



# AWS PROJECT REPORT



Submitted By

**P VIKAS**

Under the guidance of

**Yasaswi Joseph Surabhi**

## **What is AWS?**

The AWS platform is a comprehensive cloud computing solution for project deployment and management. It includes several key services:

1. EC2 (Elastic Compute Cloud): Provides resizable virtual servers, enabling easy deployment and scaling of applications on demand.
2. S3 (Simple Storage Service): Offers scalable object storage for securely storing and retrieving data, such as documents, images, and backups.
3. VPC (Virtual Private Cloud): Enables the creation of isolated virtual networks to enhance security and control over resources.
4. CodePipeline: An automated continuous integration and continuous delivery (CI/CD) service that streamlines the software release process.
5. IAM (Identity and Access Management): Manages user access and permissions to AWS resources, ensuring secure and controlled operations.

Utilizing AWS services like EC2 for computing power, S3 for data storage, VPC for network isolation, CodePipeline for automated deployments, and IAM for access control, project teams can benefit from scalability, reliability, and cost-effectiveness while deploying and managing applications in the cloud.

## **Advantages of AWS:**

AWS (Amazon Web Services) cloud platform offers numerous advantages, some of which include:

1. Scalability: AWS allows easy and rapid scaling of resources, ensuring that applications can handle varying workloads and traffic without interruptions.

2. Cost-Effectiveness: With a pay-as-you-go pricing model, users only pay for the resources they use, reducing upfront costs and eliminating the need for investing in hardware.

3. Flexibility: AWS offers a wide range of services and configurations, allowing users to choose the most suitable options for their specific needs.

4. Global Reach: AWS has data centers in multiple regions worldwide, enabling global distribution of applications and low-latency access to users.

5. Reliability and Availability: AWS provides high availability and redundancy, ensuring that applications remain accessible even in the event of hardware failures.

6. Security: AWS implements robust security measures to protect data and applications, including encryption, IAM, DDoS protection, and compliance with various industry standards.

7. Easy Deployment: AWS services like Elastic Beanstalk and AWS Lambda simplify the deployment process, making it faster and more efficient.

8. Managed Services: Many AWS services are fully managed, meaning AWS handles administrative tasks like patching, backups, and updates, allowing users to focus on their applications.

9. Innovation: AWS regularly introduces new services and features, keeping up with the latest technological advancements and enabling users to leverage cutting-edge tools.

10. Integrations: AWS integrates with various third-party tools and services, allowing users to build comprehensive and customized solutions.

11. Disaster Recovery: AWS offers disaster recovery solutions, enabling businesses to implement robust backup and recovery strategies.

Overall, AWS provides a powerful and reliable cloud infrastructure, empowering businesses and developers to build, deploy, and manage applications with ease, efficiency, and scalability.

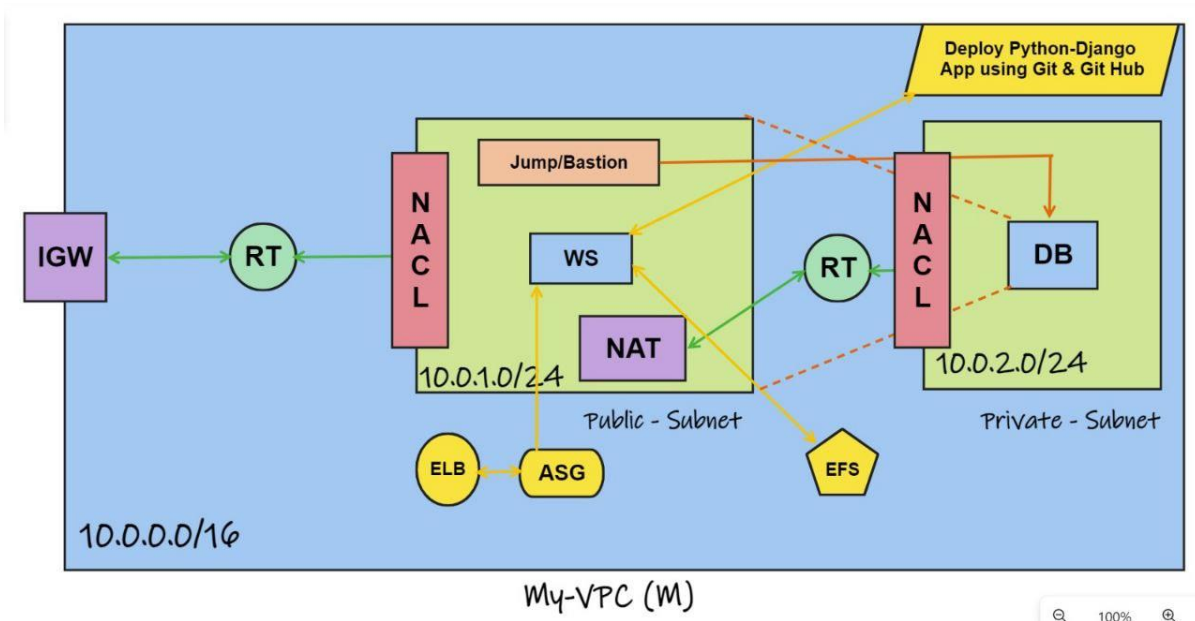
**Note:** Main reason to make AWS popular is an option called Auto Scaling Groups.

### **Usage of AWS in Real Time:**

Some of the most top companies depends upon AWS. Some of those companies are

- NASA
- NETFLIX
- AMAZON
- AIRBNB
- NASA
- SLACK
- PINTREST
- ADOBE SYSTEMS

## PROJECT:



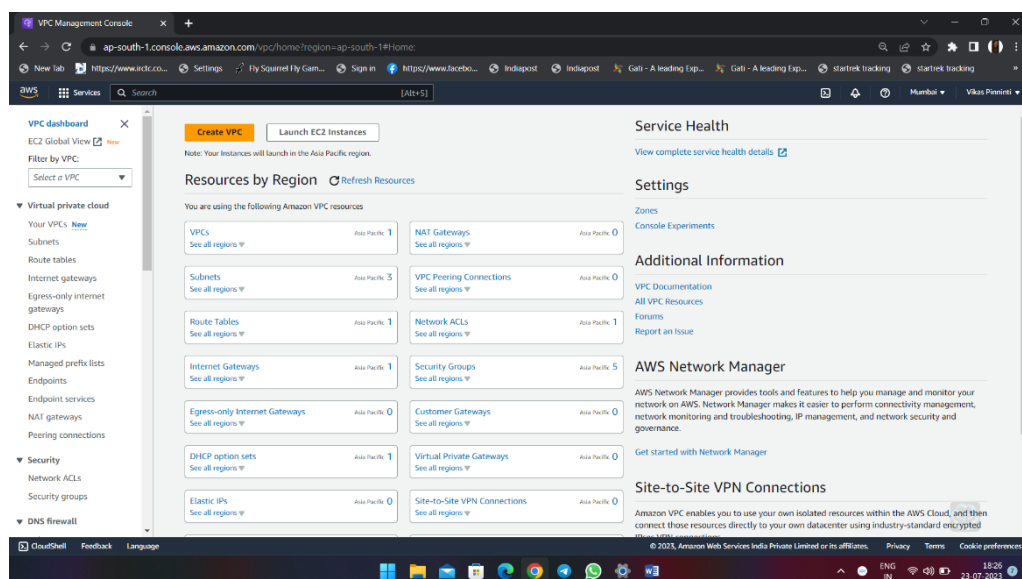
We need a network to start this project in AWS.

We use our Private Network in AWS to start this project which is called **VPC** (Virtual Private Cloud).

So, firstly we create a VPC.

To create a VPC we need to login in AWS Console (<https://aws.amazon.com/console/>) and make sure we are working in Mumbai region as project needs to be done from Mumbai region (\*mentioned).

Now in the search bar search for VPC and open it.



Now create a VPC

The given requirements to create a VPC are:

- Name – My VPC
- IPv4 CIDR – 10.0.0.0/16

After Creating the VPC, proceed to subnets and create two subnets with below configurations.

**Note:** When VPC is created

- Route Table – Main
- DHCP
- NACL – Main , are created.

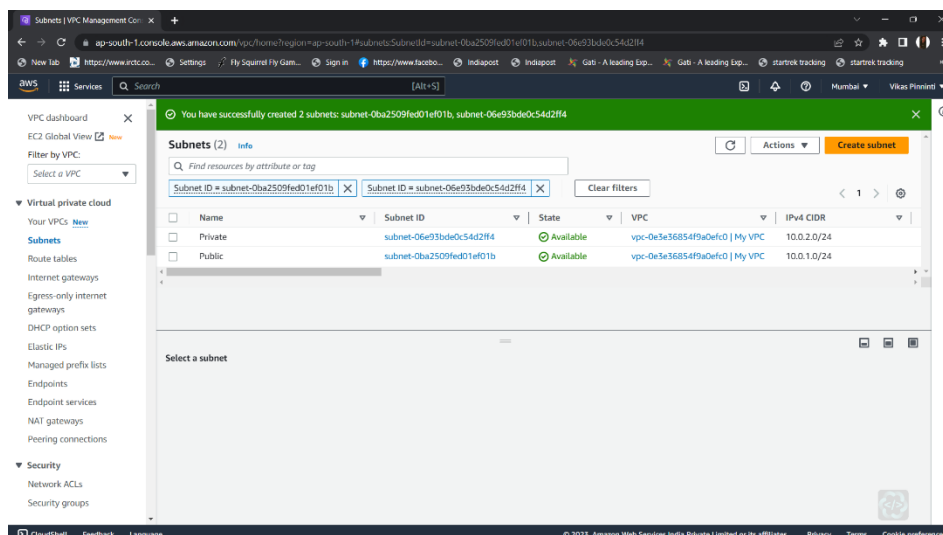
These subnets should **Private + Public** configuration because we host our database servers in Private Subnet which can be only accessible to some of company persons.

1<sup>st</sup> Subnet:

- Name – Public
- VPC – My VPC
- Availability Zone – ap-south-1a
- IPv4 CIDR – 10.0.1.0/24

2<sup>nd</sup> Subnet:

- Name - Private
- VPC – My VPC
- Availability Zone – ap-south-1b



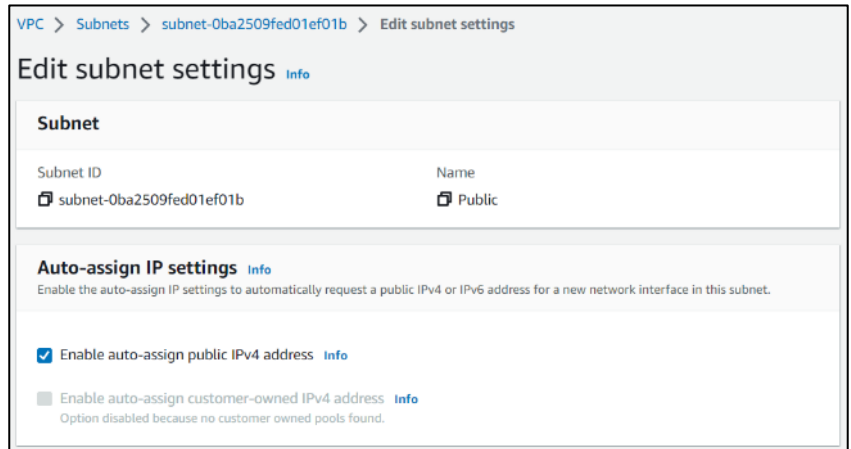
But,

When a **subnet** is created it is created in **private** nature.

To make a subnet public we need to do two things.

They are:

- Enable auto-assign public IPv4 address
- Attaching Internet Gateway to Subnet

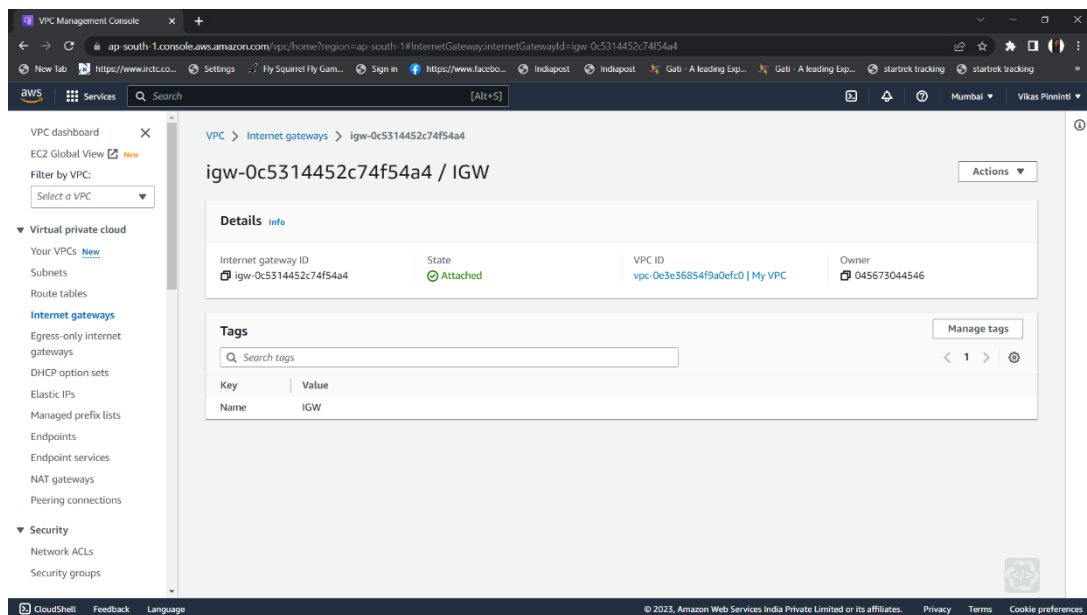


Now,

Go to internet gateways from EC2 Dashboard and create an internet gateway.

After creating an internet gateway attach it to VPC (My VPC).

**Note:** An internet gateway can be attached to only one VPC.



Now,

Go to route tables which are used to connect to internet and subnet for secured internet connection.

Now, create a Route Table with a name called **RT – Custom**

Now do the following modifications,

In the **RT – Custom**

Go to routes and edit routes and add route with

- Destination – 0.0.0.0/0 (which is used for internet)
- Target – Internet Gateway

Routes (2)	
<input type="text" value="Filter routes"/>	
Destination	Target
0.0.0.0/0	igw-0c5314452c74f54a4
10.0.0.0/16	local

Now, go to subnet associations and edit subnet associations and add **Public** subnet.

Routes	Subnet associations	Edge associations	Route propagation	Tags
<b>Explicit subnet associations (1)</b>				
<input type="text" value="Find subnet association"/>				
<div>1</div>				
Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	
Public	subnet-0ba2509fed01ef01b	10.0.1.0/24	-	

Now, go to NAT Gateways and create a NAT Gateway with the below configurations.

- Name – NAT
- Subnet – Public
- Allocate Elastic IP

VPC > NAT gateways > nat-06048f7a60bc401a7			
nat-06048f7a60bc401a7 / NAT			Actions
<b>Details</b> Info			
NAT gateway ID nat-06048f7a60bc401a7	Connectivity type Public	State Pending	State message -
NAT gateway ARN arn:aws:ec2:ap-south-1:045673044546:natgateway/nat-06048f7a60bc401a7	Primary public IPv4 address -	Primary private IPv4 address -	Primary network interface ID -
VPC vpc-0e3e36854f9a0efc0 / My VPC	Subnet subnet-0ba2509fed01ef01b / Public	Created Tuesday, July 25, 2023 at 17:25:12 GMT+5:30	Deleted -

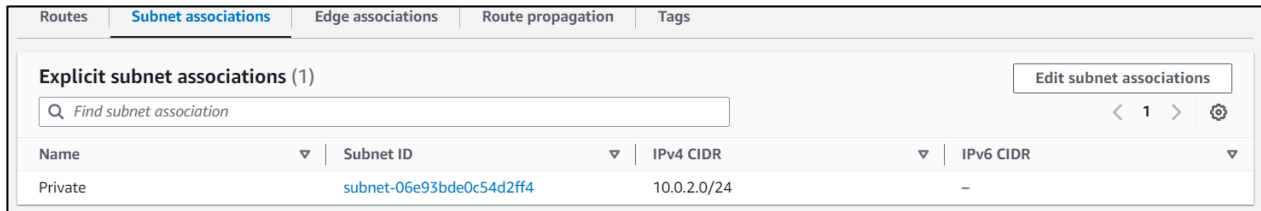
Now, again go back to route tables and edit **RT - M** which is created while creating the VPC.



In that route table go to edit routes and add a route

- Destination – 0.0.0.0/0 (which is used for internet)
- Target – NAT Gateway

And after the routes, go to subnet associations and edit subnet associations and add **Private** subnet.



Routes	Subnet associations	Edge associations	Route propagation	Tags
Explicit subnet associations (1)				
<input type="text" value="Find subnet association"/>				
<a href="#">Edit subnet associations</a>				
<div>&lt; 1 &gt; ⚙</div>				
Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	
Private	subnet-06e93bde0c54d2ff4	10.0.2.0/24	–	

We have created a network successfully.

Now, go to EC2 Dashboard.

Go to, now create an ASG...

But, to create an ASG we need

- Launch Template
- Load Balancer
- Scaling Policies

First create a Launch Template,

Go to launch templates and create a launch template with given configurations.

- Name – Project
- Description – AWS
- **Enable Auto Scaling Guidance**
- AMI – Linux 2023 (Free tier Eligible)
- Instance Type – t2.micro (Free tier Eligible)
- Key Pair – Create a new key pair named Project
- Network Settings – Create a new security group with
  - Name – Project
  - VPC – My VPC
  - Inbound Rules

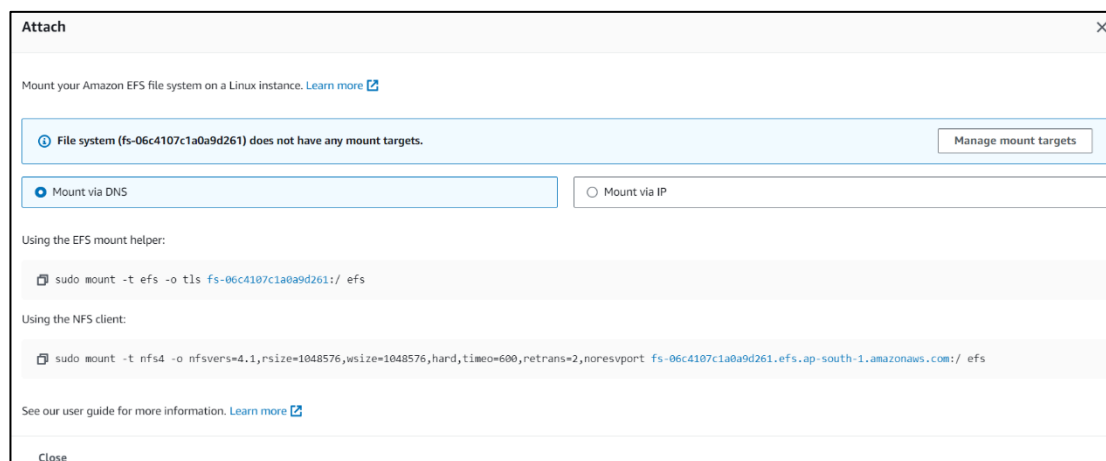
Type	Port Range	Source Type	Source	Description
ssh	20	My IP	152.58.233.249/32	Admin
HTTP	80	Anywhere	0.0.0.0/0	Everyone
Custom TCP	8000	Anywhere	0.0.0.0/0	Django

- Now, go to advanced details and scroll down till user data box and enter some bin bash commands with **EFS Attachment Url**(nfs client).

Go to EFS in aws console in a new tab and create a file system.

- Name – Project
- Select VPC – My VPC

Now open Project and click on attach and copy the NFS Client



Now comeback to EC2 tab


Now, go to Advanced Details in Launch Template and add this command in bin bash commands.

**Note:** Bin bash commands are used to convert user data box into a terminal. These are used to create an auto server where we save time and doesn't need to update these commands in the instance terminal using putty or etc.

Bin Bash Command: **#!/bin/bash**

#### User data - optional [Info](#)

Upload a file with your user data or enter it in the field.

 Choose file





```
#!/bin/bash
sudo su
yum update -y
yum install httpd -y
sudo mount -t nfs4 -o
nfsvers=4.1,rsize=1048576,wsz=1048576,hard,timeo=600,retrans=2,noresvport
fs-06c4107c1a0a9d261.efs.ap-south-1.amazonaws.com:/ Project
cd /var/www/html
echo "Webserver" > index.html
ls
pwd
service httpd start
chkconfig httpd on
cat index.html
```

### • Create a Launch Template

**Project (lt-0ff3551e105f80508)**

Actions ▼Delete template

**Launch template details**


Launch template ID  lt-0ff3551e105f80508	Launch template name  Project	Default version  1	Owner  arn:aws:iam::045673044546:root
--	---	--	---


DetailsVersionsTemplate tags


**Launch template version details**

Actions ▼Delete template version





Version  
1 (Default) ▼

Description  
 AWS

Date created  
 2023-07-25T12:25:52.000Z

Created by  
 arn:aws:iam::045673044546:root

Instance detailsStorageResource tagsNetwork interfacesAdvanced details

AMI ID  ami-072ec8f4ea4a6f2cf	Instance type  t2.micro	Availability Zone -	Key pair name  Project
Security groups -	Security group IDs  sg-081581be637444e17		

Now, create a Load Balancer with given configuration

- Load Balancer Type – Classic Load Balancer
- Name – Project
- VPC – My VPC
- Mappings – 1a
- Security Groups – Project
- Health Checks -> Advanced Health Check Settings -> Healthy Thresholds ->2

- Create Load Balancer

	Name	DNS name	State	VPC ID	Availability Zones	Type	Date created
<input checked="" type="checkbox"/>	<a href="#">Project</a>	Project-362268708.ap-sou...	-	vpc-0e3e36854f9a0efc0	ap-south-1a (aps1-az1)	classic	July 21, 2019 (UTC+5:30)

Now go to Auto Scaling Groups, Create an auto scaling group with below configurations.

- Name – ASG
- Launch Template – Project -> next
- Select VPC – My VPC
- Availability Zones – 1a,1b -> next
- Load Balancing -> attach an existing load balancer -> choose from classic load balancer -> Project
- Health Checks – Turn on Elastic load balancing health checks. -> next
- **Group size**
  - Desired Capacity – 1
  - Minimum Capacity – 1
  - Maximum Capacity – 1
- Scaling Policies (Not needed because we launch only one server)
  - Target Scaling Policies
    - Scaling Policy Name - Target Tracking Policy
    - Metric Type – Average CPU Utilization
    - Target Value – 90
    - Instances Needed – 10 seconds ->next
    - Next -> Next -> Create Auto Scaling Group

	Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max	Availability Zones
<input type="checkbox"/>	<a href="#">ASG</a>	<a href="#">Project</a>   Version Default	0	Updating capacity...	1	1	1	ap-south-1a (aps1-az1)

A web server is created now using ASG.

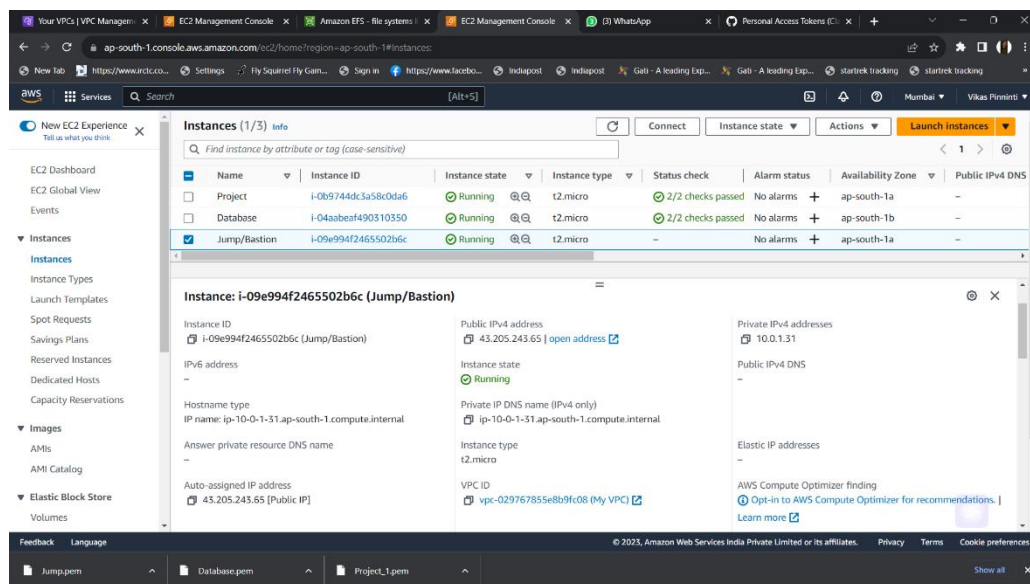
We create a **jump box** server which used to connect to database securely

To create a jump box

- Go to instance
- Launch instance
  - Name – Jump/Bastion Box
  - AMI – Amazon Linux 2023 (free tier eligible)
  - Instance type – t2.micro
  - Key Pair – Create a new key pair -> Jump
  - Network Settings -> edit
    - VPC – My VPC
    - Subnet – Public
    - Security Group -> Create a new security group

Type	Port	Source Type	Source	Description
ssh	22	My IP	152.58.236.64/32	Jump

- Launch Instance



Now we need to create a **Database server**

To create database server:

- Go to instances
- Launch instance

Give the below configurations:

- Name – Database
- AMI – Amazon Linux 2023 (Free Tier Eligible)
- Instance Type – t2.micro
- Key Pair – Create a new key pair -> Name : Database
- Network Settings -> edit
  - VPC – My VPC
  - Subnet – Private
  - Security Group Rules

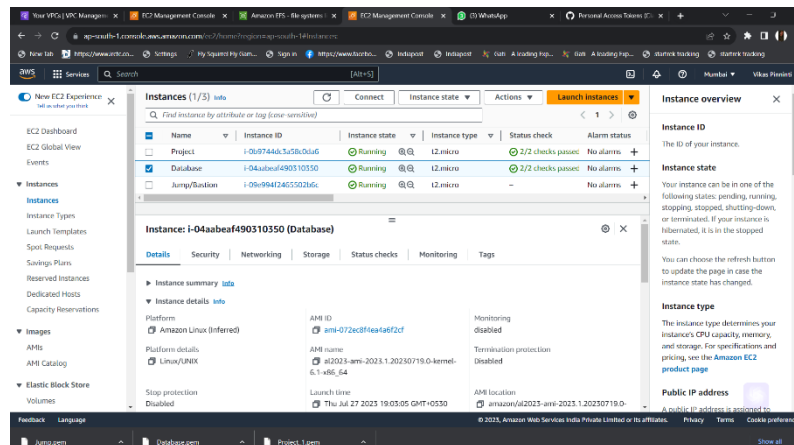
Type	Port	Source Type	Source	Description
ssh	22	Custom	Private Ip of Jump box	Jump
My SQL	3306	Custom	IPv4 CIDR range of Public Subnet	Public Subnet

**Note:**

- Here, My SQL port is opened for public subnet to pull data from database
- ssh port is opened for jump box to connect safely to database server

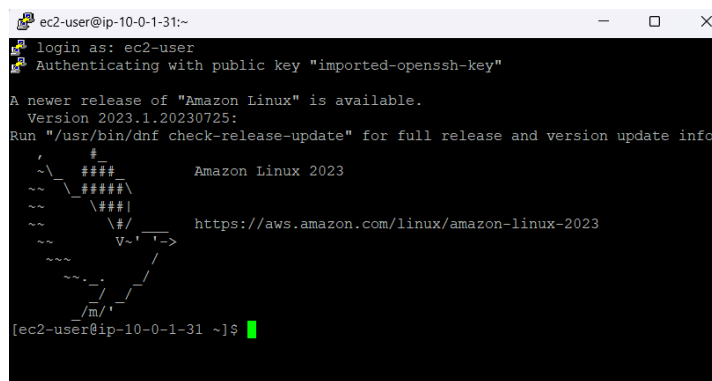
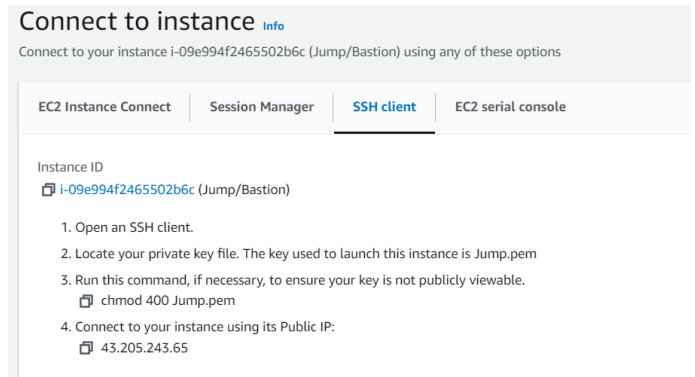
Launch Instance

Database Server Launched



## Connect to database server with Jump box

- Select jump box in instances and click on connect
- Goto SSH Client and Copy the Public IP
- Open Puttygen and convert the key pair file to .ppk
- Now, open putty and paste the public ip in host
- And go to connection -> SSH - > auth -> credentials
- Upload your private key file there and connect to instance

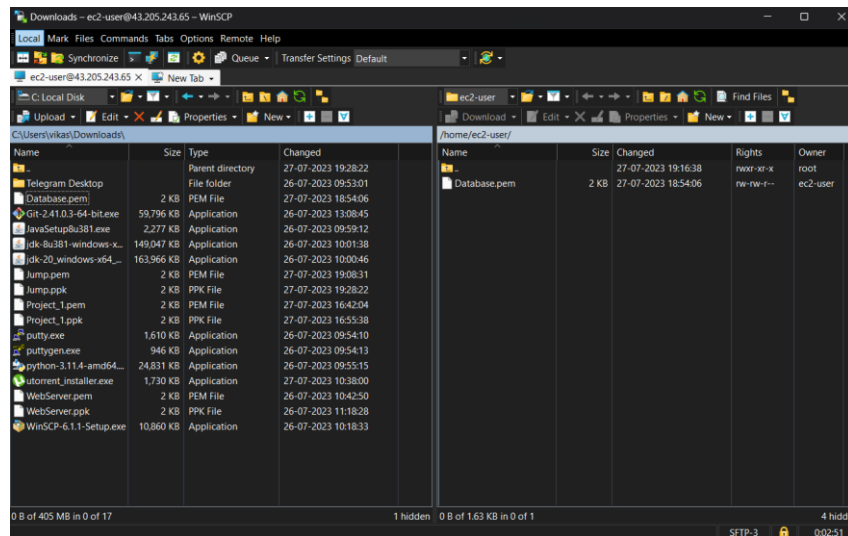


**Note:** We need to connect to Database server from here, so we need database keypair in Jump box

Now, open **winscp** (which is used to transfer key pair from local pc to instance)

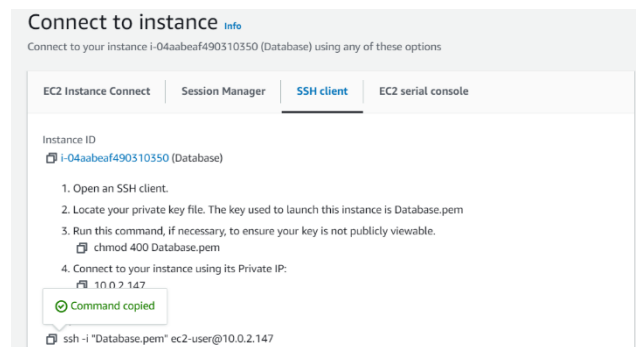
Give the following credentials

- Host: Public ip of jump box
- Username: ec2-user
- Advanced Settings -> SSH -> Authentication and Upload Private key
- Login to Winscp
- Now just drag the key file and paste it in the Jump Box
- We have successfully transferred key file
- Now terminate winscp

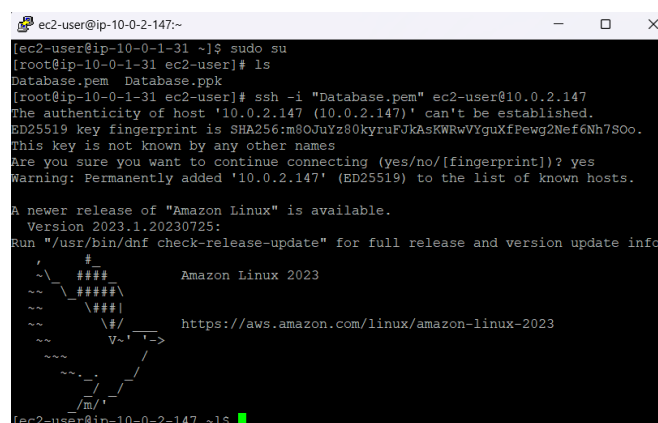


Now go to instances page in AWS Console

- Select Database Server and click on connect
- Go to ssh client and copy the example from it



Now go to jump putty instance and paste the code in the putty



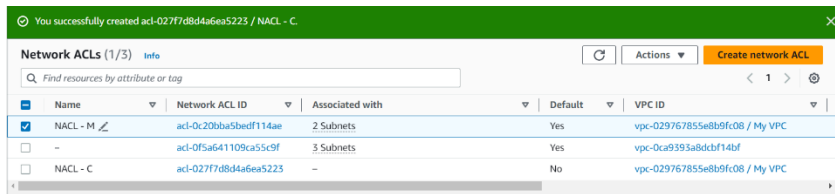
We have successfully connected to Database using Jump Box



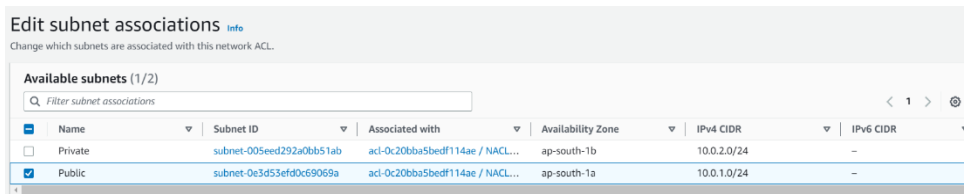
Now, go to VPC in AWS console

Go to Security-> Network ACL's

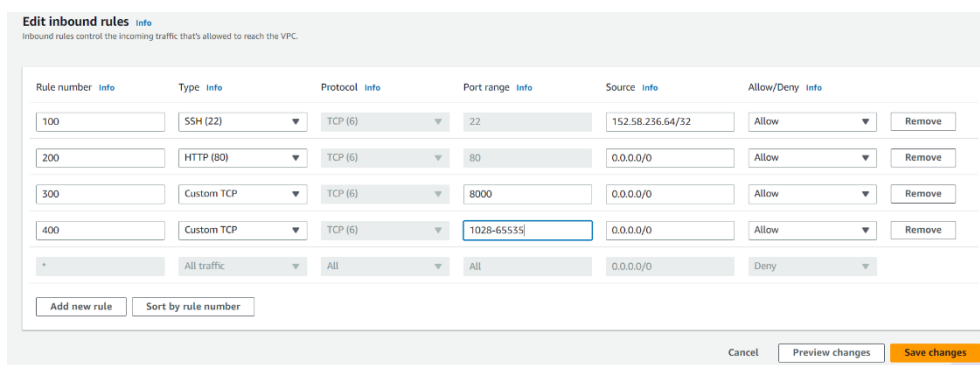
- Create a NACL
  - Name – NACL – C
  - VPC -My VPC



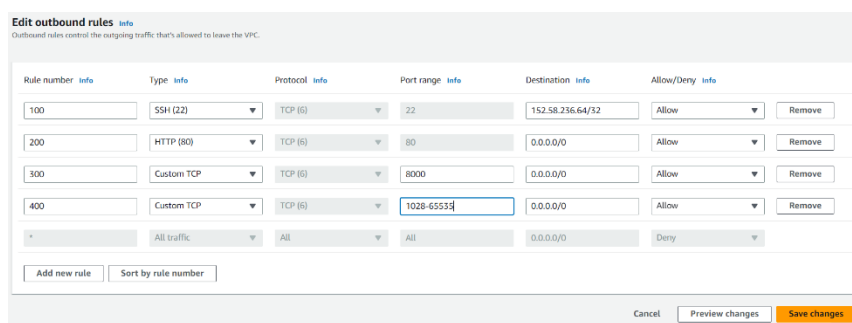
- Edit Subnet Associations and select Public Subnet



- Go to inbound rules and add the inbound rules given to security groups in Webserver or ASG



- Give the same outbound rules



**Note:** Here, we use ephemeral ports to prevent hacking at subnet level.

Now, connect the web server and connect to it using putty

- Go to instances(running)
- Select the web server and connect the web server
- Go to ssh client and copy the Public IP

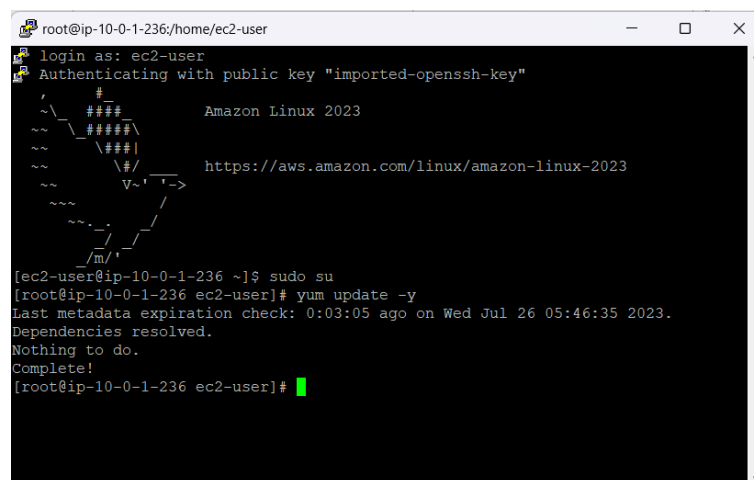
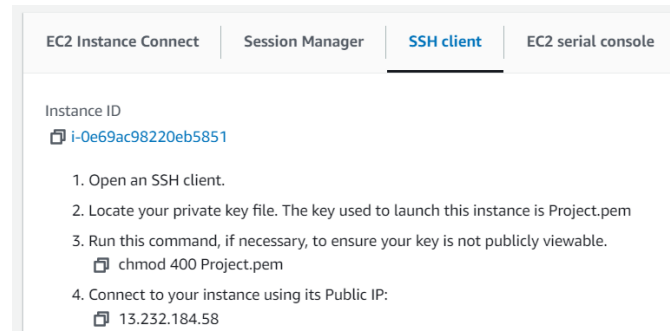
Use **Puttygen** to change the key file from **.pem** to **.ppk**

Open Puttygen

- Click on Load
- And load the Project.pem file
- Now, click on save private key and save with same name.

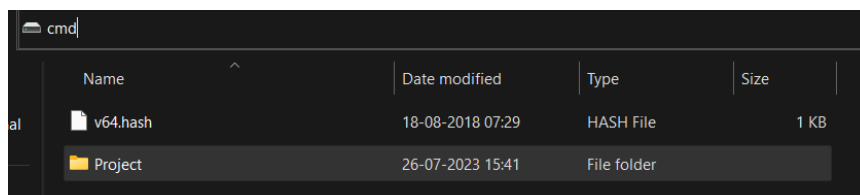
Now, open putty

- Paste the IP Address (copied from instance) in the Host Name Box.
- Now goto Connection -> ssh -> auth -> credentials
- And browse for the Project.ppk file
- Click on connect
- Username: ec2-user



Now, we create our project with Django

- Check whether python is installed in your computer else install python.
- Create a folder for Project in local disk and launch command prompt from the created folder



- Now, install a virtual environment  
**Command:** pip install virtualenvwrapper-win

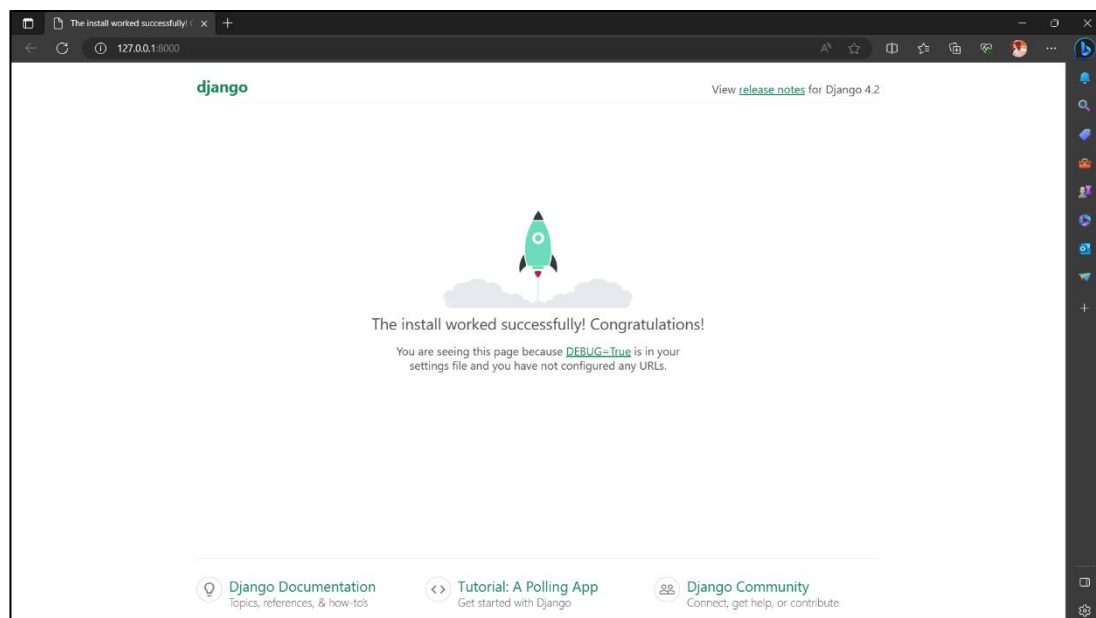
- Install Django  
**Command:** pip install django
- Create a project  
**Command:** django-admin startproject <projectname>
- Now, a project is created now we should create an app
- To create an app  
**Command:** python manage.py startapp <appname>
- Now, we should run the server properly  
**Command:** python manage.py runserver

```
C:\Windows\System32\cmd.e x + v
[notice] To update, run: python.exe -m pip install --upgrade pip
D:\Project>django-admin startproject BOV
D:\Project>python manage.py startapp Vikas
python: can't open file 'D:\Project\manage.py': [Errno 2] No such file or directory
D:\Project>cd BOV
D:\Project\BOV>python manage.py startapp Vikas
D:\Project\BOV>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin,
auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
July 26, 2023 - 16:03:26
Django version 4.2.3, using settings 'BOV.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

[26/Jul/2023 16:04:44] "GET / HTTP/1.1" 200 10664
Not Found: /favicon.ico
[26/Jul/2023 16:04:45] "GET /favicon.ico HTTP/1.1" 404 2107
[26/Jul/2023 16:05:00] "GET / HTTP/1.1" 200 10664
```



Now, to develop the app we use visual studio code. We should open root folder from Visual Studio Code.

Open Root Folder in Visual Studio Code.

Now, create two folders inside application folder using visual studio code

- static: In which we store all Images, css and javascript files
- template: In which we store all html files
- Now, go to project file and edit urls.py

```
from django.contrib import admin
from django.urls import path

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('Vikas'))
]
```

- Now, go to app and create a file named urls.py and give the below code inside it.

```
from django.urls import path
from . import views
urlpatterns=[
    path('',views.index,name='index')
]
```

- Open views.py in app folder and edit the following

```
from django.shortcuts import render
def index(request):
    return render(request,'index.html')
```

- Now, create a folder named 'templates' in Project folder which contains the source code

```
BOV > Template > index.html > ...
1 <div class="container">
2   <div class="column add-bottom">
3     <div id="mainwrap">
4       <div id="nowPlay">
5         <span id="npAction">Paused...</span><span id="npTitle"></span>
6       </div>
7       <div id="audiowrap">
8         <div id="audio0">
9           <audio id="audio1" preload controls>Your browser does not support HTML5 Audio! 🙄</audio>
10        </div>
11        <div id="tracks">
12          <a id="btnPrev">&larr;</a><a id="btnNext">&rarr;</a>
13        </div>
14      </div>
15      <div id="plwrap">
16        <ul id="plList"></ul>
17      </div>
18    </div>
19  </div>
20  <div class="column add-bottom center">
21    <p>Music by <a href="http://www.mythium.net/">Mythium</a></p>
22    <p>Download: <a href="https://archive.org/download/mythium/mythium_vbr_mp3.zip">zip</a> / <a href="https://archive.org/download/m
23  </div>
24 </div>
25
```

- Go to settings.py in Project folder and edit Templates section

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR, 'templates'],
    },
]
```

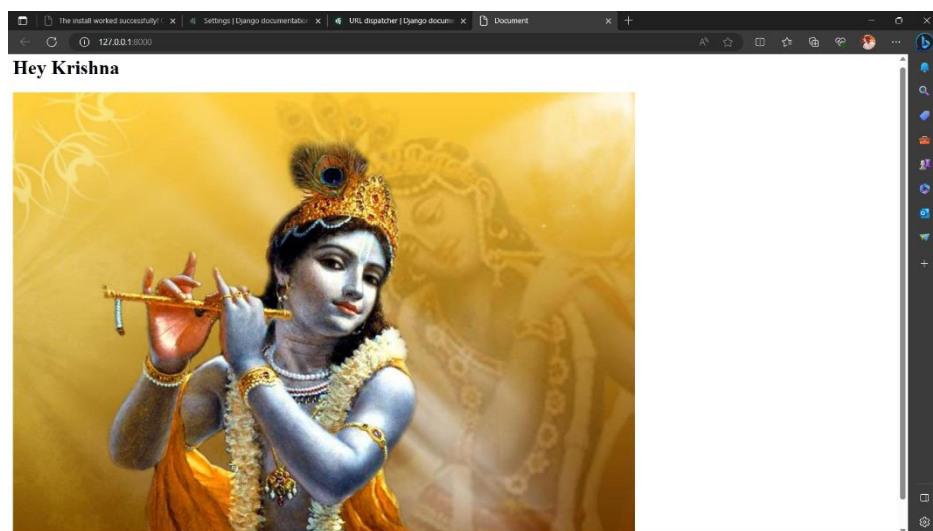
- Go to settings and Change allowed hosts

```
ALLOWED_HOSTS = ['3.111.197.195', 'local_host', '0.0.0.0',
                  '127.0.0.1', 'local_host', '0.0.0.0']
```

- Now run the server

```
PS D:\Project\Project> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
July 26, 2023 - 19:49:17
Django version 4.2.3, using settings 'Project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```



The code process is over. Now we need to upload the project file to **github**.

To upload the project open github from project file location.

And run the following commands:

- git init
- git status
- git add .
- git commit -m "Anyname"
- git remote (repository address)
- git push -u origin main/master

```
vikas@Vikas MINGW64 /d/Project (master)
$ git commit -m Vikas

vikas@Vikas MINGW64 /d/Project (master)
$ git remote add origin https://github.com/vikaspininti13/AWS.git

vikas@Vikas MINGW64 /d/Project (master)
$ git push -u origin master
Enumerating objects: 28, done.
Counting objects: 100% (28/28), done.
Delta compression using up to 2 threads
Compressing objects: 100% (24/24), done.
Writing objects: 100% (28/28), 11.52 KiB | 1.05 MiB/s, done.
Total 28 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), done.
To https://github.com/vikaspininti13/AWS.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

A screenshot of a terminal window titled "ec2-user@ip-10-0-1-50:~". The prompt shows the user is logged in as "ec2-user". The message "Authenticating with public key 'imported-openssh-key'" indicates the authentication process. A large ASCII art logo of a cat is displayed on the left side of the screen. To the right of the cat, the text "Amazon Linux 2023" is shown. Below the cat, the URL "https://aws.amazon.com/linux/amazon-linux-2023" is visible. At the bottom, the message "Last login: Wed Jul 26 09:59:21 2023 from 152.58.197.7" is printed, followed by the command prompt "[ec2-user@ip-10-0-1-50 ~]\$" and a green cursor.

```
[root@ip-10-0-1-50 ec2-user]# git clone https://github.com/vikaspinninti13/AWS.git
Cloning into 'AWS'...
remote: Enumerating objects: 35, done.
remote: Counting objects: 100% (35/35), done.
remote: Compressing objects: 100% (25/25), done.
remote: Total 35 (delta 7), reused 30 (delta 4), pack-reused 0
Receiving objects: 100% (35/35), 12.37 KiB | 6.18 MiB/s, done.
Resolving deltas: 100% (7/7), done.
[root@ip-10-0-1-50 ec2-user]#
```

```
[root@ip-10-0-1-50 AWS]# yum install python
```

- Install pip

**Command:** yum install python3-pip

```
[root@ip-10-0-1-50 AWS]# yum install python3-pip
```

- Install Django

**Command:** yum install django

```
[root@ip-10-0-1-50 AWS]# yum install django
```

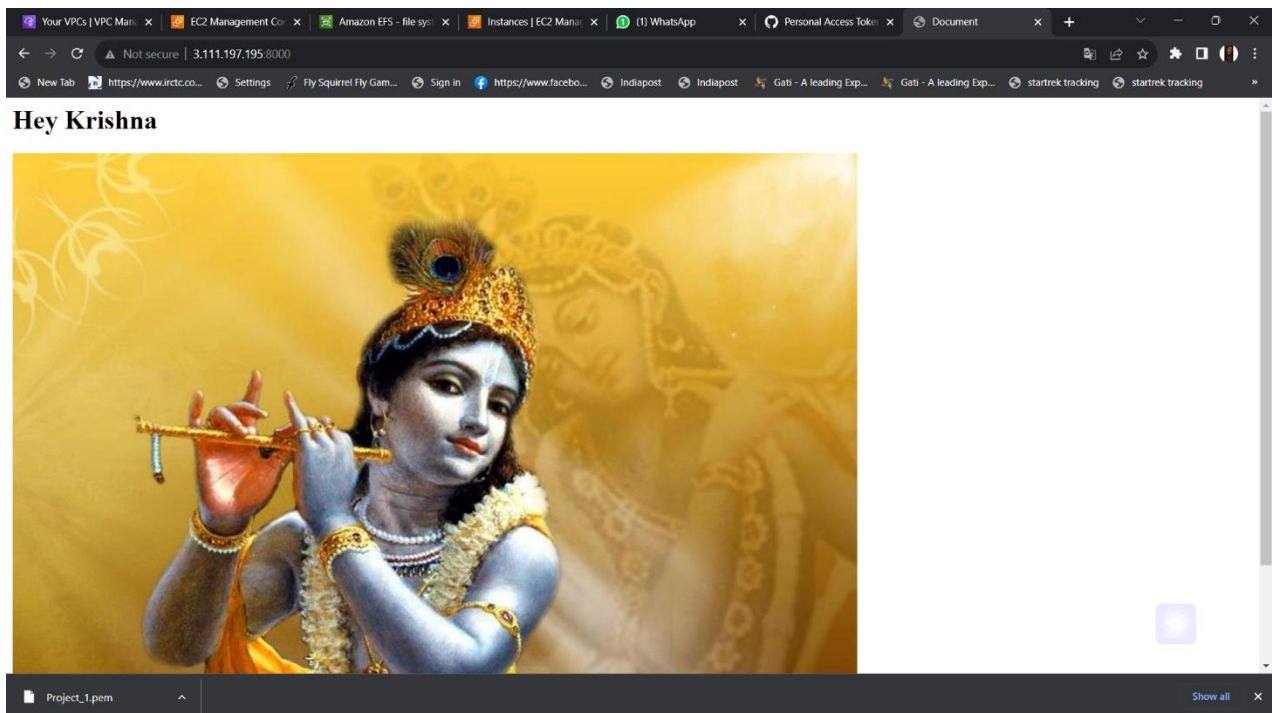
- Start the server

**Command:** python manage.py runserver 0.0.0.0:8000

```
[root@ip-10-0-1-50 Project]# python manage.py runserver 0.0.0.0:8000
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
July 26, 2023 - 15:08:47
Django version 4.2.3, using settings 'Project.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CONTROL-C.
```

Output:





## Conclusion:

From this project I had learned to create a basic html page using Python-Django in EC2 instance using my private network which is called VPC. And I have learned how to protect a website from hacking and we use NACL there. I have learned acquired great knowledge and skill by doing this project and rectified many errors which increased confidence in me.

Firstly, I had created a **Virtual Private Network** to host my project and after that I had created a web server using **ASG** which helps to create server automatically if any server is crashed. To create ASG I used Launch Template, **Load Balancer**. Load balancer helps us to balance the load to server and it is the only way to route traffic to server. I had created a **database in my private subnet** and connected to it by using **jump box** for secure login. And after creating the Web server I had created a **Django project** in my local repository by using command prompt and after creating a project I had created an application and developed it in **Visual Studio Code**. After all that I have used **git bash** and uploaded my project into my **Github** Repository. Now, I had connected my Web server and **clone** my project into my web server instance by using some commands. And finally, I have installed Python and Django in cloned repository and run the server successfully.

**Signature of Mentor**