 (ICMP, All, 0.0.0.0/0):**
  - Type: All ICMP - IPv4
  - Protocol: ICMP
  - Port range: All
  - Source type: Anywhere
  - Description - optional: e.g. SSH for admin desktop

In advanced details, at bottom enter the same script as I did earlier for EC2 instance, then click create launch template.



Goto security group and check the Inbound rules.



I have created ASG for app tier, Select Auto scaling groups in EC2 dashboard and click Create Auto scaling group.

The screenshot shows the AWS EC2 Auto Scaling Groups page. On the left, there's a navigation sidebar with sections like Volumes, Snapshots, Lifecycle Manager, Network & Security, Load Balancing, and Auto Scaling. Under Auto Scaling, 'Auto Scaling groups' is selected. The main area displays a table titled 'Auto Scaling groups (1)'. The table has columns for Name, Launch template/configuration, Instances, Status, Desired capacity, Min, and Max. One row is listed: 'web-tier-asg' with 'web-tier-template | Version Default' under Launch template/configuration, 2 instances, and a status of '-'.

Name the ASG, Select the template that is created just now for app tier. Click Next.

This is a screenshot of the 'Choose launch template' step in the AWS Auto Scaling Group creation wizard. The left sidebar shows a step-by-step process from Step 1 to Step 7, with Step 1 selected. The main form is titled 'Choose launch template' and contains the following fields:

- Name:** A text input field where 'web-tier-asg' is typed.
- Launch template:** A dropdown menu showing 'web-tier-template'.
- Version:** A dropdown menu set to 'Default (1)'.
- Description:** A text input field containing 'web-tier-template'.
- AMI ID:** 'ami-0150ccaf51ab55a51'.
- Key pair name:** '3tier'.
- Launch template details:** Shows 'Launch template: web-tier-template lt-0b6ae1d80443828efc', 'Security groups: -', and 'Security group IDs: sg-07fe0935eaef27c418'.
- Additional details:** Shows 'Storage (volumes): -' and 'Date created: Mon Jul 14 2025 18:08:30 GMT+0530 (India Standard Time)'.

At the bottom right, there are 'Cancel' and 'Next Step' buttons.

In Network setting choose the VPC and subnets, select balanced best effort, Click next.

Step 1  
Choose launch template

Step 2  
**Choose instance launch options**

Step 3 - optional  
Integrate with other services

Step 4 - optional  
Configure group size and scaling

Step 5 - optional  
Add notifications

Step 6 - optional  
Add tags

Step 7  
Review

### Choose instance launch options Info

Choose the VPC network environment that your instances are launched into, and customize the instance types and purchase options.

**Instance type requirements Info**

You can keep the same Instance attributes or instance type from your launch template, or you can choose to override the launch template by specifying different instance attributes or manually adding instance types.

Launch template	Version	Description
app-tier-template <small>?</small>	Default	app-tier-template

**Instance type**  
t2.micro

**Override launch template**

### Network Info

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

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

vpc-047f1207e00ead09 (project-1-0a-0-0)

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-0c30e2319f61bbe48 (project-app-3-tier-subnet-private1-0a)  
10.0.128.0/20

(us-east-1a) subnet-025d73c1570f208a (project-app-3-tier-subnet-private2-0a)  
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.

**Cancel** **Skip to review** **Previous** **Next**

Select attach to a new load balancer, select application load balancer, name the load balancer, then select Internal, AZs and subnets are already selected. In listeners and routing select create a target group which should be on port 80 for HTTP traffic.

**Step 1**  
Choose launch template  
  
**Step 2**  
Choose instance launch options  
  
**Step 3 - optional**  
**Integrate with other services**  
  
**Step 4 - optional**  
Configure group size and scaling  
  
**Step 5 - optional**  
Add notifications  
  
**Step 6 - optional**  
Add tags  
  
**Step 7**  
Review

**Integrate with other services - optional** Info

Use a load balancer to distribute network traffic across multiple servers. Enable service-to-service communications with VPC Lattice. Shift resources away from impaired Availability Zones with zonal shift. You can also customize health check replacements and monitoring.

**Load balancing** Info

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.

**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-0fd731207eb0eeab9 [?] project-3-tier-vpc

**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.

<input checked="" type="checkbox"/> use1-az2 (us-east-1b)	<input type="text" value="subnet-029d73c15701208a"/>
<input checked="" type="checkbox"/> use1-az1 (us-east-1a)	<input type="text" value="subnet-0329e2319f610be44"/>

**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)
HTTP	80	Create a target group  New target group name An instance target group with default settings will be created.  <input type="text" value="app-tier-tg-1"/>

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

Select No VPC Lattice service, Turn on elastic load balancing health checks.

**VPC Lattice integration options** Info

To improve networking capabilities and scalability, integrate your Auto Scaling group with VPC Lattice. VPC Lattice facilitates communications between AWS services and helps you connect and manage your applications across compute services in AWS.

Select VPC Lattice service to attach

No VPC Lattice service  
VPC Lattice will not manage your Auto Scaling group's network access and connectivity with other services.

Attach to VPC Lattice service  
Incoming requests associated with specified VPC Lattice target groups will be routed to your Auto Scaling group.

[Create new VPC Lattice service](#)

**Application Recovery Controller (ARC) zonal shift - new** Info

During an Availability Zone impairment, target instance launches towards other healthy Availability Zones.

Enable zonal shift  
New instance launches will be retargeted towards healthy Availability Zones until the zonal shift is canceled.

**Health checks**

Health checks increase availability by replacing unhealthy instances. When you use multiple health checks, all are evaluated, and if at least one fails, instance replacement occurs.

**EC2 health checks**

Always enabled

**Additional health check types - optional** Info

Turn on Elastic Load Balancing health checks Recommended  
Elastic Load Balancing monitors whether instances are available to handle requests. When it reports an unhealthy instance, EC2 Auto Scaling can replace it on its next periodic check.

EC2 Auto Scaling will start to detect and act on health checks performed by Elastic Load Balancing. To avoid unexpected terminations, first verify the settings of these health checks in the [Load Balancer console](#).

Turn on VPC Lattice health checks  
VPC Lattice can monitor whether instances are available to handle requests. If it considers a target as failed a health check, EC2 Auto Scaling replaces it after its next periodic check.

Turn on Amazon EBS health checks  
EBS monitors whether an instance's root volume or attached volume stalls. When it reports an unhealthy volume, EC2 Auto Scaling can replace the instance on its next periodic health check.

**Health check grace period** Info

This time period delays the first health check until your instances finish initializing. It doesn't prevent an instance from terminating when placed into a non-running state.

seconds

[Cancel](#) [Skip to review](#) [Previous](#) [Next](#)

In configure group size and scaling, I have selected desired capacity as 2, Minimum capacity as 2, Maximum capacity as 3, In automatic scaling selected Target tracking scaling policy, Metric type as Average CPU utilization, Target value as 50, Instances need as 50.

The screenshot shows the 'Configure group size and scaling - optional' step in the AWS Auto Scaling wizard. The left sidebar lists steps from 1 to 7, with step 4 highlighted as 'Configure group size and scaling'. The main area contains two sections: 'Group size' and 'Scaling'.

**Group size**

- Desired capacity type: Units (number of instances) (dropdown menu open)
- Desired capacity: 2 (input field)

**Scaling**

- Scaling limits: Min desired capacity (1) and Max desired capacity (3)
- Automatic scaling - optional:
  - 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. This option is selected.
- Scaling policy name: Target Tracking Policy
- Metric type: Average CPU utilization (selected)
- Target value: 50
- Instance warmup: 50 seconds
- Disable scale in to create only a scale-out policy: Unchecked checkbox

Selected no policy, and checked enable group metrics collection within cloudwatch. Click next.

**Instance maintenance policy** [Info](#)

Control your Auto Scaling group's availability during instance replacement events. This includes health checks, instance refreshes, maximum instance lifetime features and events that happen automatically to keep your group balanced, called rebalancing events.

Choose a replacement behavior depending on your availability requirements

- Mixed behavior** 
  - No policy** During rebalancing events, new instances will launch before terminating others. For all other events, instances terminate and launch at the same time.
- Prioritize availability** 
  - Launch before terminating** Launch new instances and wait for them to be ready before terminating others. This allows you to go above your desired capacity by a given percentage and may temporarily increase costs.
- Control costs** 
  - Terminate and launch** Terminate and launch instances at the same time. This allows you to go below your desired capacity by a given percentage and may temporarily reduce availability.
- Flexible** 
  - Custom behavior** Set custom values for the minimum and maximum amount of available capacity. This gives you greater flexibility in setting how far below and over your desired capacity EC2 Auto Scaling goes when replacing instances.

**Additional capacity settings**

**Capacity Reservation preference** [Info](#)

Select whether you want Auto Scaling to launch instances into an existing Capacity Reservation or Capacity Reservation resource group.

- Default** 
  - Auto Scaling uses the Capacity Reservation preference from your launch template.
- None** 
  - Instances will not be launched into a Capacity Reservation.
- Capacity Reservations only** 
  - Instances will only be launched into a Capacity Reservation. If capacity isn't available, the instances fail to launch.
- Capacity Reservations first** 
  - Instances will attempt to launch into a Capacity Reservation first. If capacity isn't available, instances will run in On-Demand capacity.

**Additional settings**

**Instance scale-in protection**  
If protect from scale in is enabled, newly launched instances will be protected from scale in by default.  
 Enable instance scale-in protection

**Monitoring** [Info](#)  
 Enable group metrics collection within CloudWatch

**Default instance warmup** [Info](#)  
The amount of time that CloudWatch metrics for new instances do not contribute to the group's aggregated instance metrics, as their usage data is not reliable yet.  
 Enable default instance warmup

[Cancel](#) [Skip to review](#) [Previous](#) [Next](#)

Click next, On next screen I have not added notifications and also tags, but I have skipped these for now. Click next.

**Auto Scaling groups (2)** [Info](#)

Last updated less than a minute ago

Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max	Availability Zones
app-tier-asg	app-tier-template   Version Default	0	Updating capacity...	2	2	3	2 Availability Zones
web-tier-asg	web-tier-template   Version Default	2	-	2	2	3	2 Availability Zones

ASG for app tier is created and its working.

I tried ssh to private ec2, showing timeout, It's a good sign.

```
C:\Users\nikhi>cd downloads  
C:\Users\nikhi\Downloads>ssh -i "3tier.pem" ec2-user@ec2-54-166-113-178.compute-1.amazonaws.com  
ssh: connect to host ec2-54-166-113-178.compute-1.amazonaws.com port 22: Connection timed out
```

I tried ping private EC2 from public EC2. Its working.

```
C:\Users\nikhi\Downloads>ssh -i "3tier.pem" ec2-user@ec2-13-219-223-21.compute-1.amazonaws.com  
The authenticity of host 'ec2-13-219-223-21.compute-1.amazonaws.com (13.219.223.21)' can't be established.  
ED25519 key fingerprint is SHA256:thJKjmx9aYQnYi36HX16Pe1c/RLDhBA6d53qTehMsXc.  
This key is not known by any other names.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added 'ec2-13-219-223-21.compute-1.amazonaws.com' (ED25519) to the list of known hosts.  
  
[ec2-user@ip-10-0-12-158 ~]$ ping 10.0.148.177  
PING 10.0.148.177 (10.0.148.177) 56(84) bytes of data.  
64 bytes from 10.0.148.177: icmp_seq=1 ttl=127 time=1.94 ms  
64 bytes from 10.0.148.177: icmp_seq=2 ttl=127 time=0.750 ms  
64 bytes from 10.0.148.177: icmp_seq=3 ttl=127 time=0.912 ms  
64 bytes from 10.0.148.177: icmp_seq=4 ttl=127 time=1.20 ms  
64 bytes from 10.0.148.177: icmp_seq=5 ttl=127 time=0.869 ms  
64 bytes from 10.0.148.177: icmp_seq=6 ttl=127 time=0.718 ms
```

Head back to the VPC dashboard, select Route tables, and select one of the route tables that was automatically created when we created our VPC. I only have 1 subnet associated with this table so click on Edit subnet associations.

Name	Route table ID	Explicit subnet associations	Edge associations	Main	VPC	Owner ID
-	rtb-05908ef1a1768acaf	-	-	Yes	vpcl-0fd731207eb0eeab9   proj...	285013783190
-	rtb-012cc6173fa/eca9	-	-	Yes	vpcl-0679226678d2341	285013783190
project-3-tier-rtb-public	rtb-0a13e3f3deafcf5167	2 subnets	-	No	vpcl-0fd731207eb0eeab9   proj...	285013783190
project-3-tier-rtb-private2-us-east-1b	rtb-026b30f1ff7f7014f4	subnet-029d73c15701f2...	-	No	vpcl-0fd731207eb0eeab9   proj...	285013783190
project-3-tier-rtb-private4-us-east-1b	rtb-06d3e8cb21ae620d12	subnet-018756070282d7...	-	No	vpcl-0fd731207eb0eeab9   proj...	285013783190
project-3-tier-rtb-private1-us-east-1a	rtb-05908ef1a1768acaf	subnet-0c30e2319f61bb...	-	No	vpcl-0fd731207eb0eeab9   proj...	285013783190
project-3-tier-rtb-private3-us-east-1a	rtb-06f62a69cb2d11de4	subnet-0200c32b631c87...	-	No	vpcl-0fd731207eb0eeab9   proj...	285013783190

Add another subnet that is private.

Available subnets (2/6)					
<input type="text"/> Filter subnet associations					
	Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table ID
<input checked="" type="checkbox"/>	project-app-3-tier-subnet-private2-u...	subnet-029d73c15701f208a	10.0.144.0/20	-	<a href="#">rtb-026b30f1ff7f014f4 / project-3-tier-rtb-private2-us-...</a>
<input type="checkbox"/>	project-data-3-tier-subnet-private4...	subnet-018756070282d7bc8	10.0.176.0/20	-	<a href="#">rtb-0d63ecb21ae620d12 / project-3-tier-rtb-private4-us-...</a>
<input checked="" type="checkbox"/>	project-app-3-tier-subnet-private1-u...	subnet-0c30e2319f61bbe48	10.0.128.0/20	-	<a href="#">rtb-05908ef1a1768cacf / project-3-tier-rtb-private1-us-...</a>
<input type="checkbox"/>	project-web-3-tier-subnet-public2-u...	subnet-0fe96dc591c69a905	10.0.16.0/20	-	<a href="#">rtb-0a13e3fdeabfc5167 / project-3-tier-rtb-public...</a>
<input type="checkbox"/>	project-data-3-tier-subnet-private3...	subnet-0200c3b631c875cc	10.0.160.0/20	-	<a href="#">rtb-06f62a69cb2d11de4 / project-3-tier-rtb-private3-us-...</a>
<input type="checkbox"/>	project-web-3-tier-subnet-public1-u...	subnet-04e8fb45abe5ad8	10.0.0.0/20	-	<a href="#">rtb-0a13e3fdeabfc5167 / project-3-tier-rtb-public...</a>

Selected subnets	
<a href="#">subnet-0c30e2319f61bbe48 / project-app-3-tier-subnet-private1-us-east-1a</a>	<a href="#">subnet-029d73c15701f208a / project-app-3-tier-subnet-private2-us-east-1b</a>

## Task 4: Creating Database Tier

Go to RDS console and select subnet groups and click create subnet group. Name the subnet group, select VPC, add subnet in which database private ones AZs and subnet. Click Next.

**Create DB subnet group**

To create a new subnet group, give it a name and a description, and choose an existing VPC. You will then be able to add subnets related to that VPC.

### Subnet group details

**Name**  
You won't be able to modify the name after your subnet group has been created.  
 Must contain from 1 to 255 characters. Alphanumeric characters, spaces, hyphens, underscores, and periods are allowed.

**Description**

**VPC**  
Choose a VPC identifier that corresponds to the subnets you want to use for your DB subnet group. You won't be able to choose a different VPC identifier after your subnet group has been created.  
 (VPC identifier: vpc-000000000000000000)

### Add subnets

**Availability Zones**  
Choose the Availability Zones that include the subnets you want to add.  
 (Search, Sort, Filter)

**Subnets**  
Choose the subnets that you want to add. The list includes the subnets in the selected Availability Zones.  
 (Search, Sort, Filter)

Subnet ID: subnet-000000000000000000 | CIDR block: 10.0.1.0/24  
 Subnet ID: subnet-000000000000000000 | CIDR block: 10.0.2.0/24

ⓘ For Multi-AZ DB clusters, you must select 3 subnets in 3 different Availability Zones.

**Subnets selected [2]**

Availability zone	Subnet name	Subnet ID	CIDR block
us-east-1a	project-data-1-1st-subnet-private-us-east-1a	subnet-020000000000000000	10.0.160.0/20
us-east-1b	project-data-1-1st-subnet-private-us-east-1b	subnet-01875607028207bcd	10.0.176.0/20

Back in the RDS console click Create database. Select the MySQL DB.

**Create database [Info](#)**

**Choose a database creation method**

- Standard create You set all of the configuration options, including ones for availability, security, backups, and maintenance.
- Easy create Use recommended best-practice configurations. Some configuration options can be changed after the database is created.

**Engine options**

Engine type [Info](#)

<input type="radio"/> Aurora (MySQL Compatible)	<input type="radio"/> Aurora (PostgreSQL Compatible)	<input checked="" type="radio"/> MySQL
<input type="radio"/> PostgreSQL	<input type="radio"/> MariaDB	<input type="radio"/> Oracle
<input type="radio"/> Microsoft SQL Server	<input type="radio"/> IBM Db2	<b>IBM Db2</b>

**Edition**

MySQL Community

**Engine version [Info](#)**  
View the engine versions that support the following database features.

**Hide filters**

- Show only versions that support the Multi-AZ DB cluster [Info](#)  
Create a Multi-AZ DB cluster with one primary DB instance and two readable standby DB instances. Multi-AZ DB clusters provide up to 2x faster transaction commit latency and automatic failover in typically under 35 seconds.
- Show only versions that support the Amazon RDS Optimized Writes [Info](#)  
Amazon RDS Optimized Writes improves write throughput by up to 2x at no additional cost.

**Engine version**

**MySQL 8.0.21**

Enable RDS Extended Support [Info](#)  
Amazon RDS Extended Support is a paid offering. By selecting this option, you consent to being charged for this offering if you are running your database major version past the RDS end of standard support date for that version. Check the end of standard support date for your major version in the [RDS for MySQL documentation](#).

There are also availability and durability options; however, with the free-tier, none are available. We do not need them either.

**Templates**  
Choose a sample template to meet your use case.

- Production Use defaults for high availability and fast, consistent performance.
- Dev/Test This instance is intended for development use outside of a production environment.
- Free Tier Use RDS Free Tier to develop new applications, test existing applications, or gain hands-on experience with Amazon RDS. [Info](#)

**Availability and durability**

**Deployment options [Info](#)**  
Choose the deployment option that provides the availability and durability needed for your use case. AWS is committed to a certain level of uptime depending on the deployment option you choose. Learn more in the [Amazon RDS service level agreement \(SLA\)](#).

- Multi-AZ DB cluster deployment (3 instances)  
Creates a primary DB instance with two readable standbys in separate Availability Zones. This setup provides:
  - 99.95% uptime
  - Redundancy across Availability Zones
  - Increased read capacity
  - Reduced write latency
- Multi-AZ DB instance deployment (2 instances)  
Creates a primary DB instance with a non-readable standby instance in a separate Availability Zone. This setup provides:
  - 99.95% uptime
  - Redundancy across Availability Zones
- Single-AZ DB instance deployment (1 instance)  
Creates a single DB instance with no standby instance. This setup provides:
  - 99.5% uptime
  - No data redundancy

Under Settings, name your DB and create a master username and password. This username and password these should be different than your AWS account user login, as it is specific to the database you are creating.

**Settings**

**DB instance identifier** [Info](#)  
Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current AWS Region.

**Credentials Settings**

**Master username** [Info](#)  
Type a login ID for the master user of your DB instance.

**Credentials management**  
You can use AWS Secrets Manager or manage your master user credentials.

**Managed in AWS Secrets Manager - most secure**  
RDS generates a password for you and manages it throughout its lifecycle using AWS Secrets Manager.

**self-managed**  
Create your own password or have RDS create a password that you manage.

**Auto generate password**  
Amazon RDS can generate a password for you, or you can specify your own password.

**Master password** [Info](#)

**Password strength** Very strong

Minimum constraints: At least 8 printable ASCII characters. Can't contain any of the following symbols: / " \* @

**Confirm master password** [Info](#)

Under Instance configuration, the burstable classes option is pre-selected because it's the only one available for the free tier. I left my instance type as a db.t2.micro. You can add storage as needed; I left mine on default settings.

**Instance configuration**

The DB instance configuration options below are limited to those supported by the engine that you selected above.

**DB instance class** [Info](#)  
 **Show instance classes that support Amazon RDS Optimized Writes** [Info](#)  
Amazon RDS Optimized Writes improves write throughput by up to 2x at no additional cost.  
 **Include previous generation classes**

**Burstable classes (includes t classes)**

**db.t2.micro**  
2 vCPUs • 4 GiB RAM • Network: Up to 2,085 Mbps

**Storage**

**Storage type** [Info](#)  
Provisioned IOPS SSD (io2) storage volumes are now available.  
 **General Purpose SSD (gp2)**  
Baseline performance determined by volume size

**Allocated storage** [Info](#)

**Additional storage configuration**

**Storage autoscaling** [Info](#)  
Provides dynamic scaling support for your database's storage based on your application's needs.  
 **Enable storage autoscaling**  
Enabling this feature will allow the storage to increase after the specified threshold is exceeded.

**Maximum storage threshold** [Info](#)  
Changes will apply when your database autoscales to the specified threshold.

I am going to set up our network manually so choose not to connect to an EC2 resource. Select the proper VPC; the subnet group that you created earlier should be listed as default. Select Create new VPC security group (firewall).

**Connectivity** [Info](#)

**Compute resource**  
Choose whether to set up a connection to a compute resource for this database. Setting up a connection will automatically change connectivity settings so that the compute resource can connect to this database.

**Don't connect to an EC2 compute resource**  
Don't set up a connection to a compute resource for this database. You can manually set up a connection to a compute resource later.

**Connect to an EC2 compute resource**  
Set up a connection to an EC2 compute resource for this database.

**Network type** [Info](#)  
To use dual-stack mode, make sure that you associate an IPv6 CIDR block with a subnet in the VPC you specify.

**IPv4**  
Your resources can communicate only over the IPv4 addressing protocol.

**Dual-stack mode**  
Your resources can communicate over IPv4, IPv6, or both.

**Virtual private cloud (VPC)** [Info](#)  
Choose the VPC. The VPC defines the virtual networking environment for this DB instance.

**project-5-tier-vpc (Vpc-0f4751207650ec6b9)**  
6 Subnets, 2 Availability Zones

Only VPCs with a corresponding DB subnet group are listed.

**DB subnet group** [Info](#)  
Choose the DB subnet group. The DB subnet group defines which subnets and IP ranges the DB instance can use in the VPC that you selected.

**database-subnet-group-new**  
Subnets: 3 Availability Zones: 2

**Public access** [Info](#)

**Yes**  
RDS assigns a public IP address to the database. Amazon EC2 instances and other resources outside of the VPC can connect to your database. Resources inside the VPC can also connect to the database. Choose one or more VPC security groups that specify which resources can connect to the database.

**No**  
RDS doesn't assign a public IP address to the database. Only Amazon EC2 instances and other resources inside the VPC can connect to your database. Choose one or more VPC security groups that specify which resources can connect to the database.

**VPC security group (firewall)** [Info](#)  
Choose one or more VPC security groups to allow access to your database. Make sure that the security group rules allow the appropriate incoming traffic.

**Choose existing**  
Choose existing VPC security groups

**Create new**  
Create new VPC security group

**New VPC security group name**  
**date-boring**

**Availability Zone** [Info](#)

**us-east-1a**

**RDS Proxy**  
RDS Proxy is a fully managed, highly available database proxy that improves application scalability, resiliency, and security.

**Create an RDS Proxy** [Info](#)  
RDS automatically creates an IAM role and a Secrets Manager secret for the proxy. RDS has additional costs. For more information, see [Amazon RDS Proxy pricing](#).

**Certificate authority - optional** [Info](#)  
Using a server certificate provides an extra layer of security by validating that the connection is being made to an Amazon database. It does so by checking the server certificate that is automatically installed on all databases that you provision.

**rds-ca-sa2048-g1 (default)**  
Expires: May 26, 2061

If you don't select a certificate authority, RDS chooses one for you.

**Additional configuration**

**Tags - optional**  
A tag consists of a case-sensitive key-value pair.  
No tags associated with the resource.

**Add new tag**  
You can add up to 50 more tags.

**Database authentication**

**Database authentication options** [Info](#)

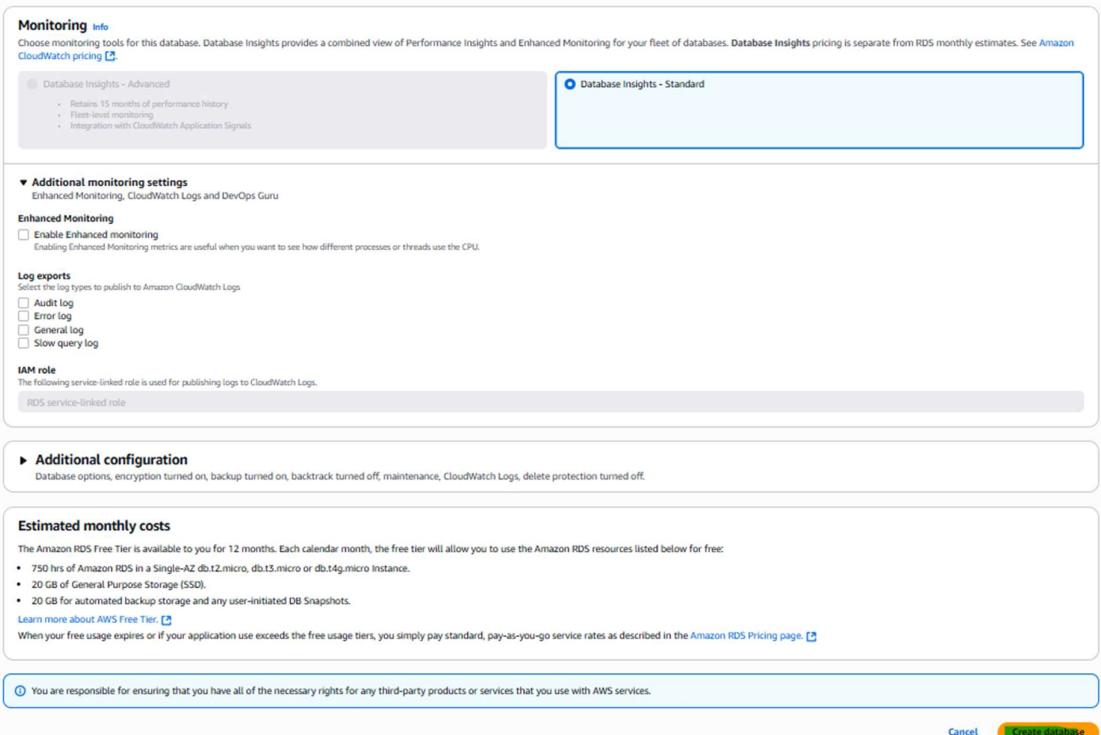
**Password authentication**  
Authenticates using database passwords.

**Password and IAM database authentication**  
Authenticates using the database password and user credentials through AWS IAM users and roles.

**Password and Kerberos authentication**  
Choose a directory in which you want to allow authorized users to authenticate with this DB instance using Kerberos Authentication.

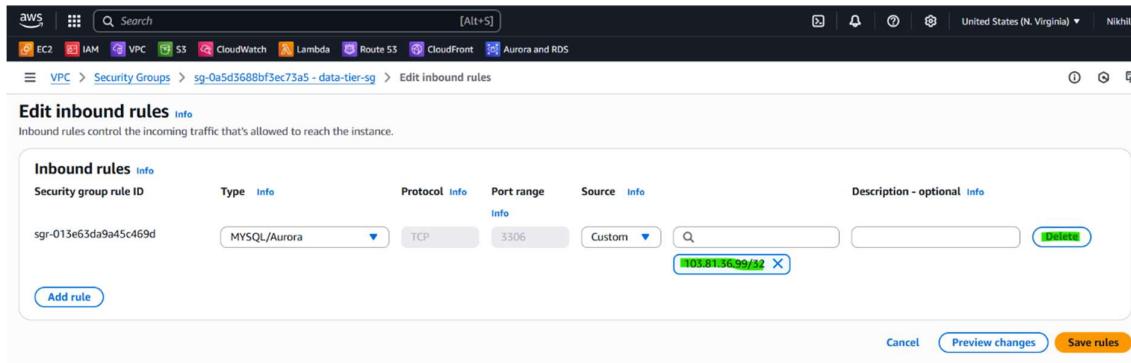
In Database authentication I left the default checked.

Click the “Create database” button.



Go to the VPC console, select Security groups on the left side menu, and then find the database tier security group you just created.

Select the security group that you just created. You need to edit inbound rules; by default the database SG has an inbound rule to allow MySQL/Aurora traffic on port 3306 from your IP address. Delete this rule.



Create a new rule for MySQL/Aurora on port 3306: for the Source, select Custom to add your security group for your application layer (tier-2 SG).

The screenshot shows the 'Edit inbound rules' page in the AWS VPC console. A new rule is being added for MySQL/Aurora on TCP port 3306. The source is set to 'Custom' and the value is 'sg-07a32bbc306e583e'. The 'Save rules' button is highlighted.

In the last step for our database tier, we need to make sure that the route table we associate with our databases private subnets has both subnets listed in subnet associations. If not, add the other subnet, and save.

The screenshot shows the 'Route tables' page in the AWS VPC console. A route table is associated with multiple subnets. The 'Save associations' button is highlighted.

The screenshot shows the 'Subnet associations' dialog box. Available subnets include 'project-app-3-tier-subnet-private2-u...' and 'project-data-3-tier-subnet-private4...'. Selected subnets are 'subnet-0200c32b631c875cc' and 'subnet-018756070282d7bc8'. The 'Save associations' button is highlighted.

Our three-tier architecture is done! We have already tested our web and application layers, but we are going to go a step further here.

## Task5: Testing

We can't directly SSH to the database, but we can use an SSH forwarding agent to achieve this.

You need to add your access key pair file to your keychain. To do this, first make sure you are in your local host (use the command exit to get out of any EC2 instance you're connected to). Then use the following command:

```
PS C:\WINDOWS\system32> cd C:\Users\nikhi\Downloads
PS C:\Users\nikhi\Downloads> Start-Service ssh-agent
PS C:\Users\nikhi\Downloads> Get-Service ssh-agent | Set-Service -StartupType Automatic
PS C:\Users\nikhi\Downloads> ssh-add C:\path\to\your\3tier.pem
C:\path\to\your\3tier.pem: No such file or directory
PS C:\Users\nikhi\Downloads> ssh-add "C:\Users\nikhi\Downloads\3tier.pem"
Identity added: C:\Users\nikhi\Downloads\3tier.pem (C:\Users\nikhi\Downloads\3tier.pem)
PS C:\Users\nikhi\Downloads>
```

Now that your key pair file is added to your keychain, the SSH agent will scan through all of the keys associated with the keychain and find your matching key.

Now reconnect to the web tier EC2; however, this time use -A to specify you want to use the SSH agent.

```
PS C:\Users\nikhi\Downloads>
PS C:\Users\nikhi\Downloads> ssh -A ec2-user@13.219.223.21
The authenticity of host '13.219.223.21 (13.219.223.21)' can't be established.
ED25519 key fingerprint is SHA256:thJKjmx9aYQnYi36HX16Pe1c/RLDhBA6d53qTehMsXc.
This host key is known by the following other names/addresses:
    C:\Users\nikhi/.ssh/known_hosts:88: ec2-13-219-223-21.compute-1.amazonaws.com
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '13.219.223.21' (ED25519) to the list of known hosts.

,      #
~\_  ####_      Amazon Linux 2023
~~ \#####\
~~  \###|
~~   \|/ __| https://aws.amazon.com/linux/amazon-linux-2023
~~    \~'-'>
~~     / \
~~-.--/_/ \
~~  /_/
~/m/'

Last login: Mon Jul 14 13:15:49 2025 from 103.81.36.99
[ec2-user@ip-10-0-12-158 ~]$
```

Once you are logged back into your tier-1 EC2, use the following command to check if the SSH agent forwarded the private key.

ssh-add -l

```
[ec2-user@ip-10-0-12-158 ~]$ ssh-add -l  
2048 SHA256:ArCm+eWb10lb7VwV1Ts286ciUKWkqEP0uR1wcdeQ1BE C:\Users\nikhi\Downloads\3tier.pem (RSA)  
2048 SHA256:av3noIcqkYT1/a8uYEF1zjP76LjxiLrTa7lut9xHbdY .\3tier.pem (RSA)  
[ec2-user@ip-10-0-12-158 ~]$
```

Our key pair has been forwarded to our public instance. Go copy your tier-2 application layer private IP address and copy it into the next command.

```
ssh -A ec2-user@<private-ip-address>
```

SSH'd from your public tier 1 web instance into your private tier 2 application instance.

```
[ec2-user@ip-10-0-12-158 ~]$ ssh -A ec2-user@10.0.131.58
The authenticity of host '10.0.131.58 (10.0.131.58)' can't be established.
ED25519 key fingerprint is SHA256:8XZv1gI6R5Vt/Lvw6b87hssfniQg8G6wEBtUXJQ7kgA.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.131.58' (ED25519) to the list of known hosts.

          _#
         ~\_ #####_      Amazon Linux 2023
        ~~ \#####\
        ~~  \###|
        ~~   \#/   __ https://aws.amazon.com/linux/amazon-linux-2023
        ~~    \~' '-'>
        ~~~   /
        ~~.~.  /
           /_/
           /m/'

[ec2-user@ip-10-0-131-58 ~]$
```

## Installed database server (mariadb)

```
sudo dnf install mariadb105-server
```

```
[ec2-user@ip-10-0-131-58 ~]$ sudo dnf install mariadb105-server
Last metadata expiration check: 0:43:49 ago on Mon Jul 14 13:13:01 2025.
Dependencies resolved.

-----
```

Package	Architecture	Version	Repository	Size
<b>Installing:</b>				
<b>mariadb105-server</b>	x86_64	3:10.5.29-1.amzn2023.0.1	amazonlinux	10
<b>Installing dependencies:</b>				
mariadb-connector-c	x86_64	3.3.10-1.amzn2023.0.1	amazonlinux	211
mariadb-connector-c-config	noarch	3.3.10-1.amzn2023.0.1	amazonlinux	9.9
mariadb105	x86_64	3:10.5.29-1.amzn2023.0.1	amazonlinux	28
mariadb105-common	x86_64	3:10.5.29-1.amzn2023.0.1	amazonlinux	212
mariadb105-ermsg	x86_64	3:10.5.29-1.amzn2023.0.1	amazonlinux	36
mysql-selinux	noarch	1.0.4-2.amzn2023.0.1	amazonlinux	177
perl-B	x86_64	1.80-477.amzn2023.0.7	amazonlinux	153
perl-DBD-MariaDB	x86_64	1.22-1.amzn2023.0.4	amazonlinux	153

Log into RDS MySQL database. Need RDS endpoint, user name, and password. To find RDS database endpoint, navigate to the database that created and find the endpoint under Connectivity & Security.

```
mysql -h <rds-database-endpoint> -P 3306 -u <username> -p
```

```
[ec2-user@ip-10-0-131-58 ~]$ mysql -h data-tier.cybuae22uvgg.us-east-1.rds.amazonaws.com -P 3306 -u admin -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MySQL connection id is 29
Server version: 8.0.41 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>
```

Successfully connected to MySQL database from the application tier. I have connectivity with all of our tiers.

By implementing a 3-tier architecture on AWS, organizations and developers can build applications that are not only robust and secure, but also easily scalable and maintainable over time. This architecture reflects best practices in cloud infrastructure design and serves as a strong foundation for deploying production-grade web applications. In this project, the 3-tier model is brought to life using various AWS services to demonstrate how cloud-native solutions can be used to meet modern application demands.