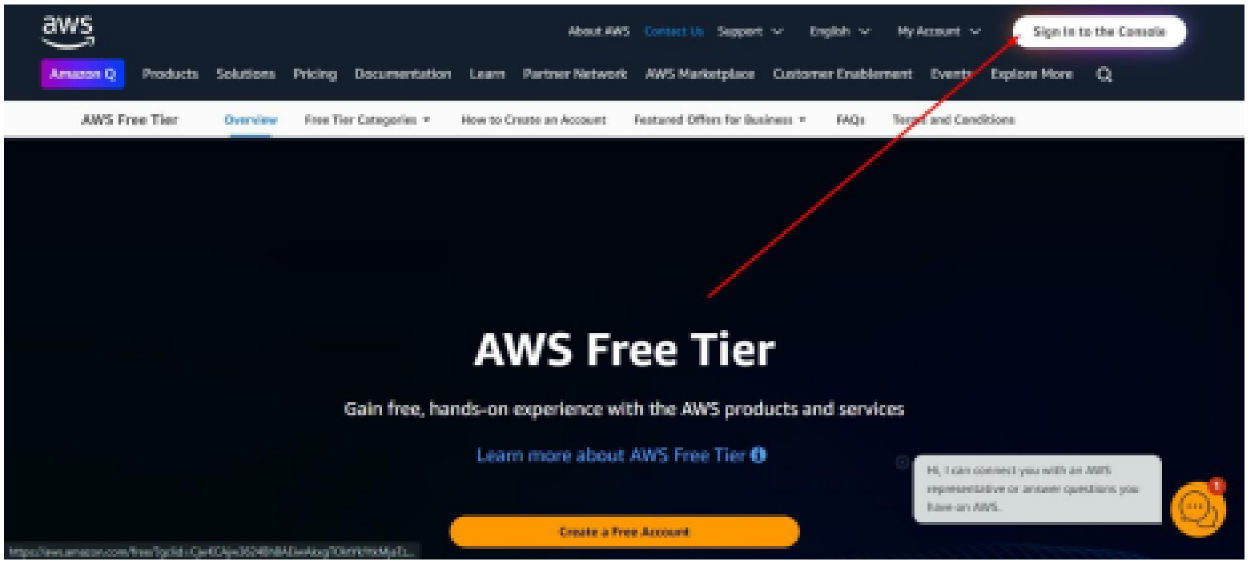
Launching an EC2 instance and hosting a server

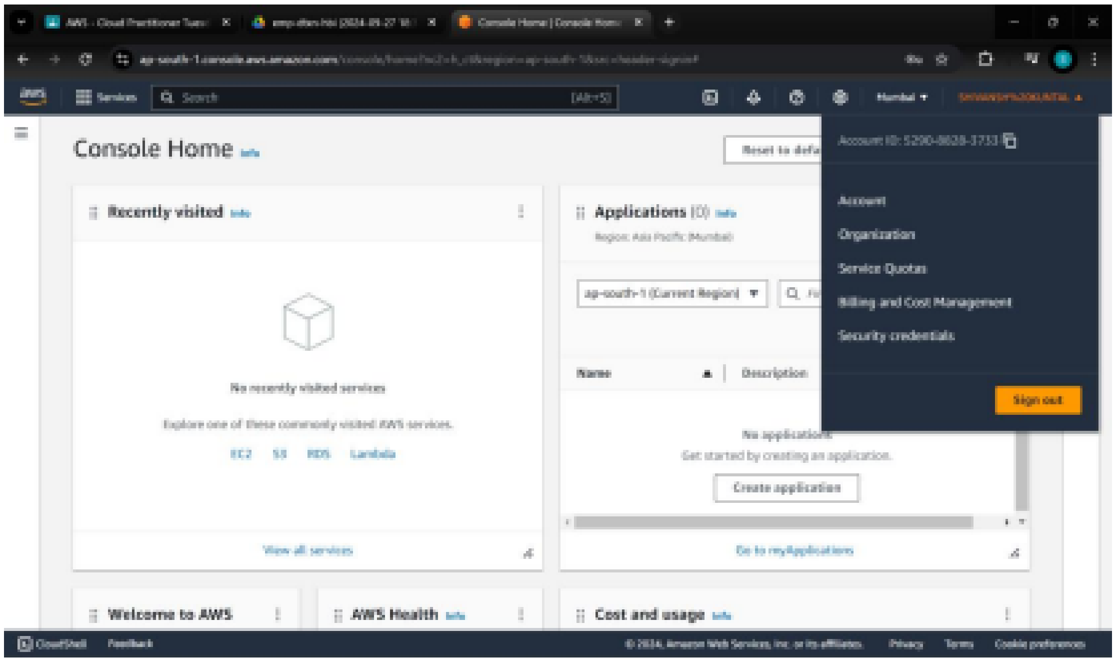
NAME: SHIVANSH KUNTAL

STEP 1:

Logging Into Your AWS Console by Creating a Root User Account.

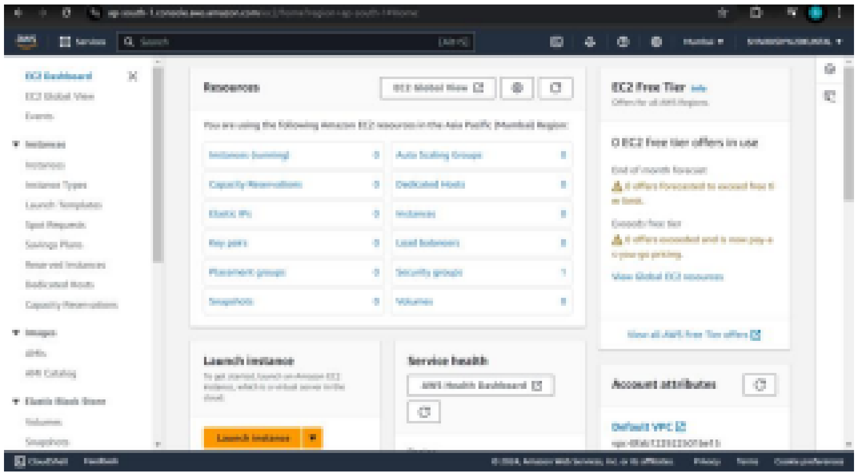


STEP 2:



After logging into the AWS Management Console, I navigated to the **Compute** section and selected **EC2** (Elastic Compute Cloud) to manage virtual servers. From the **EC2 Dashboard**, I could launch instances by selecting the desired OS, instance type, and configurations,

STEP 3:

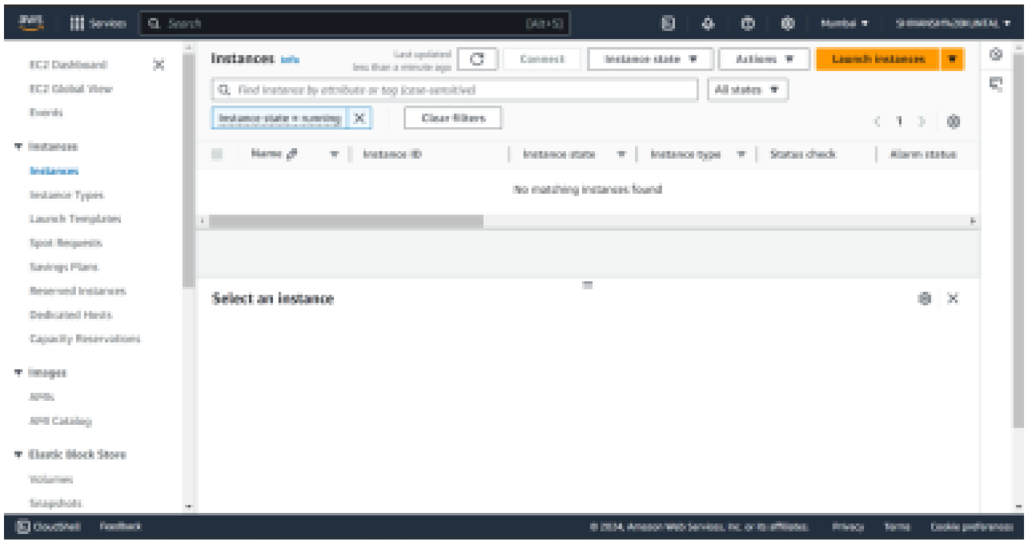


After selecting **EC2**, I was directed to the **EC2 Dashboard** homepage, displaying an overview of instances, key pairs, security groups, and more. The dashboard offers quick access to features like launching instances, managing volumes, and configuring security, making it easy to create and manage virtual servers for my project .

Choose Mumbai as region on top right.

.

STEP 6:



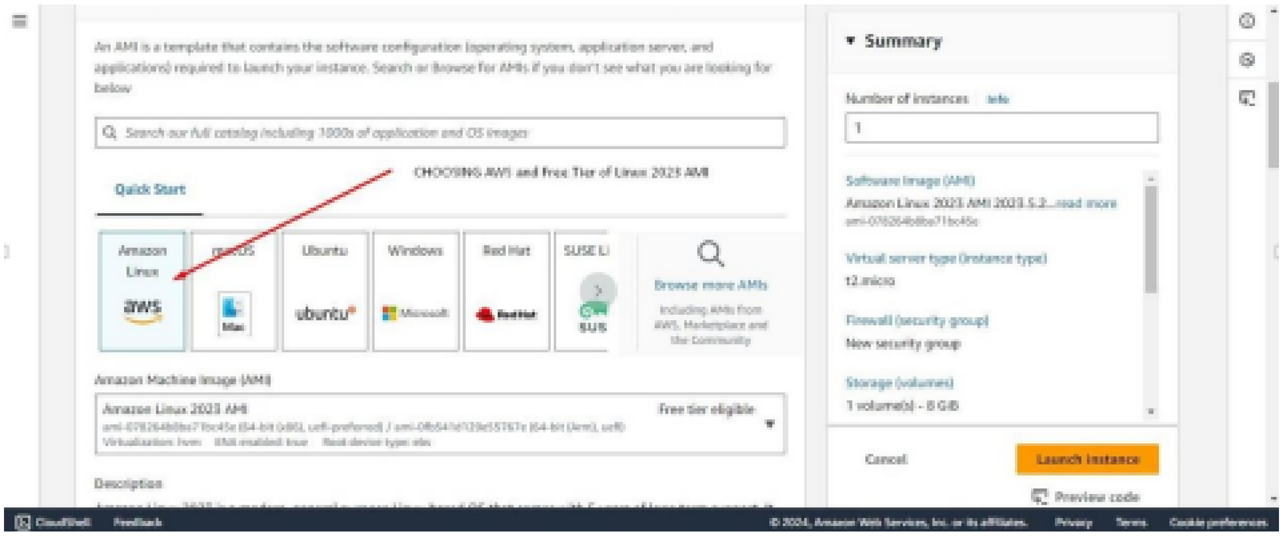
After reviewing the running instances, I clicked **Launch Instances** on the Instances page to start creating a new virtual server. This took me to the configuration screen, where I selected key specifications like the Amazon Machine Image (AMI), instance type, and network settings, initiating the process of setting up the server to host my website.

STEP 7:

Created instance named as 22BCY10290-LINUX

.

STEP 8:


An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

CHOOSING AWS and Free Tier of Linux 2023 AMI

Search our full catalog including 1000s of application and OS images
Quick Start
Amazon Linux
macOS
Ubuntu
Windows
Red Hat
SUSE Li
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community
Summary
Cancel
Launch instance
Preview code
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

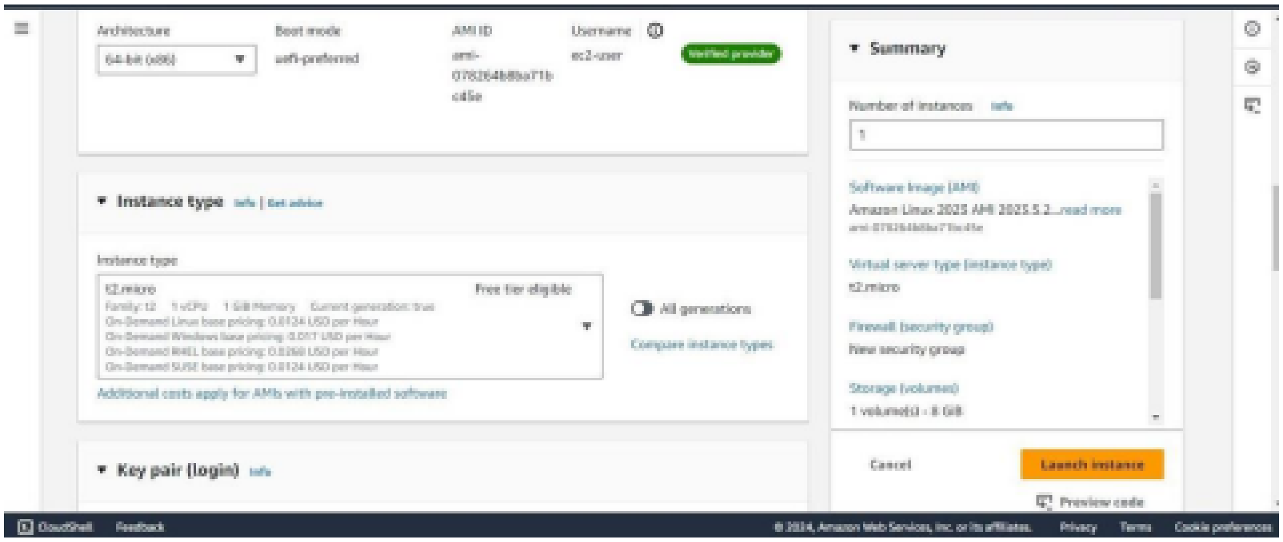
Amazon Machine Image (AMI)
Amazon Linux 2023 AMI — Free tier eligible
ami-078264b8ba71bc45e (64-bit (x86), uefi-preferred) / ami-0fb541d129e55767e (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Number of instances: 1
Software Image (AMI): Amazon Linux 2023 AMI 2023.5.2...read more
ami-078264b8ba71bc45e
Virtual server type (instance type): t2.micro
Firewall (security group): New security group
Storage (volumes): 1 volume(s) - 8 GiB

Next, I proceeded to choose the **Amazon Machine Image (AMI)**. To stay within the free tier, I selected the **AWS Free Tier Eligible** option and chose the **Linux 2023 AMI**.

STEP 9:



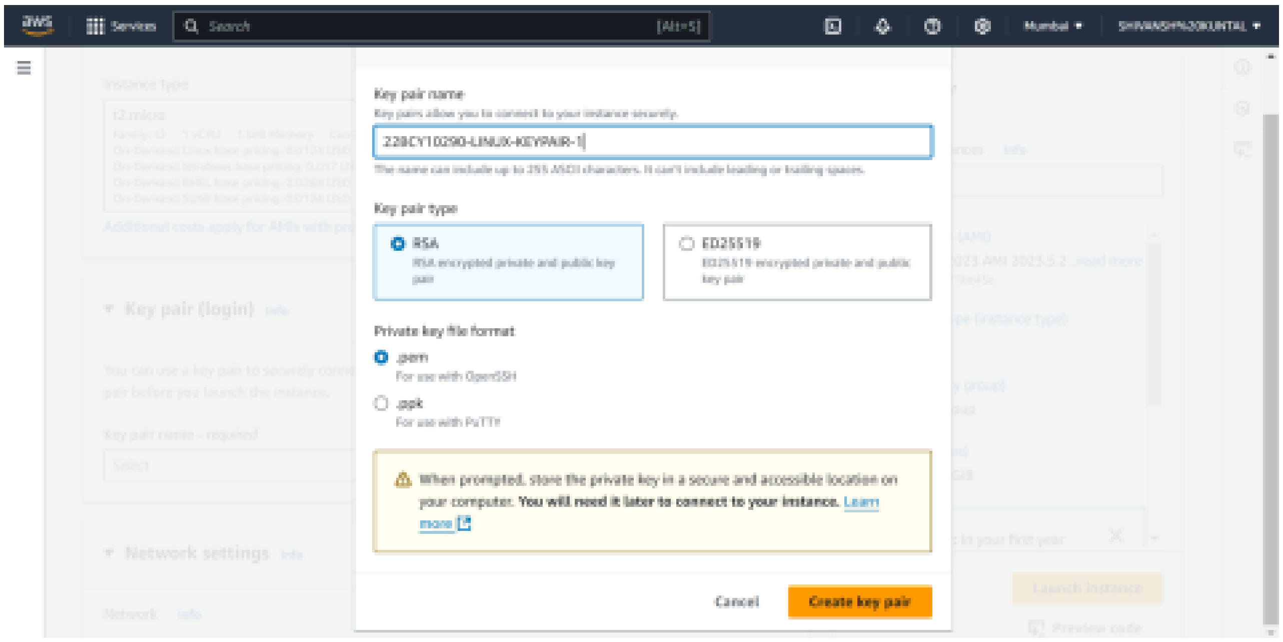
Architecture: 64-bit (x86)
Boot mode: uefi-preferred
AMI ID: ami-078264b8ba71bc45e
Username: ec2-user

Instance type
t2.micro — Free tier eligible
Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Linux base pricing: 0.0124 USD per Hour
On-Demand Windows base pricing: 0.017 USD per Hour
On-Demand RHEL base pricing: 0.0268 USD per Hour
On-Demand SUSE base pricing: 0.0124 USD per Hour
Additional costs apply for AMIs with pre-installed software

Number of instances: 1
Software Image (AMI): Amazon Linux 2023 AMI 2023.5.2...read more
ami-078264b8ba71bc45e
Virtual server type (instance type): t2.micro
Firewall (security group): New security group
Storage (volumes): 1 volume(s) - 8 GiB

Next, I proceeded to **choose the instance type** for my EC2 instance. I selected **t2.micro**, which is a popular choice within the AWS Free Tier. The **"t2"** designation refers to the instance family optimized for burstable performance, which means it can handle varying workloads efficiently. The **"micro"** indicates the smallest size within this family, providing 1 vCPU and 1 GB of memory. This configuration is ideal for lightweight applications and testing environments.

STEP 10:


Key pair name

Key pairs allow you to connect to your instance securely.

22BCY10290-LINUX-KEYPAIR-1

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

[✓ RSA] RSA encrypted private and public key pair

ED25519 ED25519 encrypted private and public key pair

Private key file format

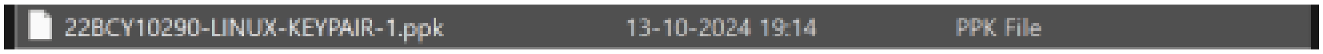
[✓ .pem] For use with OpenSSH

.ppk For use with PuTTY

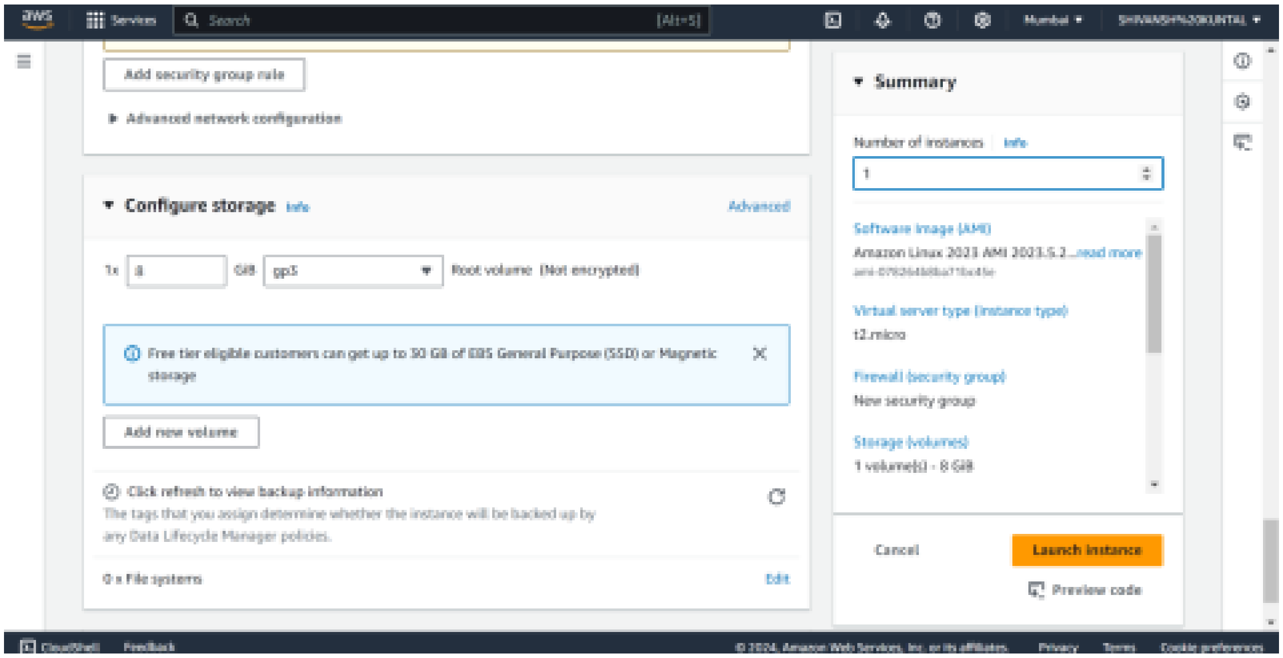
When prompted, store the private key in a secure and accessible location on your computer. **You will need it later to connect to your instance.** Learn more

After selecting the instance type, I moved on to the next step: **creating a key pair for login**

I created a key pair named **22BCY10290-LINUX-KEYPAIR-1** for secure access to my EC2 instance. This RSA-type key pair enables secure data transmission and is commonly used for SSH access to servers. The private key is saved in the **.ppk** format, designed for use with PuTTY, a popular SSH client for Windows. This .ppk file contains the private key necessary for authenticating and establishing a secure connection to my EC2 instance. It's crucial to keep this file safe and not share it, as it grants access to my instance. With this key pair created, I can securely log in to my EC2 instance using SSH, ensuring an encrypted and protected connection.


22BCY10290-LINUX-KEYPAIR-1.ppk 13-10-2024 19:14 PPK File

STEP 11:


Configure storage

1x 8 GiB gp3 Root volume (Not encrypted)

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage

Click refresh to view backup information

The tags that you assign determine whether the instance will be backed up by any Data Lifecycle Manager policies.

0 x File systems

Summary

Number of instances: 1

Software Image (AMI): Amazon Linux 2023 AMI 2023.5.2...read more

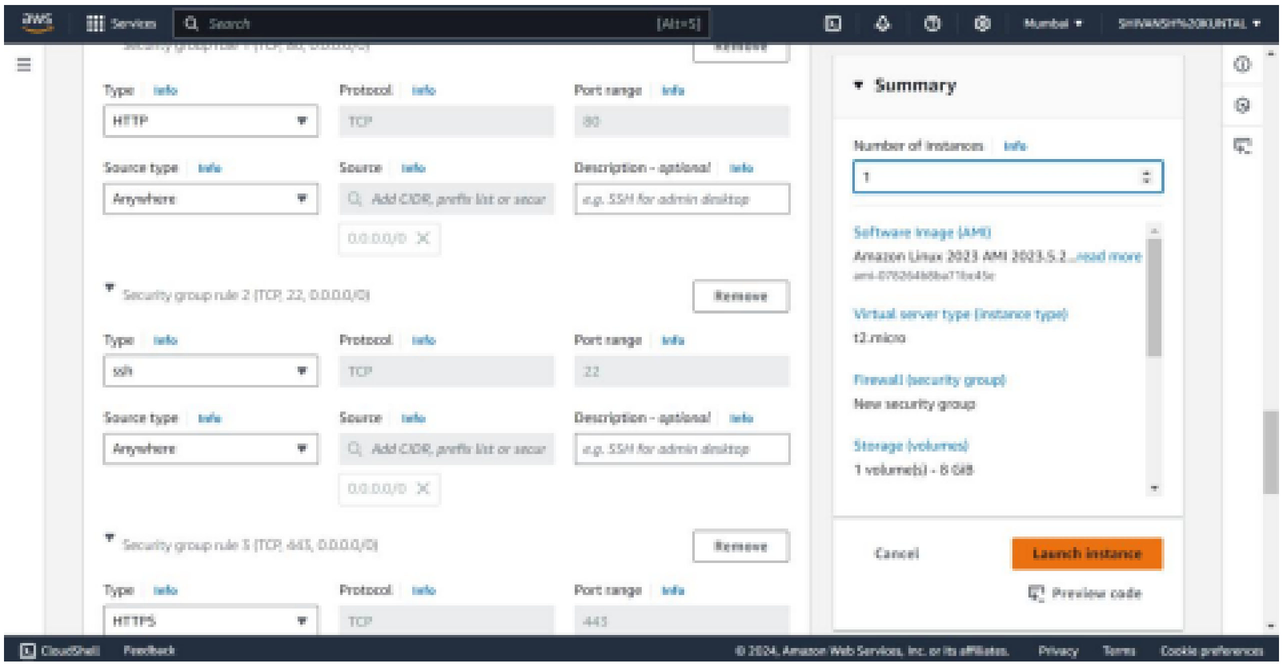
Virtual server type (instance type): t2.micro

Firewall (security group): New security group

Storage (volumes): 1 volume(s) - 8 GiB

CONFIGURE STORAGE

STEP 12:



Security group rule 1 (TCP, 80, 0.0.0.0/0)

Type: HTTP
Protocol: TCP
Port range: 80
Source type: Anywhere
Source: 0.0.0.0/0
Description - optional: e.g. SSH for admin desktop

Security group rule 2 (TCP, 22, 0.0.0.0/0)

Type: ssh
Protocol: TCP
Port range: 22
Source type: Anywhere
Source: 0.0.0.0/0
Description - optional: e.g. SSH for admin desktop

Security group rule 3 (TCP, 443, 0.0.0.0/0)

Type: HTTPS
Protocol: TCP
Port range: 443

Summary

Number of instances: 1

Software image (AMI)
Amazon Linux 2023 AMI 2023.5.2...read more
ami-0702648bba71bc45c

Virtual server type (instance type)
t2.micro

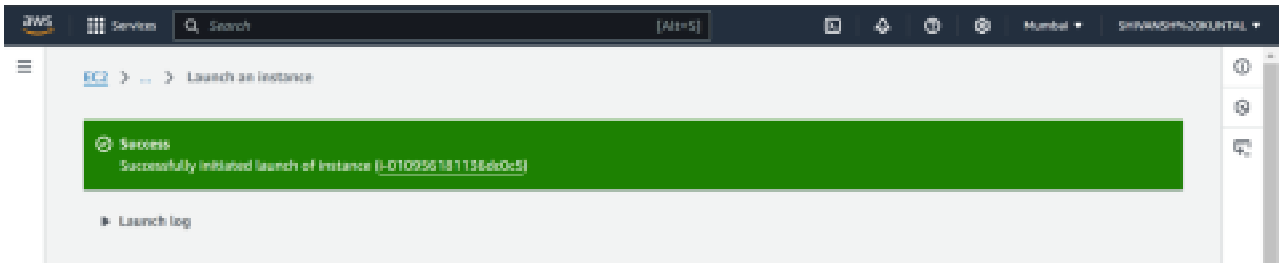
Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

I created a security group named **Aman Security Group** for my web server to manage access to my EC2 instance. I set the type to **SSH** with the source type as **Anywhere (0.0.0.0/0)**, allowing remote management from any IP address.

Next, I added two inbound rules: one for **HTTP (port 80)** to enable web traffic to my static website and another for **HTTPS (port 443)** to ensure secure web traffic, encrypting data exchanged between users and my site. These configurations allow access for both secure and non secure web traffic while maintaining SSH management capabilities.

STEP 13:



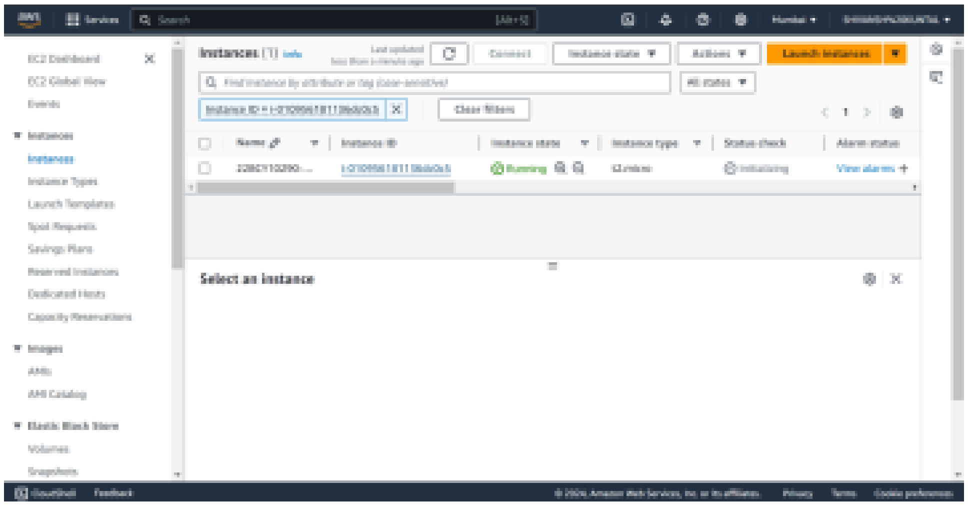
EC2 > ... > Launch an instance

Success
Successfully initiated launch of instance (i-010956181158dc0c5)

Launch log

Successfully launched our instance after configuring it according to our requirements.

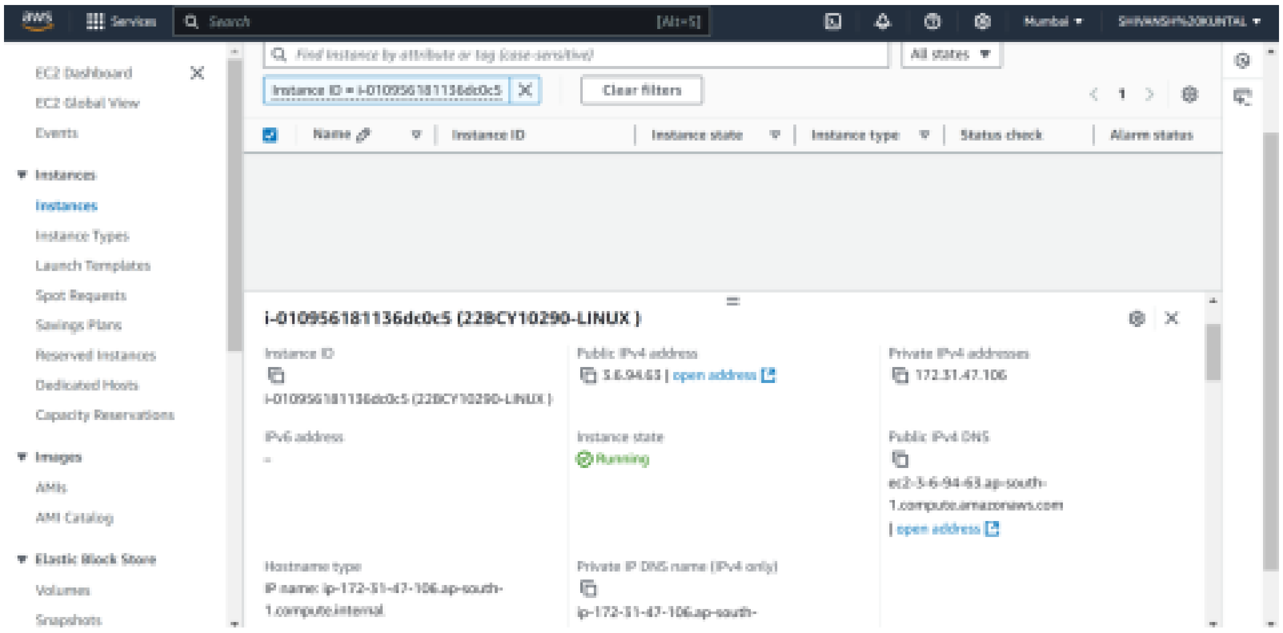
STEP 14:



	Name	Instance ID	Instance state	Instance type	Status check	Alarm status
	22BCY10290-...	i-010956181136dc0c5	Running	t2.micro	Initializing	View alarms +

Select an instance

STEP 15:



i-010956181136dc0c5 (22BCY10290-LINUX)

Instance ID: i-010956181136dc0c5 (22BCY10290-LINUX)

Public IPv4 address: 3.6.94.63 | open address

Private IPv4 addresses: 172.31.47.106

IPv6 address: –

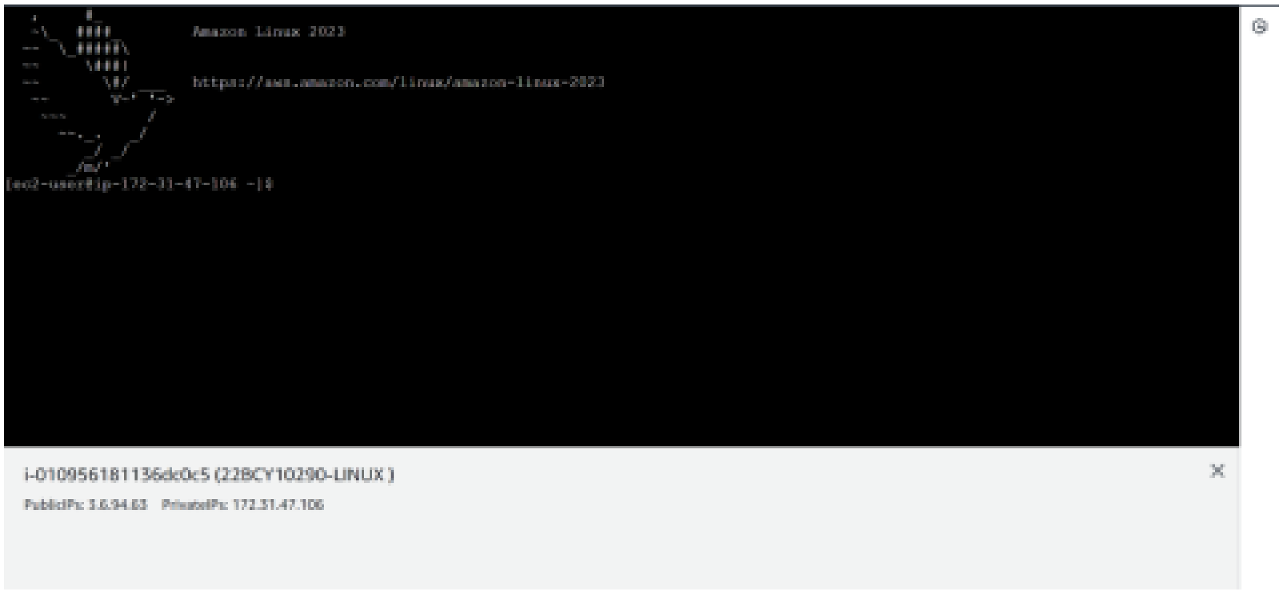
Instance state: Running

Public IPv4 DNS: ec2-3-6-94-63.ap-south-1.compute.amazonaws.com | open address

Hostname type: IP name: ip-172-31-47-106.ap-south-1.compute.internal

Private IP DNS name (IPv4 only): ip-172-31-47-106.ap-south-

STEP 16:



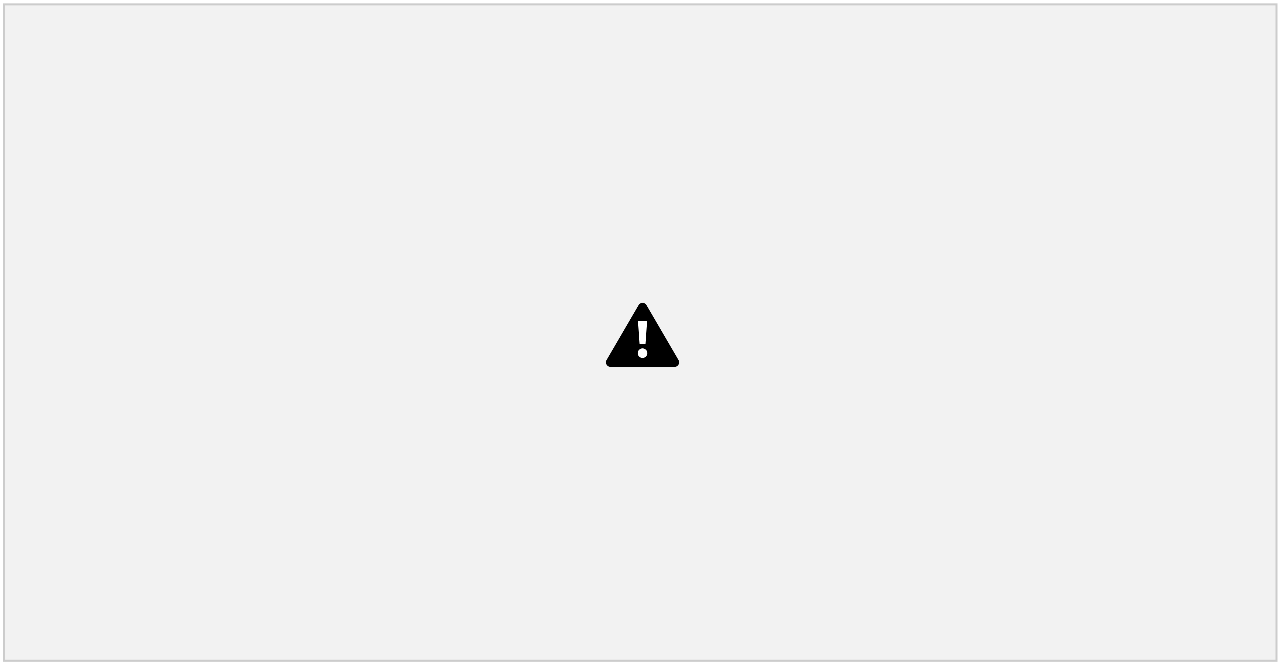
```
        Amazon Linux 2023

        https://aws.amazon.com/linux/amazon-linux-2023

[ec2-user@ip-172-31-47-106 ~]$
```

i-010956181136dc0c5 (22BCY10290-LINUX)

PublicIPs: 3.6.94.63   PrivateIPs: 172.31.47.106

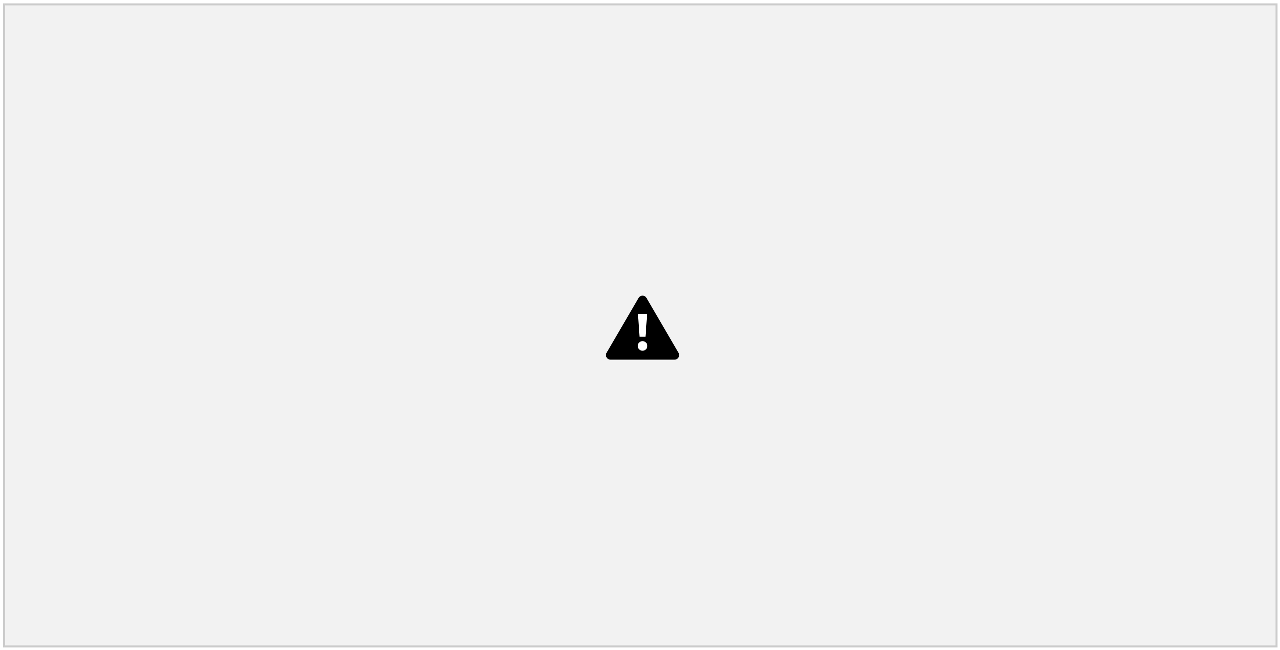
Our server is up and running. We can view our public ip and private ip for reference.

STEP 17:



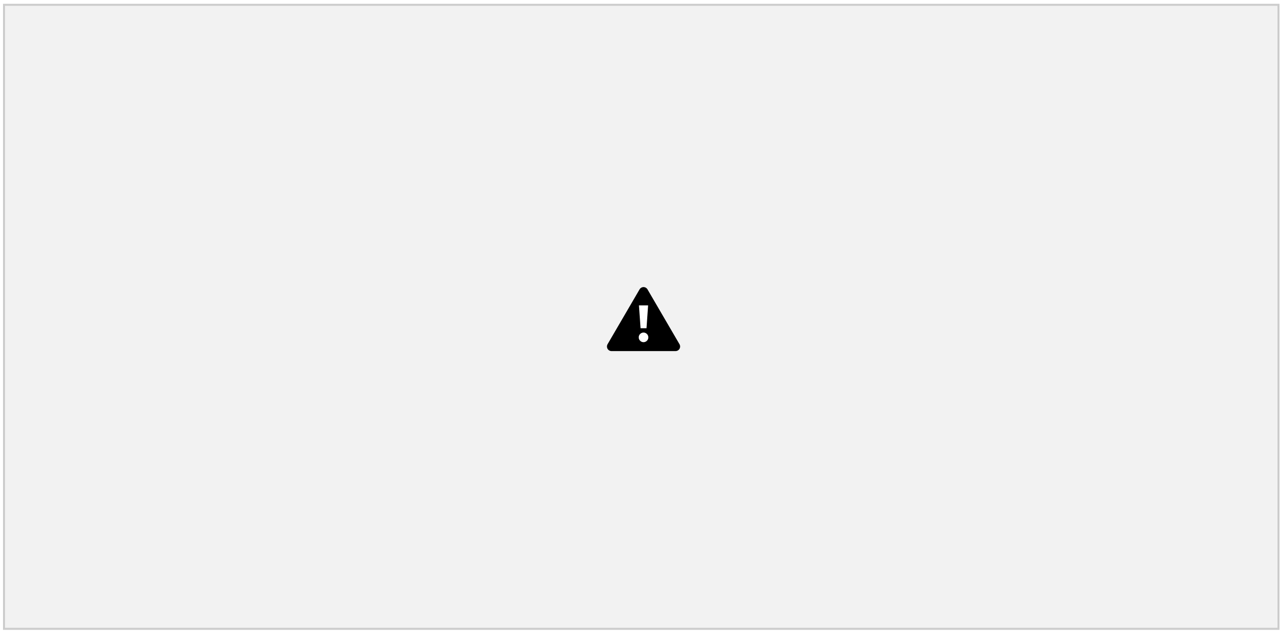
DOWNLOADED Apache http server

STEP 18:



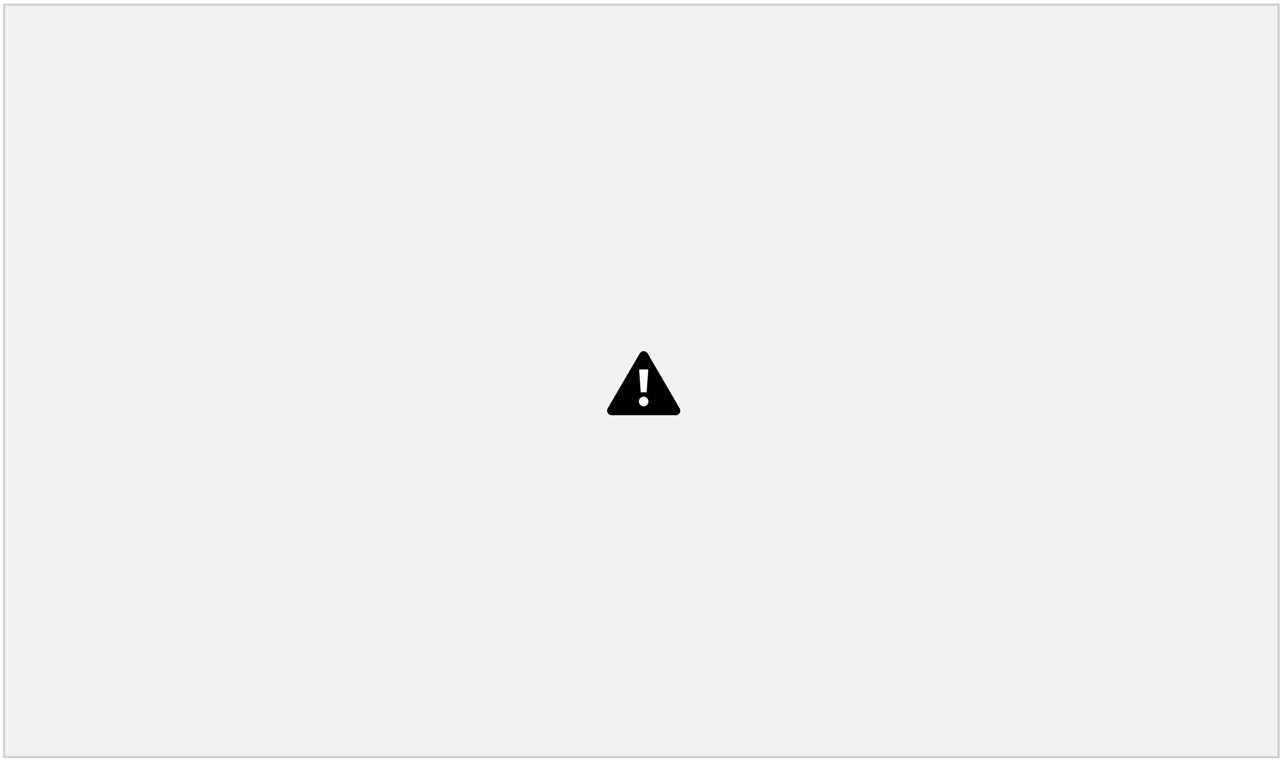
DOWNLOADED APPLICATION TO DEPLOY USING FREE-CSS .COM

WEBSITE
STEP 19:



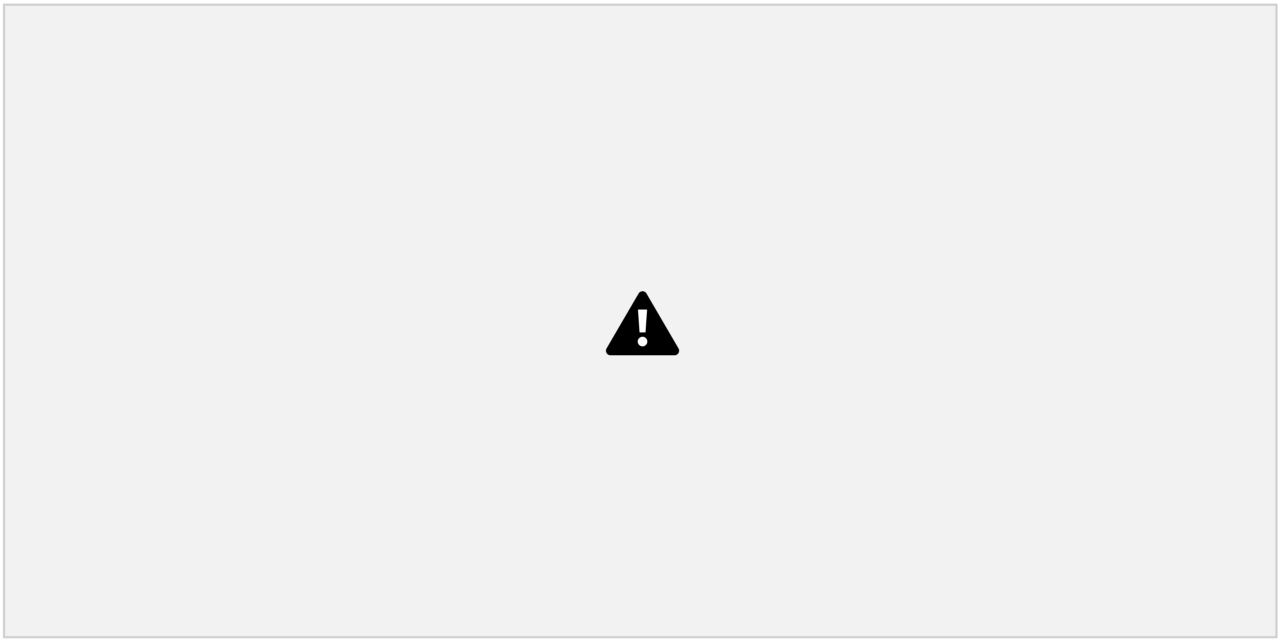
Enabled apache http server to deploy our web application

STEP 20:

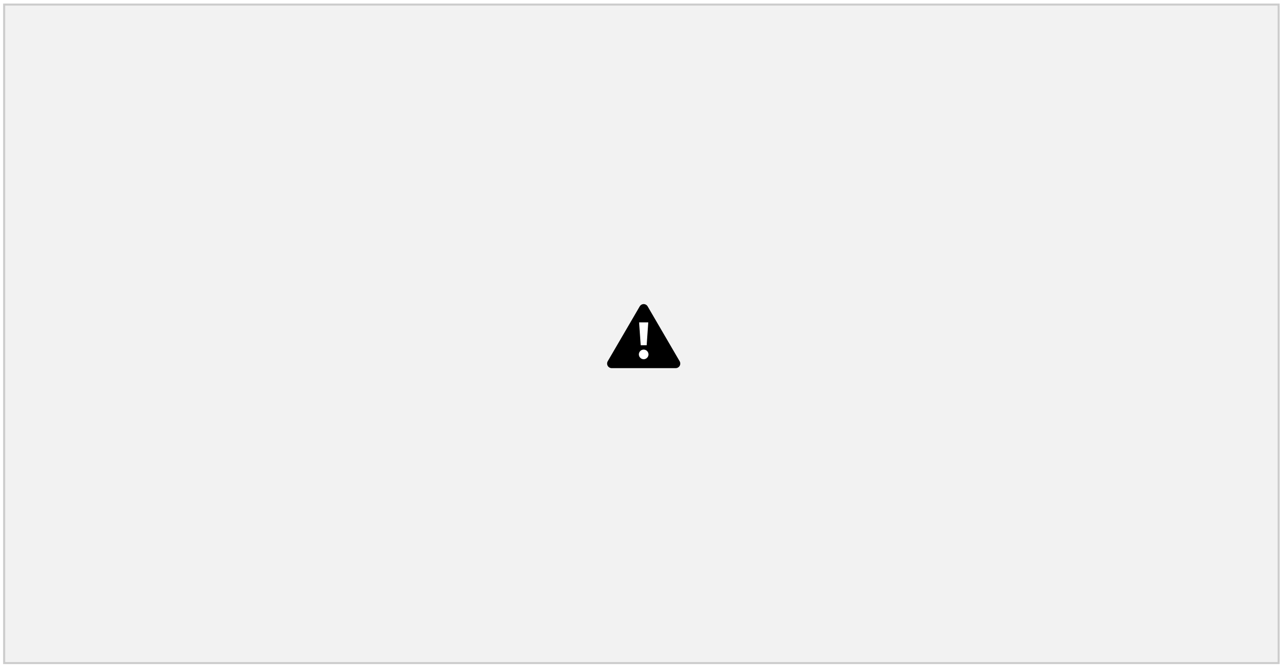


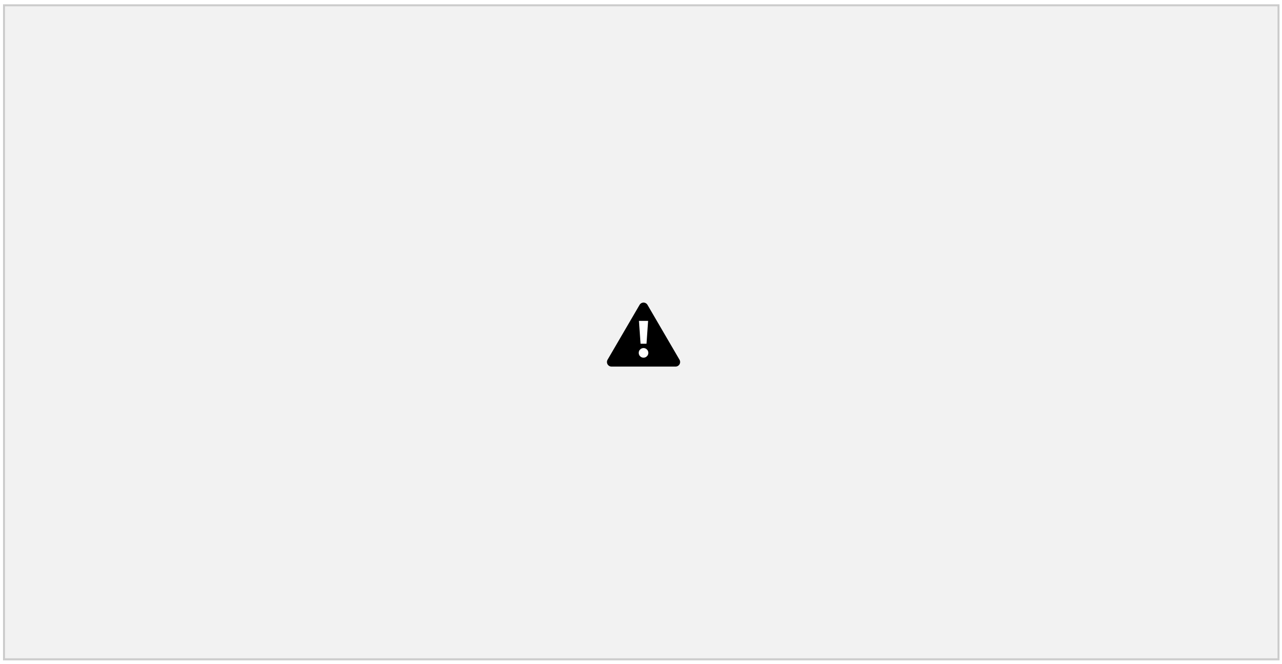
ACCESSED OUR WEB APP USING OUR PUBLIC IP ASSIGNED : 3.6.94.63

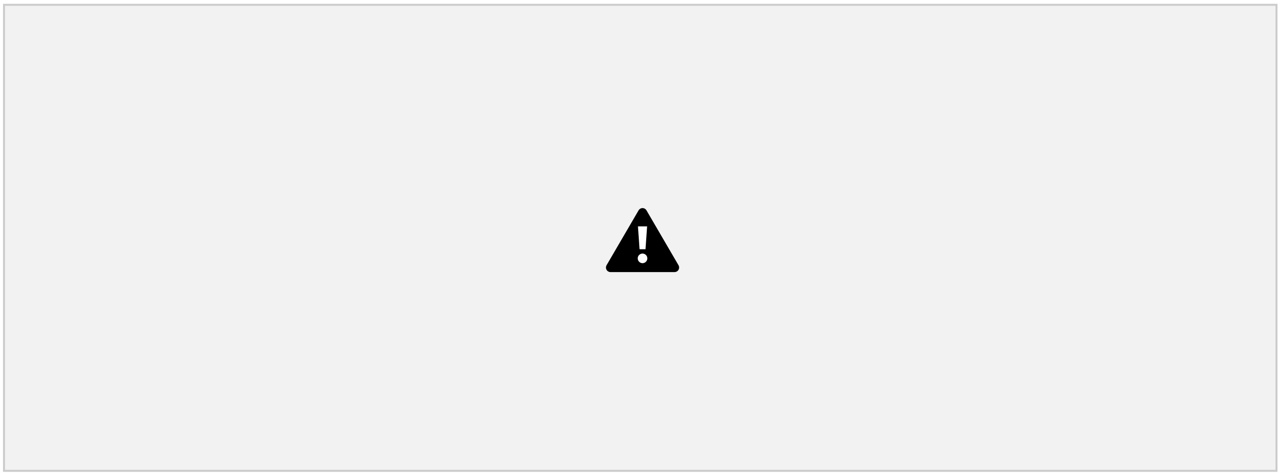
STEP 21:



STOPPING OUR INSTANCE AND TERMINATING IT.







CONCLUSION

In conclusion, I successfully launched and configured an EC2 instance to host a static website using Apache. The process involved setting up the server environment, downloading a static template, and verifying that the website was accessible via the instance's IP address. I also learned how to manage the instance by starting, stopping, and understanding the implications of dynamic IP addresses. This hands-on experience enhanced my understanding of cloud infrastructure and web hosting. Overall, the project demonstrated the power and flexibility of AWS in deploying web applications.