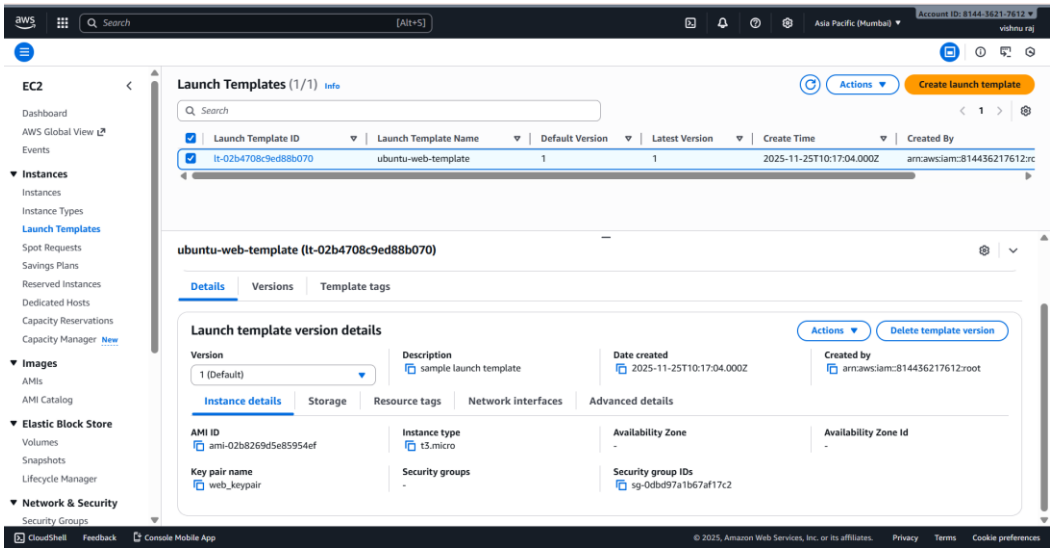
AWS

ASG

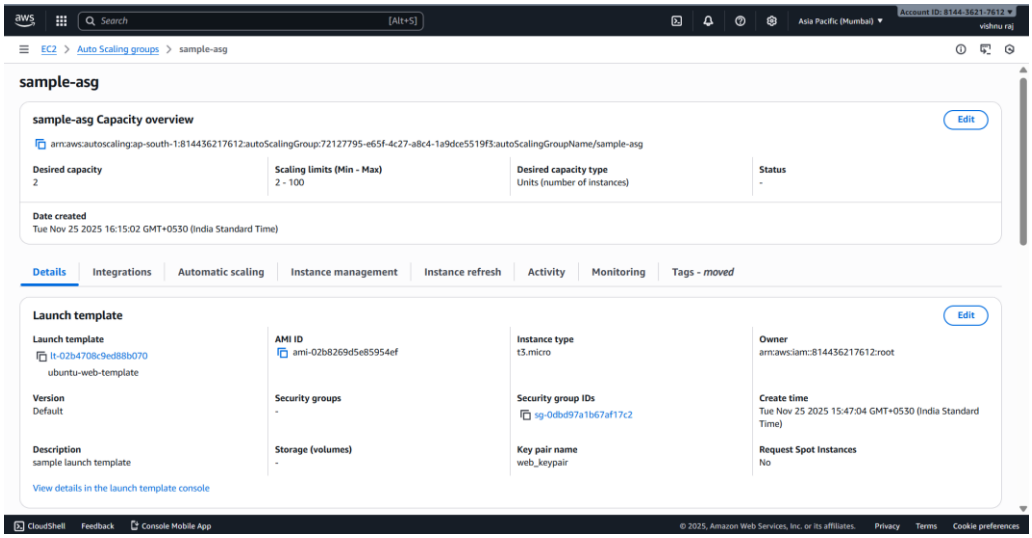
1. **Create a Launch Template**
    - AWS Console → EC2 → Launch Templates → **Create launch template**
    - Template Name: ubuntu-web-template
    - AMI: Ubuntu Server 24.04 LTS and choose: t2.micro
    - Select key pair
    - Choose the **EC2 security group**
    - Scroll to *Advanced details* → *User data*, paste:

      #!/bin/bash
      apt update
      apt install -y apache2
      systemctl start apache2
      systemctl enable apache2
      echo "<h1>Server Details</h1> Hostname: $(hostname) IP Address: $(hostname -I | cut -d\" \" -f1) " > /var/www/html/index.html

    - Click **Create launch template**



2. **Create an Auto Scaling Group**
    - Go to AWS Console → EC2 → Auto Scaling Groups → **Create Auto Scaling Group**
    - Provide Auto Scaling group name
    - Select the **Launch template**
    - Choose VPC & Subnets
    - Attach Load Balancer
        - Attach to an existing load balancer
        - Select **Application Load Balancer**
        - Select **Target Group**
    - Configure Group Size:
        - Desired capacity: 2
        - Minimum capacity: 2
        - Maximum capacity: 100
    - Scaling Policies: None
    - Click **Create Auto Scaling Group**

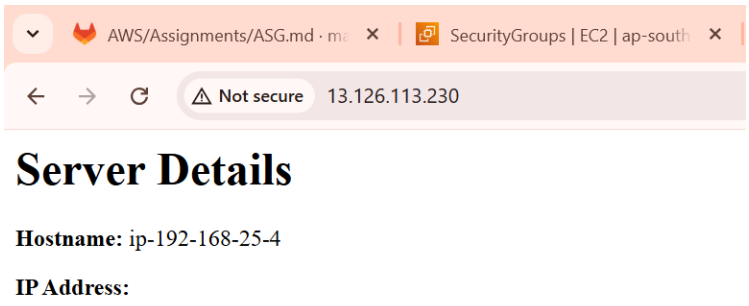


3. **Create an Application Load Balancer (ALB)**
    - AWS Console → EC2 → Load Balancers → **Create Load Balancer** → Application Load Balancer
    - Provide **Load balancer name**
    - **Scheme:** Internet-facing and **IP address type:** IPv4

- Choose your VPC and select 2 public subnets
- Create/choose a Security Group
    - Inbound: HTTP port 80
    - Outbound: All traffic
- Create target group:
    - Target type: Instances
    - Provide Target group name
    - Protocol: HTTP
    - Port: 80
    - Health Check Protocol: HTTP
    - Health Check Path: /
- Create Load Balancer

4. **Security Groups**
    - Create Load Balancer SG
        - EC2 → Security Groups → Create security group
        - Provide security group name
        - Inbound rule:
            - Type: **HTTP**, Port 80, Source: **0.0.0.0/0**
        - Create
    - Create EC2 Instance SG
        - EC2 → Security Groups → Create security group
        - Provide security group name
        - Inbound rule:
            - Type: **HTTP**, Port 80
            - Source: **choose** Load Balancer SG
        - Create

5. **Test the setup**



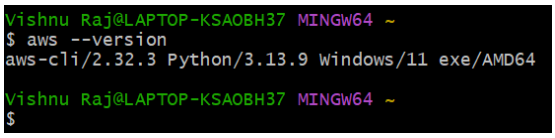
CLI

1. **AWS CLI Installation**
    - Linux

      curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"

      unzip awscliv2.zip

      sudo ./aws/install
    - Verify:
        - aws –version



2. **IAM User Creation (via AWS Console or CLI)**
    - Go to **IAM → Users → Add user**
    - Enable **Programmatic access**
    - Attach policies directly → **AdministratorAccess** → Create user
    - Navigate to **IAM → Users**
    - Click your username
    - Click the **Security credentials** tab
    - Scroll to **Access keys** and click **Create access key**
    - Select **Programmatic access**
    - Download **Access Key ID** and **Secret Access Key**
    - Click **Create**

3. **AWS CLI Configuration**

- Set up CLI authentication using the IAM credentials

  aws configure

  # Enter the following:

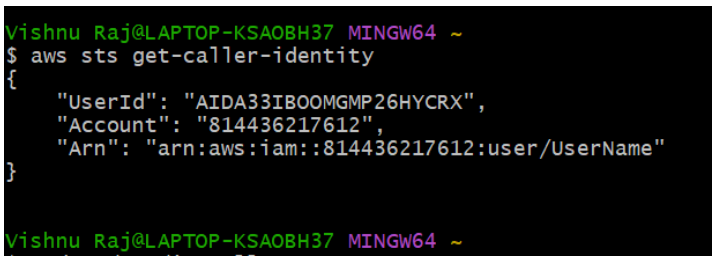
  # AWS Access Key ID [None]: <Your AccessKeyID>

  # AWS Secret Access Key [None]: <Your SecretAccessKey>

  # Default region name [None]: us-east-1

  # Default output format [None]: json

- Validate configuration:
  - aws sts get-caller-identity



4. **Create AWS Resources Using CLI**

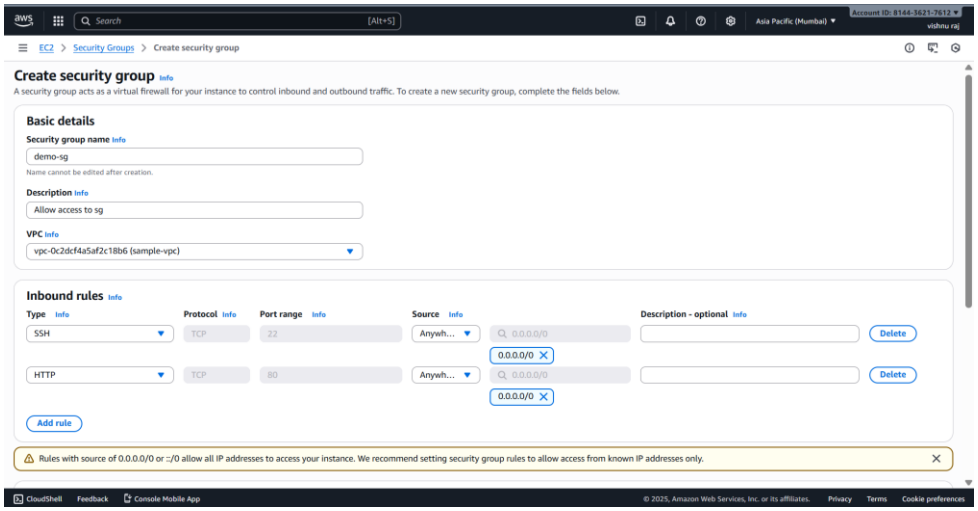
a) EC2 Instance

- Go to **EC2 → Instances → Launch instances**
- Provide instances name
- Provide a name for **EC2 Launch**
- AMI: **Amazon Linux 2**
- Instance type: **t3.micro**
- Provide a name for **Key Pair**
- Provide security group

b) Security Group

- Create a **Security Group:**
  - Go to EC2 Console → Network & Security → Security Groups → Create security group
  - Provide a name for **Security Groups**
  - VPC: Select your custom VPC

- Add **Inbound Rules**:
  - Inbound SSH traffic on **port 22**
  - Inbound HTTP traffic on **port 80**
- Outbound Rules: **Allow all traffic**
- Click **Create security group**



- Attach this security group to the EC2 instance
  - EC2 → Instances → Select instance → Actions → Security → Change Security Groups → Add Security Group

c) EBS Volume

- Create a Volume
  - Go to EC2 → Elastic Block Store → Volumes → Create volume
  - Size: **8 GiB or more**
  - Volume type: gp3
  - Availability Zone: **MUST match your instance's AZ**
  - Click **Create Volume**
- Attach the volume to the EC2 instance
  - Select the volume → **Actions → Attach Volume**
  - Choose your EC2 instance
  - Device name: /dev/sdf
  - Click Attach

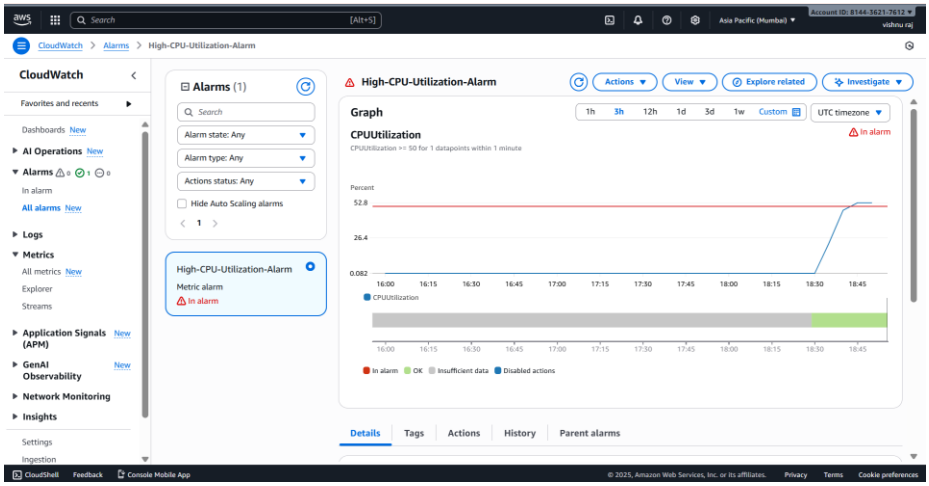
d) Create an AMI

- Go to AWS Console → **EC2 → Instances**
- Click **Actions → Image and Templates → Create Image**
- Provide image name
- Click **Create Image**

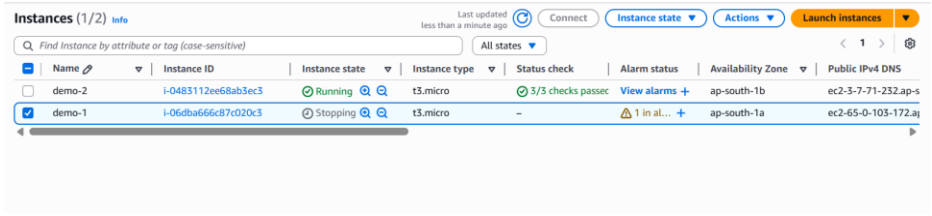
CloudWatch

1. **CPU Utilization Alarm**
    - Go to **CloudWatch → Alarms → Create alarm**
    - Choose metric:
        - Browse → **EC2** → **Per-Instance Metrics**
        - Select **CPUUtilization** for your instance
        - Select Metric
    - Conditions:
        - **Threshold type:** Static
        - **Greater than** 50
        - **Period:** 1 minute
        - **Datapoints required:** 5 of 5
    - Notification:
        - Create new topic → give mail ID → create topic
    - Name your alarm:
        - High-CPU-Utilization-Alarm
    - Create Alarm

CPU Utilization Alarm



EC2 instance automatically stop after the CPU Utilization hit 50%



2. Disk Usage Alarm

- Configure CloudWatch Agent for Custom Disk Usage Metric
  - Install CloudWatch Agent

    sudo apt update

    sudo apt install amazon-cloudwatch-agent -y

  - Create the configuration file:

    sudo nano /opt/aws/amazon-cloudwatch-agent/bin/config.json
  - Paste this content:

```
{
  "metrics": {
    "append_dimensions": {
      "InstanceId": "${aws:InstanceId}"
    },
    "metrics_collected": {
      "disk": {
        "measurement": [
```

```
            {"name": "disk_used_percent", "rename": "DiskUsage", "unit":
        "Percent"}
          ],
          "resources": ["*"],
          "ignore_file_system_types": ["sysfs", "devtmpfs", "overlay"]
        }
      }
    }
    }
```

  - Save and exit.
- Start the CloudWatch Agent

  sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl \

  -a fetch-config \

  -m ec2 \

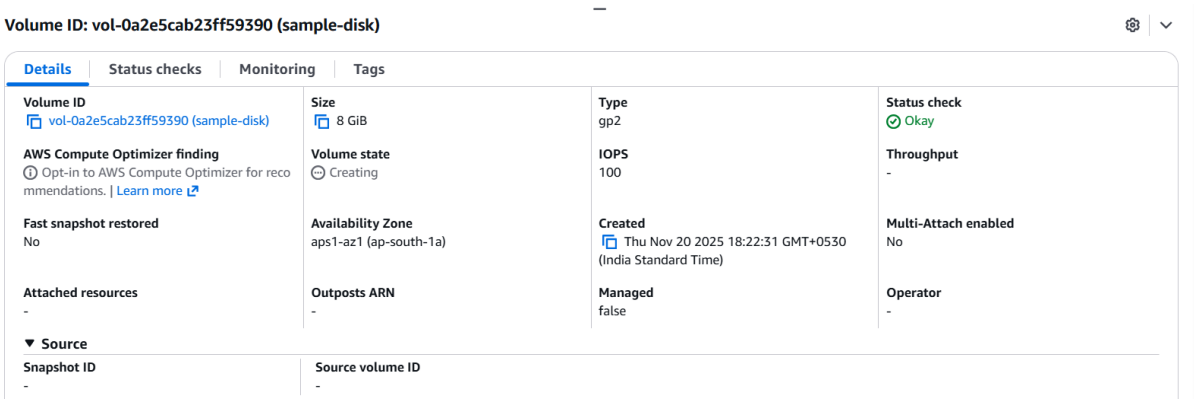
  -c file:/opt/aws/amazon-cloudwatch-agent/bin/config.json \

  -s
- Verify agent is running:

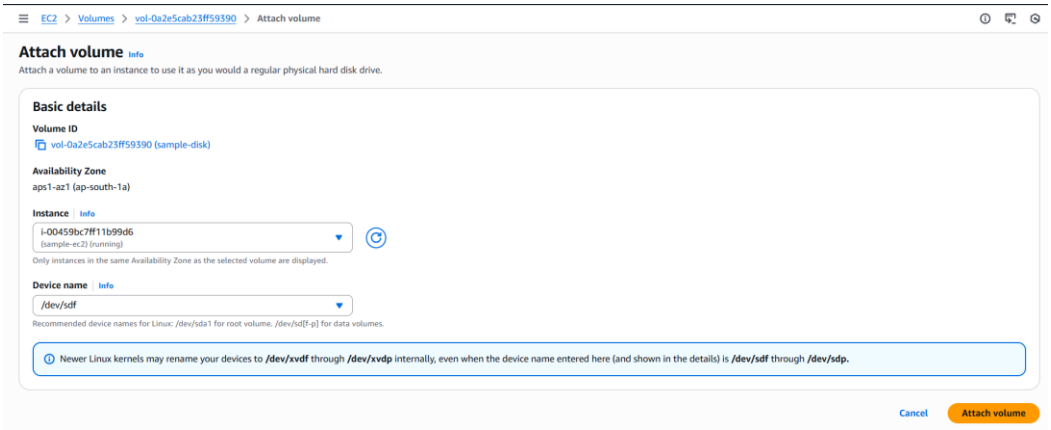
  sudo systemctl status amazon-cloudwatch-agent
- CloudWatch → Metrics → CWAgent → DiskUsage
- InstanceId: your EC2 instance
- Configure Alarm Conditions:
  - **Statistic:** Average
  - **Period:** 1 minute
  - **Threshold type:** Static
  - Whenever DiskUsage > 75%
  - **For:** 5 consecutive periods (5 minutes)
- Configure Actions
  - Alarm state trigger: **ALARM**
  - Notification: SNS topic → ProdMonitoringAlerts
- Name the Alarm
  - Name: HighDiskUsage
  - Description: Exceeds 75% for 5 consecutive minutes
- Click **Create Alarm**.

EBS

1. **EBS Volume Creation**
    - Go to **EC2 → Elastic Block Store → Volumes**
    - Click **Create Volume**
    - Choose:
        - **Volume Type:** gp2
        - **Size:** 8 GiB
        - Availability Zone: Must match the EC2 instance's AZ (e.g., ap-south-1a)
    - Click **Create Volume**



2. **Volume Attachment**
    - Select the newly created volume
    - Click **Actions → Attach Volume**
    - Select your EC2 instance
    - Specify device name
    - Attach Volume



3. Instance Configuration

- SSH into the instance:

  ssh -i "web_keypair.pem" ubuntu@<public-ip>

```
Vishnu Raj@LAPTOP-KSAOBH37 MINGW64 ~/Downloads
$ ssh -i "web_keypair.pem" ubuntu@15.206.124.85
The authenticity of host '15.206.124.85 (15.206.124.85)' can't be established.
ED25519 key fingerprint is SHA256:Daxy2aOSEC1EVAWOwaHj1ye3fbHelf+iLx7yWoRWdqI.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '15.206.124.85' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1015-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Thu Nov 20 13:05:37 UTC 2025

  System load:  0.0               Temperature:           -273.1 C
  Usage of /:   25.8% of 6.71GB   Processes:             109
  Memory usage: 22%               Users logged in:       0
  Swap usage:   0%                IPv4 address for ens5: 192.168.25.13

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-192-168-25-13:~$
```

- Format the volume:

  mkfs.ext4 /dev/nvme1n1

```
root@ip-192-168-25-13:~# mkfs.ext4 /dev/nvme1n1
mke2fs 1.47.0 (5-Feb-2023)
Found a dos partition table in /dev/nvme1n1
Proceed anyway? (y,N) y
Creating filesystem with 2097152 4k blocks and 524288 inodes
Filesystem UUID: a6bd9c0f-34b1-47b1-b817-f4d4b08c74b6
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

root@ip-192-168-25-13:~#
```

- Create mount directory:

  mount /dev/nvme1n1 /mnt/data

```
root@ip-192-168-25-13:/# mount /dev/nvme1n1 /mnt/data
root@ip-192-168-25-13:/# df -l
Filesystem      1K-blocks    Used Available Use% Mounted on
/dev/root         7034376 1862948   5155044  27% /
tmpfs              468076       0    468076   0% /dev/shm
tmpfs              187232     904    186328   1% /run
tmpfs                5120       0      5120   0% /run/lock
efivarfs              128       4       120   3% /sys/firmware/efi/efivars
/dev/nvme0n1p16    901520   91032    747360  11% /boot
/dev/nvme0n1p15    106832    6250    100582   6% /boot/efi
tmpfs               93612      12     93600   1% /run/user/1000
/dev/nvme1n1      8154588      28   7718748   1% /mnt/data
```

- Automatically mounted on reboot
    - Get UUID:

      blkid /dev/xvdf
    - Add entry to /etc/fstab:

      nano /etc/fstab

```
root@ip-192-168-25-13:/# blkid /dev/nvme1n1
/dev/nvme1n1: UUID="a6bd9c0f-34b1-47b1-b817-f4d4b08c74b6" BLOCK_SIZE="4096" TYPE="ext4"
root@ip-192-168-25-13:/# nano /etc/fstab
```

    - Add:

      UUID=fa33f940-2c9c-4efd-8806-9e9c05789a12 /mnt/data ext4 defaults,nofail 0 2
    - Save and close.

```
  GNU nano 7.2                                                                  /etc/fstab
UUID=fa33f940-2c9c-4efd-8806-9e9c05789a12  /mnt/data  ext4  defaults,nofail  0  2
LABEL=cloudimg-rootfs   /        ext4   discard,commit=30,errors=remount-ro     0 1
LABEL=BOOT      /boot   ext4    defaults        0 2
LABEL=UEFI      /boot/efi       vfat    umask=0077      0 1
```

Test Case 1: Disk Formatting and Mounting

Verification:

- df -h | grep /mnt/data

```
ubuntu@ip-192-168-25-13:~$ df -h | grep /mnt/data
/dev/nvme1n1      7.8G   24K  7.4G   1% /mnt/data
ubuntu@ip-192-168-25-13:~$
```

Test Case 2: Data Persistence

- Create a file inside the mounted directory:

  sudo bash -c 'echo "Test Data" > /mnt/data/testfile.txt'

```
root@ip-192-168-25-13:~# sudo bash -c 'echo "Test Data" > /mnt/data/testfile.txt'
root@ip-192-168-25-13:~# cat /mnt/data/testfile.txt
Test Data
root@ip-192-168-25-13:~#
```

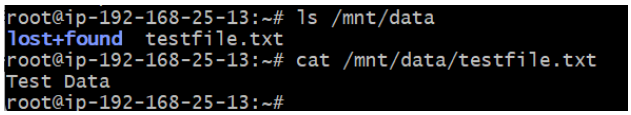
- Reboot instance:

  Reboot

After reconnecting:

ls /mnt/data

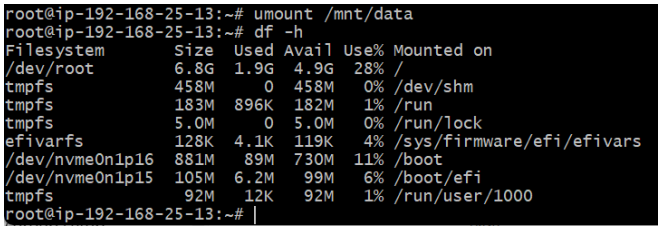
cat /mnt/data/testfile.txt



Test Case 3: Unmount and Remount

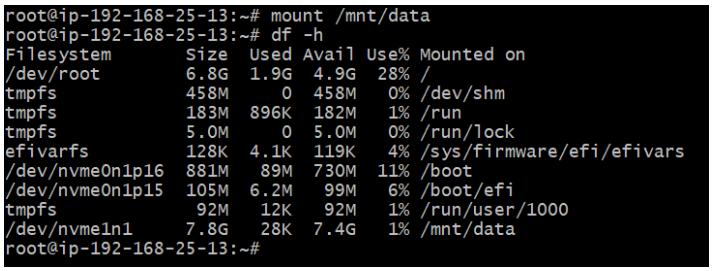
- Unmount:

  umount /mnt/data



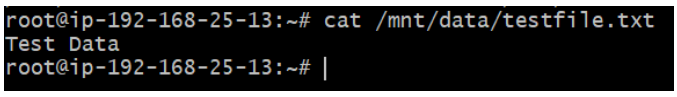
- Remount:

  mount /mnt/data



- Validate:

  cat /mnt/data/testfile.txt



Test Case 4: Volume Detachment

- Unmount safely:

  umount /mnt/data

```
root@ip-192-168-25-13:~# umount /mnt/data
root@ip-192-168-25-13:~# df -h
Filesystem       Size  Used Avail Use% Mounted on
/dev/root        6.8G  1.9G  4.9G  28% /
tmpfs            458M     0  458M   0% /dev/shm
tmpfs            183M  888K  182M   1% /run
tmpfs            5.0M     0  5.0M   0% /run/lock
efivarfs         128K  4.1K  119K   4% /sys/firmware/efi/efivars
/dev/nvme0n1p16  881M   89M  730M  11% /boot
/dev/nvme0n1p15  105M  6.2M   99M   6% /boot/efi
tmpfs             92M   12K   92M   1% /run/user/1000
root@ip-192-168-25-13:~#
```

- Detach using console

  EC2 → Volumes → Select Volume → Actions → **Detach Volume**

- Verify on instance:

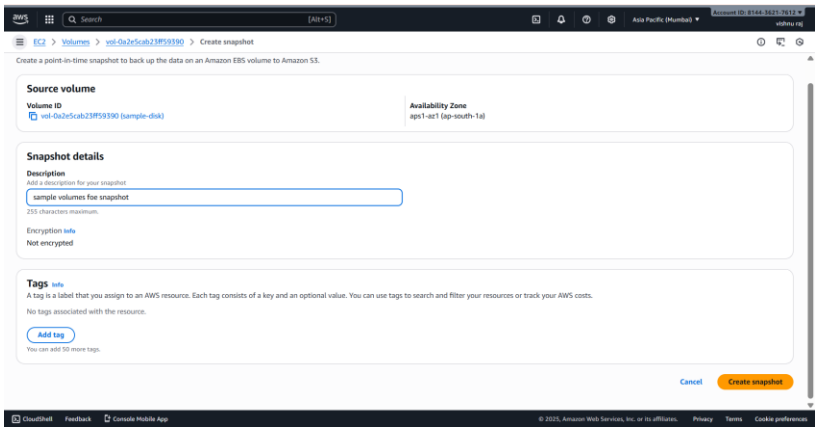
  lsblk

```
root@ip-192-168-25-13:~# lsblk
NAME          MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0           7:0    0 27.6M  1 loop /snap/amazon-ssm-agent/11797
loop1           7:1    0 73.9M  1 loop /snap/core22/2133
loop2           7:2    0 50.8M  1 loop /snap/snapd/25202
loop3           7:3    0   74M  1 loop /snap/core22/2163
nvme0n1       259:0    0    8G  0 disk
├─nvme0n1p1   259:2    0    7G  0 part /
├─nvme0n1p14  259:3    0    4M  0 part
├─nvme0n1p15  259:4    0  106M  0 part /boot/efi
└─nvme0n1p16  259:5    0  913M  0 part /boot
root@ip-192-168-25-13:~#
```

Test Case 5: Snapshot and Restore

- Snapshot Creation:

  Volumes → Select volume → **Actions → Create Snapshot**



- Create New Volume from Snapshot:

  Snapshots → Select snapshot → **Create Volume**

- Attach New Volume & Verify:

  sudo mount /dev/nvme1n1/mnt/data

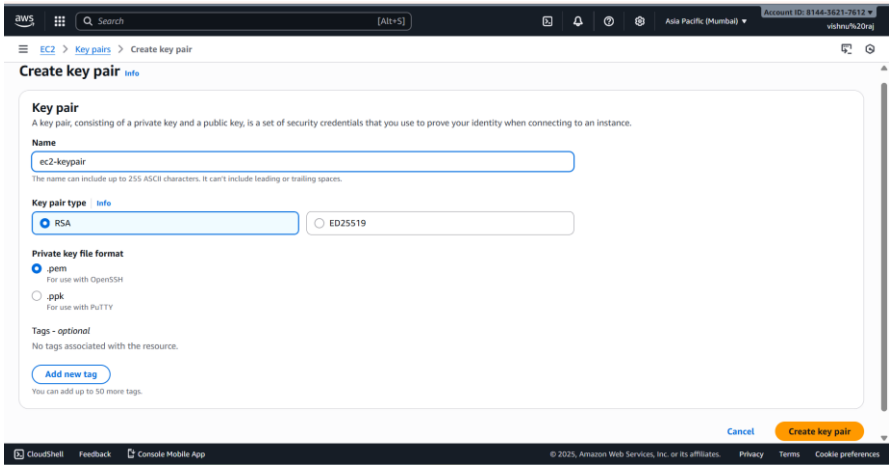
  ls /mnt/data

  cat /mnt/data/testfile.txt

```
root@ip-192-168-25-13:~# mount /dev/nvme1n1 /mnt/data
root@ip-192-168-25-13:~# df -h
Filesystem       Size  Used Avail Use% Mounted on
/dev/root        6.8G  1.9G  4.9G  28% /
tmpfs            458M     0  458M   0% /dev/shm
tmpfs            183M  908K  182M   1% /run
tmpfs            5.0M     0  5.0M   0% /run/lock
efivarfs         128K  4.1K  119K   4% /sys/firmware/efi/efivars
/dev/nvme0n1p16  881M   89M  730M  11% /boot
/dev/nvme0n1p15  105M  6.2M   99M   6% /boot/efi
tmpfs             92M   12K   92M   1% /run/user/1000
/dev/nvme1n1     7.8G   28K  7.4G   1% /mnt/data
root@ip-192-168-25-13:~# ls /mnt/data
lost+found  testfile.txt
root@ip-192-168-25-13:~# cat /mnt/data/testfile.txt
Test Data
root@ip-192-168-25-13:~#
```

EC2

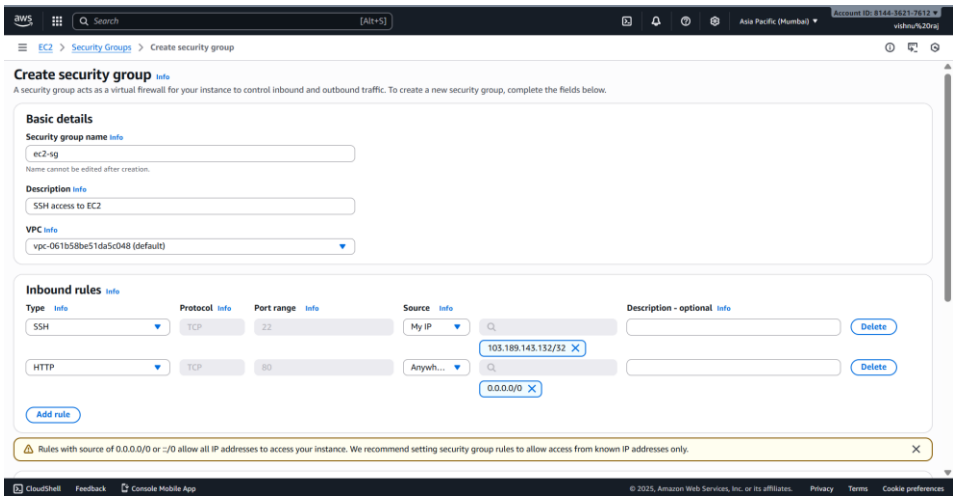
1. **Key Pair**
    - Go to EC2 Console → Network & Security → Key Pairs
    - Click **Create Key Pair**
    - Provide a name for **Key Pair**
    - Select:
        - **Key pair type:** RSA
        - **File format:** .pem (for Linux/Mac) or .ppk (for PuTTY)
    - Click **Create key pair** (The key is automatically downloaded)



2. **Security Group**
    - Go to EC2 Console → Network & Security → Security Groups → Create security group
    - Provide a name for **Security Groups**
    - VPC: Select your custom VPC
    - Add **Inbound Rules**:

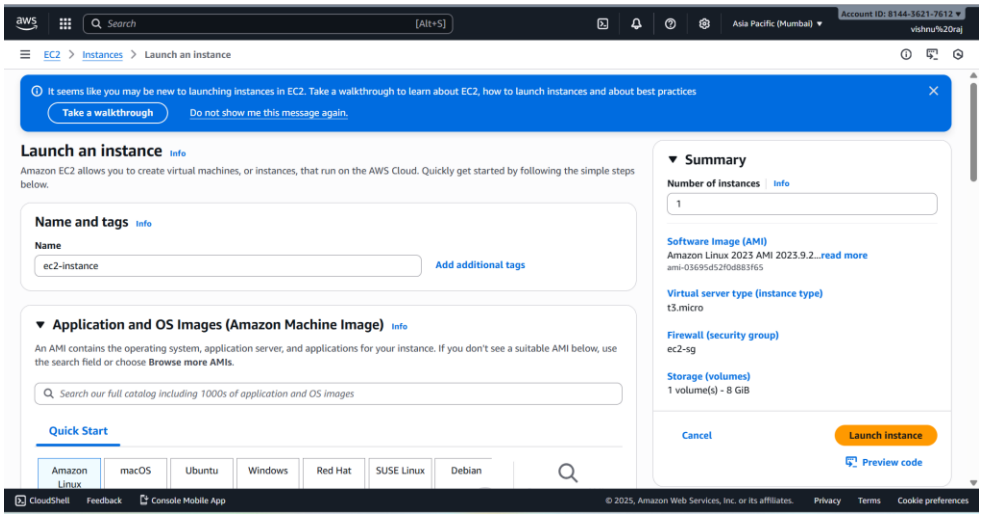
Type	Port	Source
SSH	22	Your IP (e.g., 123.45.67.89/32)
HTTP	80	0.0.0.0/0

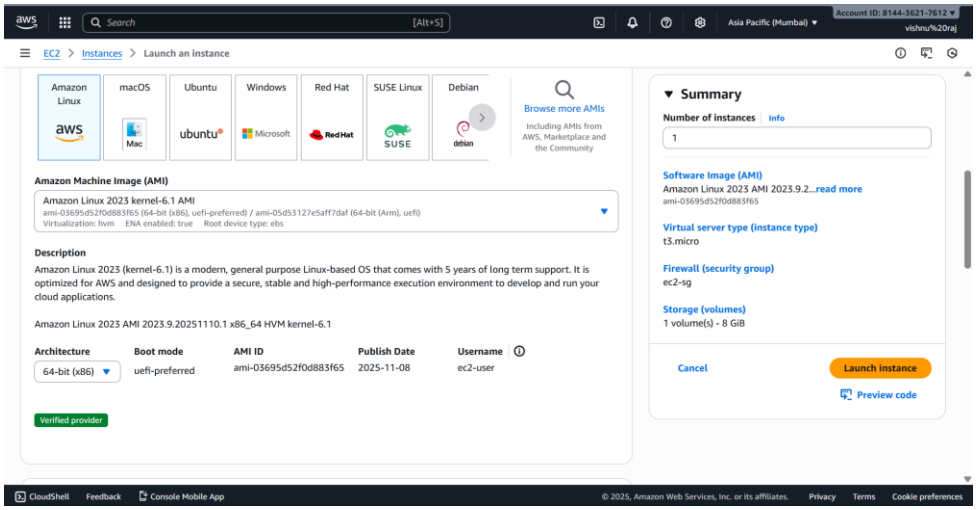
- Outbound Rules: **Allow all traffic**
- Click **Create security group**

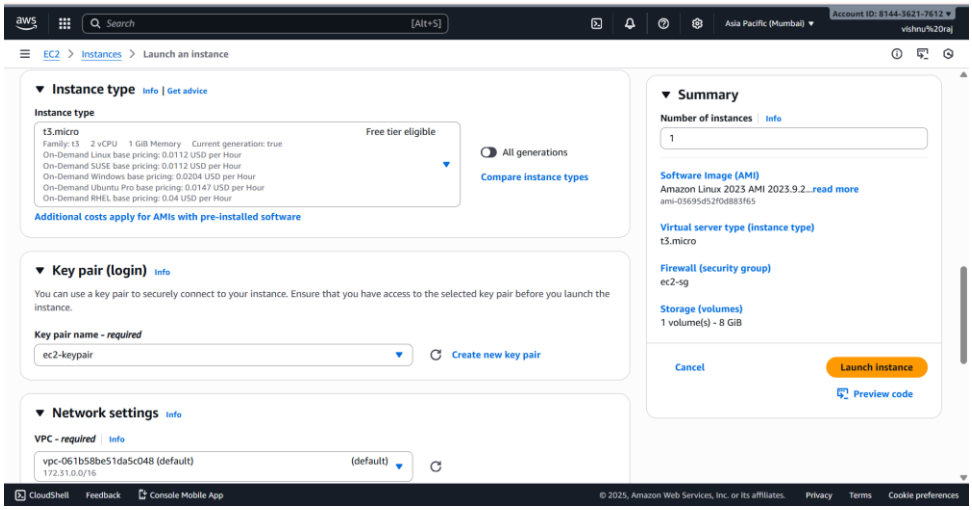


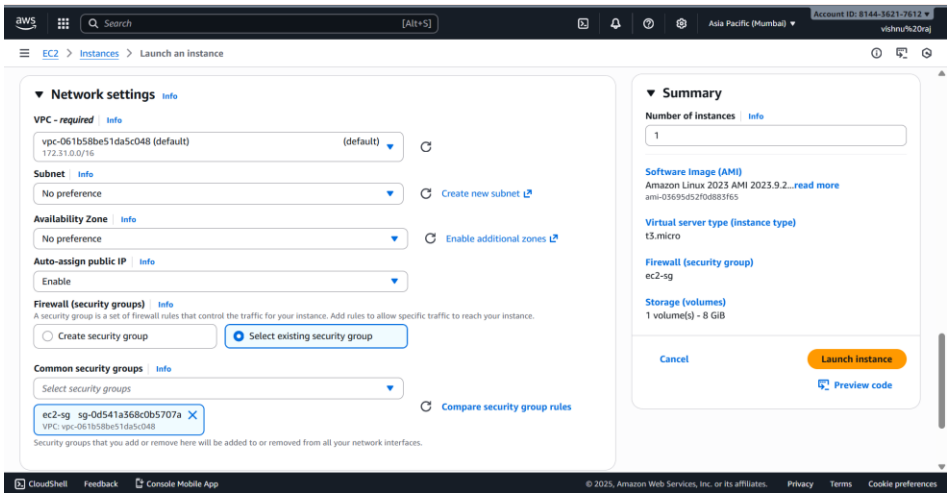
3. EC2 Instance

- EC2 → Instances → **Launch Instance**
- Provide a name for **EC2 Launch**
- AMI: **Amazon Linux 2**
- Instance type: **t3.micro**
- Provide a name for **Key Pair**
- Network Settings:
  - Add VPC
  - Auto-assign Public IP: **Enable**
  - Security Group: **Choose existing → ec2-sg**
- Click Launch


Search [Alt+S]
Asia Pacific (Mumbai)
Account ID: 8144-3621-7612
vishnu%20raj
EC2 > Instances > Launch an instance
It seems like you may be new to launching instances in EC2. Take a walkthrough to learn about EC2, how to launch instances and about best practices
Take a walkthrough
Do not show me this message again.
Launch an instance Info
Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.
Name and tags Info
Name
ec2-instance
Add additional tags
Application and OS Images (Amazon Machine Image) Info
An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose Browse more AMIs.
Search our full catalog including 1000s of application and OS images
Quick Start
Amazon Linux
macOS
Ubuntu
Windows
Red Hat
SUSE Linux
Debian
Summary
Number of instances Info
1
Software Image (AMI)
Amazon Linux 2023 AMI 2023.9.2...read more
ami-03695d52f0d883f65
Virtual server type (instance type)
t3.micro
Firewall (security group)
ec2-sg
Storage (volumes)
1 volume(s) - 8 GiB
Cancel
Launch instance
Preview code
CloudShell
Feedback
Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Search [Alt+S]
Asia Pacific (Mumbai)
Account ID: 8144-3621-7612
vishnu%20raj
EC2 > Instances > Launch an instance
Amazon Linux
macOS
Ubuntu
Windows
Red Hat
SUSE Linux
Debian
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community
Amazon Machine Image (AMI)
Amazon Linux 2023 kernel-6.1 AMI
ami-03695d52f0d883f65 (64-bit (x86), uefi-preferred) / ami-05d53127e5aff7daf (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs
Description
Amazon Linux 2023 (kernel-6.1) is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.
Amazon Linux 2023 AMI 2023.9.20251110.1 x86_64 HVM kernel-6.1
Architecture
64-bit (x86)
Boot mode
uefi-preferred
AMI ID
ami-03695d52f0d883f65
Publish Date
2025-11-08
Username
ec2-user
Verified provider
Summary
Number of instances Info
1
Software Image (AMI)
Amazon Linux 2023 AMI 2023.9.2...read more
ami-03695d52f0d883f65
Virtual server type (instance type)
t3.micro
Firewall (security group)
ec2-sg
Storage (volumes)
1 volume(s) - 8 GiB
Cancel
Launch instance
Preview code
CloudShell
Feedback
Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Search [Alt+S]
Asia Pacific (Mumbai)
Account ID: 8144-3621-7612
vishnu%20raj
EC2 > Instances > Launch an instance
Instance type Info | Get advice
Instance type
t3.micro
Free tier eligible
Family: t3 2 vCPU 1 GiB Memory Current generation: true
On-Demand Linux base pricing: 0.0112 USD per Hour
On-Demand SUSE base pricing: 0.0112 USD per Hour
On-Demand Windows base pricing: 0.0204 USD per Hour
On-Demand Ubuntu Pro base pricing: 0.0147 USD per Hour
On-Demand RHEL base pricing: 0.04 USD per Hour
All generations
Compare instance types
Additional costs apply for AMIs with pre-installed software
Key pair (login) Info
You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.
Key pair name - required
ec2-keypair
Create new key pair
Network settings Info
VPC - required Info
vpc-061b58be51da5c048 (default)
172.31.0.0/16
(default)
Summary
Number of instances Info
1
Software Image (AMI)
Amazon Linux 2023 AMI 2023.9.2...read more
ami-03695d52f0d883f65
Virtual server type (instance type)
t3.micro
Firewall (security group)
ec2-sg
Storage (volumes)
1 volume(s) - 8 GiB
Cancel
Launch instance
Preview code
CloudShell
Feedback
Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences



4. Networking

- EC2 → Instances → **Launch Instance** → **connect**
- Select SSH client → steps to connect using SSH is given
- Open **Git Bash** → Go to the **Key Pair** downloaded path
- Connect to your instance using its Public DNS

  Eg: -

  ssh -i "ec2-keypair.pem" ec2-user@<public-ip>

- To verify internet connectivity use “ping google.com”

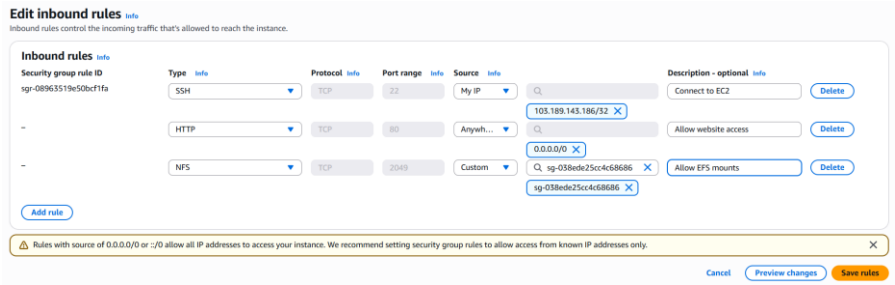
```
Vishnu Raj@LAPTOP-KSAOBH37 MINGW64 ~/Downloads
$ ssh -i "ec2-keypair.pem" ec2-user@65.0.27.55
The authenticity of host '65.0.27.55 (65.0.27.55)' can't be established.
ED25519 key fingerprint is SHA256:gfDmXntos43SmrdN90kwxQhZZ2tOQNOoYsgHzH3gQZs.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '65.0.27.55' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-3-0 ~]$ ping google.com
PING google.com (142.250.67.206) 56(84) bytes of data.
64 bytes from bom12s08-in-f14.1e100.net (142.250.67.206): icmp_seq=1 ttl=114 tim
e=2.18 ms
64 bytes from bom12s08-in-f14.1e100.net (142.250.67.206): icmp_seq=2 ttl=114 tim
e=2.19 ms
64 bytes from bom12s08-in-f14.1e100.net (142.250.67.206): icmp_seq=3 ttl=114 tim
e=2.18 ms
64 bytes from bom12s08-in-f14.1e100.net (142.250.67.206): icmp_seq=4 ttl=114 tim
e=2.19 ms
64 bytes from bom12s08-in-f14.1e100.net (142.250.67.206): icmp_seq=5 ttl=114 tim
e=2.18 ms
64 bytes from bom12s08-in-f14.1e100.net (142.250.67.206): icmp_seq=6 ttl=114 tim
e=2.19 ms
64 bytes from bom12s08-in-f14.1e100.net (142.250.67.206): icmp_seq=7 ttl=114 tim
e=2.18 ms
^C
--- google.com ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6009ms
rtt min/avg/max/mdev = 2.181/2.184/2.191/0.003 ms
[ec2-user@ip-172-31-3-0 ~]$
```

EFS

1. **VPC and Networking Setup**
    - VPC → Your VPCs → Create VPC
    - Select **2 Public Subnets** in different Availability Zones
    - Ensure proper routing and an Internet Gateway for internet access
2. **Security Group Configuration**
    - Go to EC2 Console → Network & Security → Security Groups → Create security group
    - Provide a name for **Security Groups**
    - VPC: Select your custom VPC
    - Add **Inbound Rules**:

Type	Port	Source	Purpose
SSH	22	Your IP	Connect to EC2
HTTP	80	0.0.0.0/0	Allow website access
NFS	2049	SG-Web	Allow EFS mounts

    - Outbound Rules: **Allow all traffic**
    - Click **Create security group**

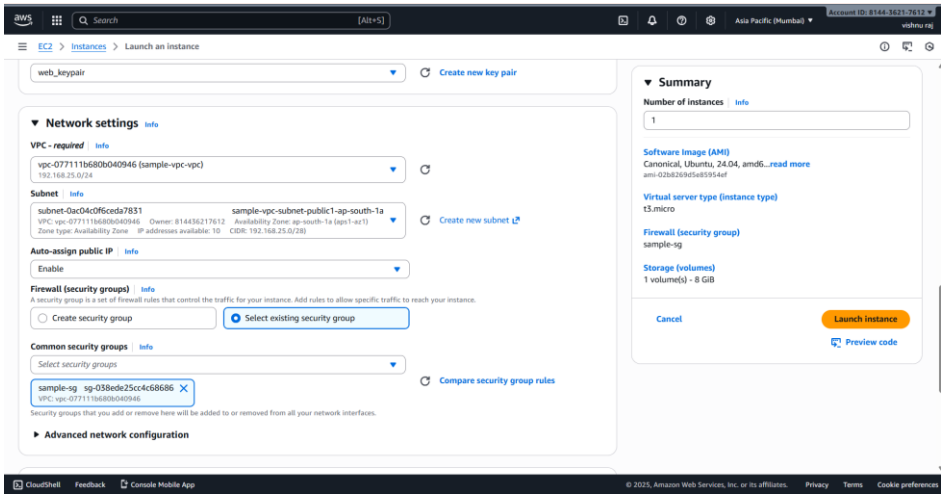


3. **Create EC2 Instances**

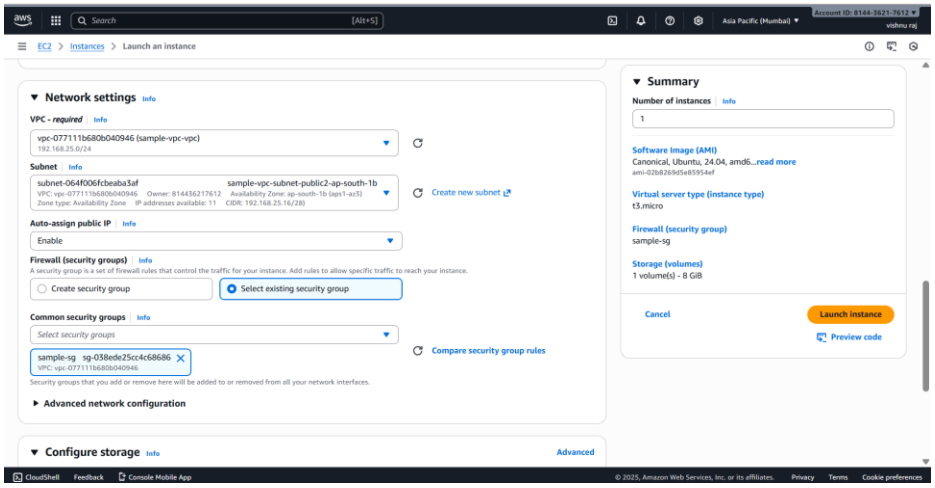
    Launch two EC2 instances:
    - Amazon Linux 2
    - Subnet-A → EC2-1
    - Subnet-B → EC2-2

- Assign Public IP (Enable Auto-assign Public IP)
- Attach SG-Web

EC2-1 Network settings

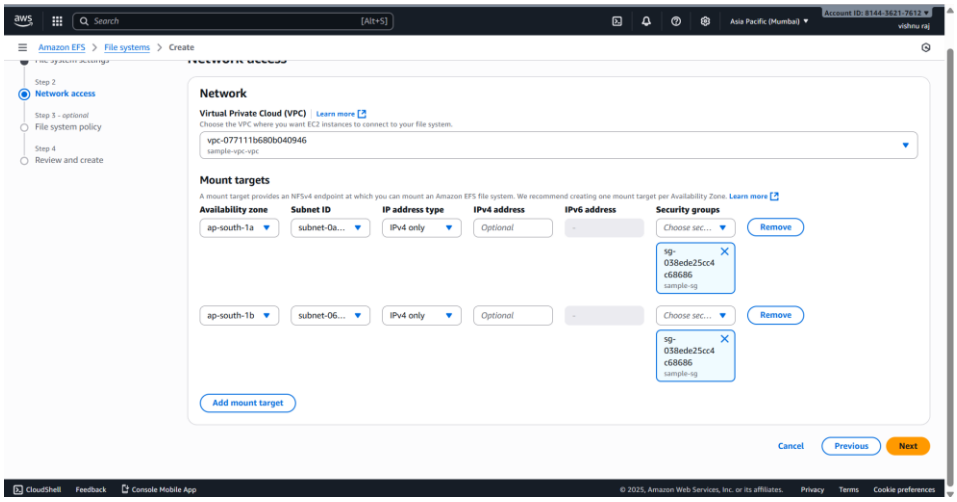


EC2-2 Network settings



4. Create and Configure EFS

- Go to EFS Console
- Create File System
- Choose your VPC
- Select Customize for more option
- Enable mount targets in **both subnets**
- Attach SG-Web (same SG as EC2)
- Finally click create



- Mount EFS on Both EC2 Instances
  - Connect to EC2 instance

    ssh -i "web_keypair.pem" ubuntu@PUBLIC-IP
  - Install EFS utils on both EC2 instance

    sudo apt-get update
    sudo apt-get -y install git binutils rustc cargo pkg-config libssl-dev gettext
    git clone https://github.com/aws/efs-utils
    cd efs-utils
    ./build-deb.sh
    sudo apt-get -y install ./build/amazon-efs-utils*deb
  - Mount EFS

    sudo mount -t efs fs-00cc75f7f48769ba8:/ /var/www/html
  - Make mount permanent

    sudo vi /etc/fstab

    Add this line:

    Fs-00cc75f7f48769ba8:/ /var/www/html efs defaults,_netdev 0 0

    Save & reboot
  - Create index.html on both EC2 instance

    echo "Hello World from EFS" | sudo tee /var/www/html/index.html
    cat /var/www/html/index.html

EC2-1 instance

```
ubuntu@ip-192-168-25-10:~$ echo "Hello World from EFS" | sudo tee /var/www/html/index.html
Hello World from EFS
ubuntu@ip-192-168-25-10:~$ cat /var/www/html/index.html
Hello World from EFS
ubuntu@ip-192-168-25-10:~$
```

EC2-2 instance

```
ubuntu@ip-192-168-25-20:~$ sudo mkdir -p /var/www/html
ubuntu@ip-192-168-25-20:~$ echo "Hello World from EFS" | sudo tee /var/www/html/
index.html
Hello World from EFS
ubuntu@ip-192-168-25-20:~$ cat /var/www/html/index.html
Hello World from EFS
ubuntu@ip-192-168-25-20:~$
```

5. **Test Website**
    - Open in browser:
        - http://PUBLIC-IP-OF-EC2-1
        - http://PUBLIC-IP-OF-EC2-2
    - Both should show the SAME content:

Hello World from EFS

6. **High Availability Test**
    - Stop / terminate EC2-1.
    - Open website in browser using **EC2-2 Public IP**:
        - http://PUBLIC-IP-OF-EC2-2

ELB

Set up a scalable and highly available website using **Elastic Load Balancer (ELB)** and **Apache Web Servers** hosted on EC2 instances

1. Create two EC2 instance with same VPC and Availability Zone
    - EC2 → Instances → **Launch Instance**
    - Provide a name for **EC2 Launch**
    - AMI: **Ubuntu**
    - Instance type: **t3.micro**
    - Provide a name for **Key Pair**
    - Network Settings:
        - Add VPC
        - Auto-assign Public IP: **Enable**
        - Security Group: **Choose existing → web-sg**
    - Click Launch
2. Install Apache Web Server

    sudo apt update -y

    sudo apt install -y apache2

    sudo systemctl enable apache2

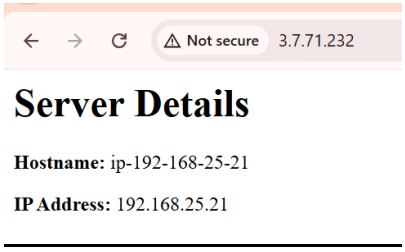
    sudo systemctl start apache2

    sudo systemctl status apache2
3. Create a simple HTML page
    - First open the html page
        - cd /var/www/html
    - Remove the html page and add new html page
        - rm -f index.html
        - vim index.html
    - Add below command to display the server's **hostname** and **IP address**

echo "<h1>Server Details</h1> Hostname: $(hostname) IP Address: $(hostname -I | cut -d\" \" f1) " | sudo tee /var/www/html/index.html

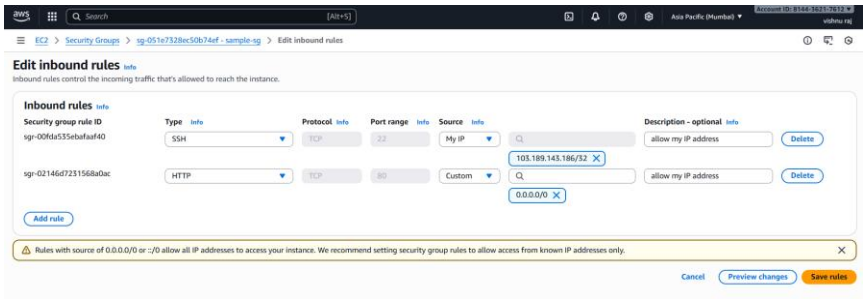
Output for first EC2 machine

Output for second EC2 machine

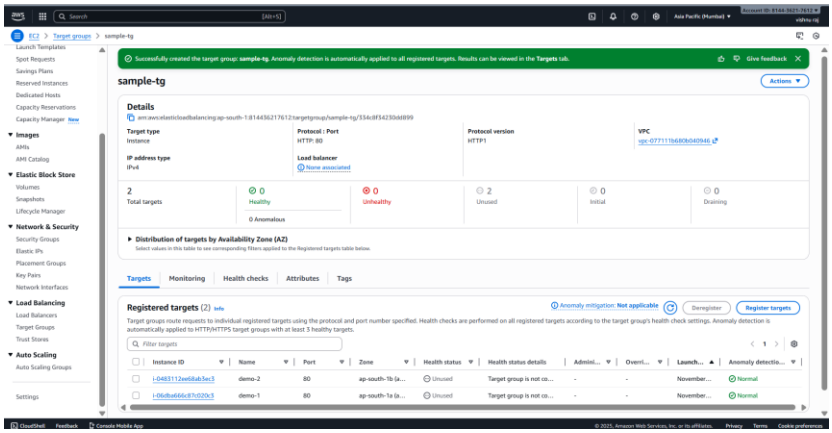


---

4. Security Group for EC2:
    - Allow inbound TCP 22 from your IP
    - Allow inbound TCP 80 from 0.0.0.0/0

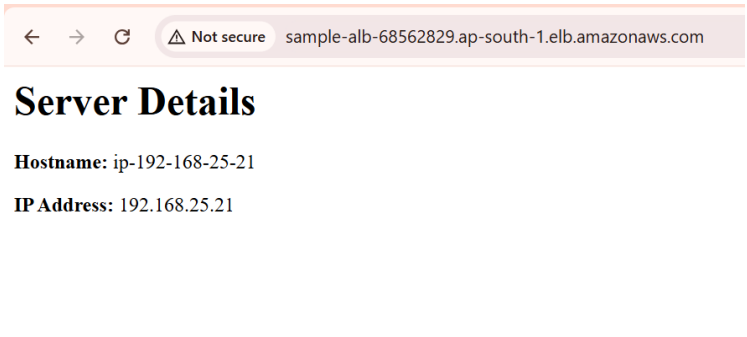


5. Create Target Group:
    - EC2 → Target groups → **Create target group**
    - Provide **Target group name**
    - Provide Protocol as HTTP and Port as 80
    - Give VPC
    - Health check path: /
    - Register both EC2 instances as targets (port 80)
    - Create target group



6. Create an Application Load Balancer:
    - EC2 → Load balancers → **Create Application Load balancer**
    - Provide Load balancer name
    - Scheme: Internet-facing
    - Provide VPC and Security groups
    - Create Listener rule: Forward HTTP 80 to the Target Group
    - Create Load balancer
7. Verify Load Balancer
    - Copy ALB DNS name is shown in the console after creation. Open in a browser and refresh repeatedly and you should see the Hostname/IP alternate between the two instances

Output



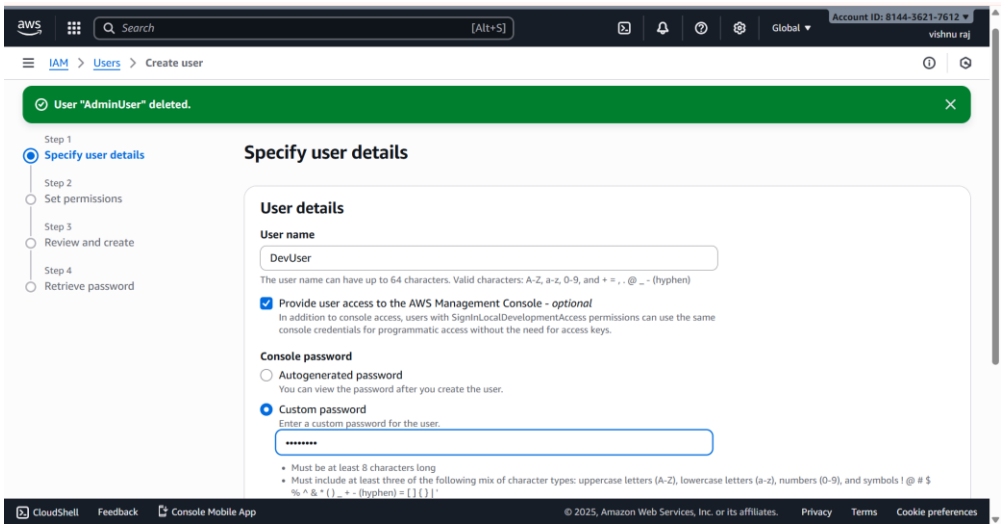
IAM

1. **Create IAM Users & Groups**

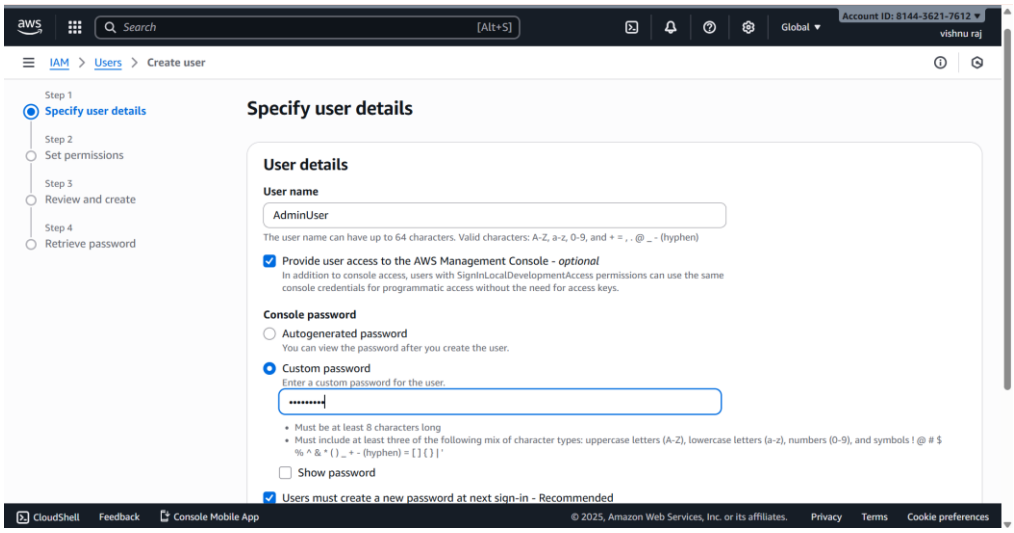
   Create IAM Users:

   - Go to **IAM Console → Users → Create User**
   - Create:
     - DevUser
     - AdminUser
   - Choose **Password** or **Access Key** as required by assignment.

DevUser:



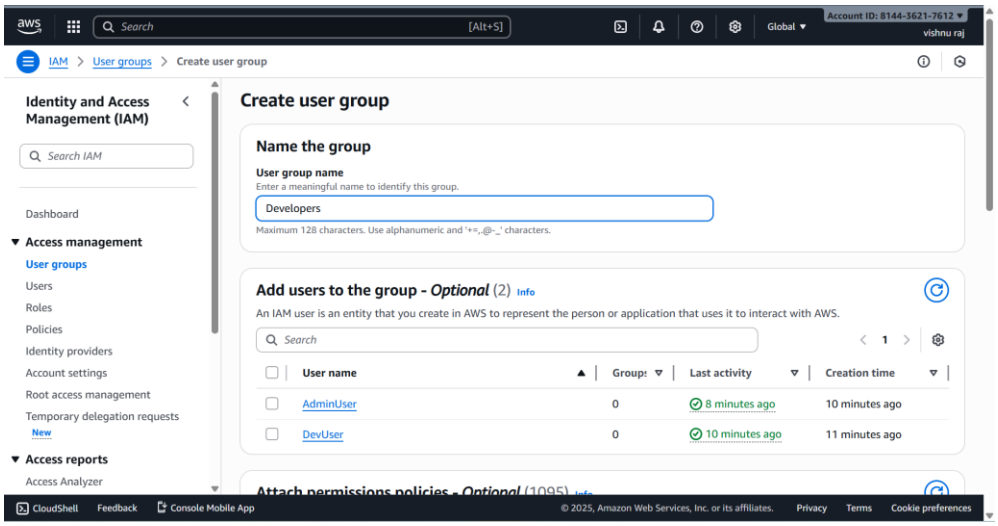
AdminUser:



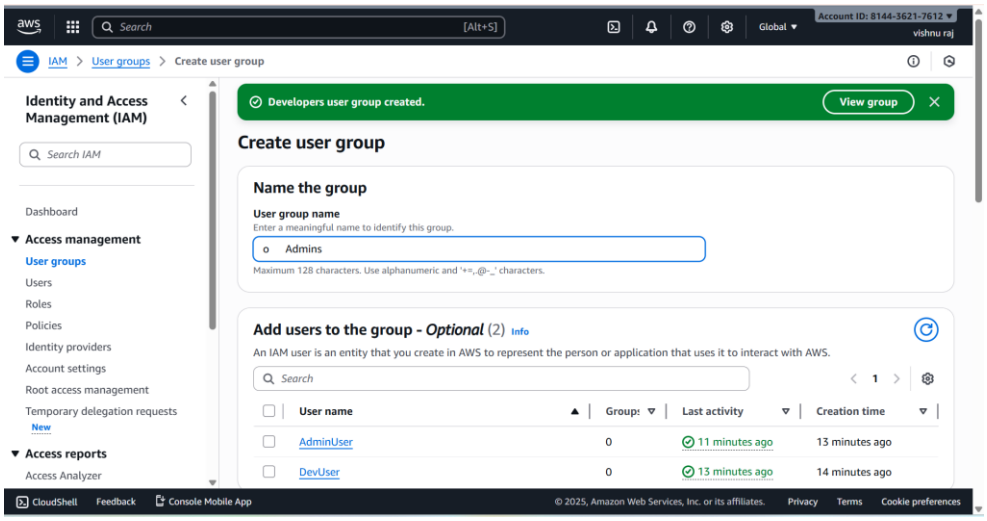
Create IAM Groups:

- Go to **IAM → User Groups → Create Group**
- Create:
  - Developers
  - Admins

Developers:



Admins:

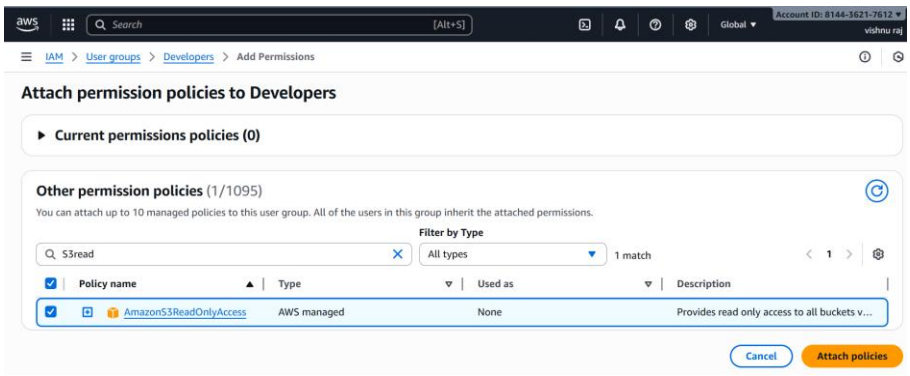


Assign Users to Groups:

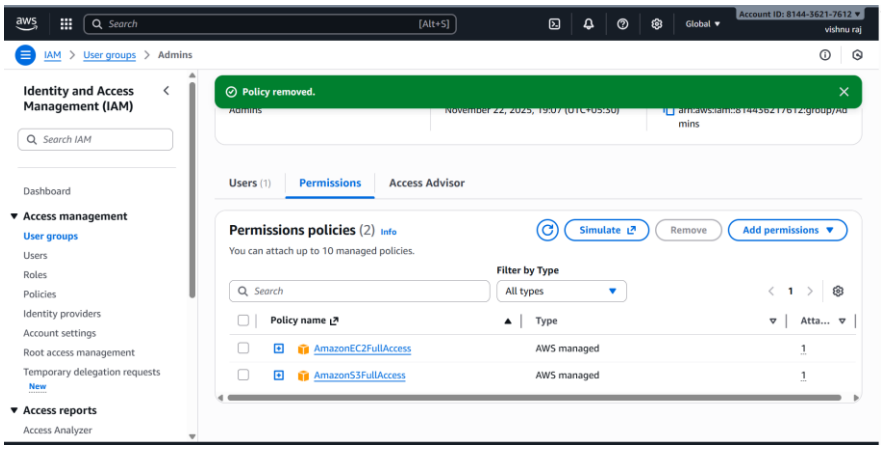
- Go to each group → **Add Users**
  - Add **DevUser → Developers**
  - Add **AdminUser → Admins**

Attach Policies to Groups

- For Developers Group
    - IAM → User Groups → Developers → Add Permissions → Attach Policies → AmazonS3ReadOnlyAccess



- For Admins Group
    - IAM → User Groups → Admins → Add Permissions → Attach Policies → AmazonEC2FullAccess and AmazonS3FullAccess



2. **Create IAM Role for AWS Service Access**

- Create an IAM Role
    - IAM → **Roles → Create Role**
    - Trusted entity type: **AWS Service**
    - Choose: **EC2**
    - Continue and attach: AmazonS3FullAccess
    - Name the role: EC2-S3-Access-Role

3. **Set Up Cross-Account IAM Role**
    - Create the Cross-Account Role in Account A
        - Go to IAM → **Roles → Create Role**
        - Trusted entity type: **Another AWS Account**
        - Enter **Account B's Account ID**
        - Continue
    - Use sts:AssumeRole to allow access to **Account A's S3 bucket**
        - IAM → Roles → CrossAccountS3ReadRole → Permissions → Add Inline Policy

```
{
  "Version": "2012-10-17",
  "Statement": [
   {
    "Effect": "Allow",
    "Principal": {
     "AWS": "arn:aws:iam::<ACCOUNT_B_ID>:root"
    },
    "Action": "sts:AssumeRole"
   }
  ]
}
```

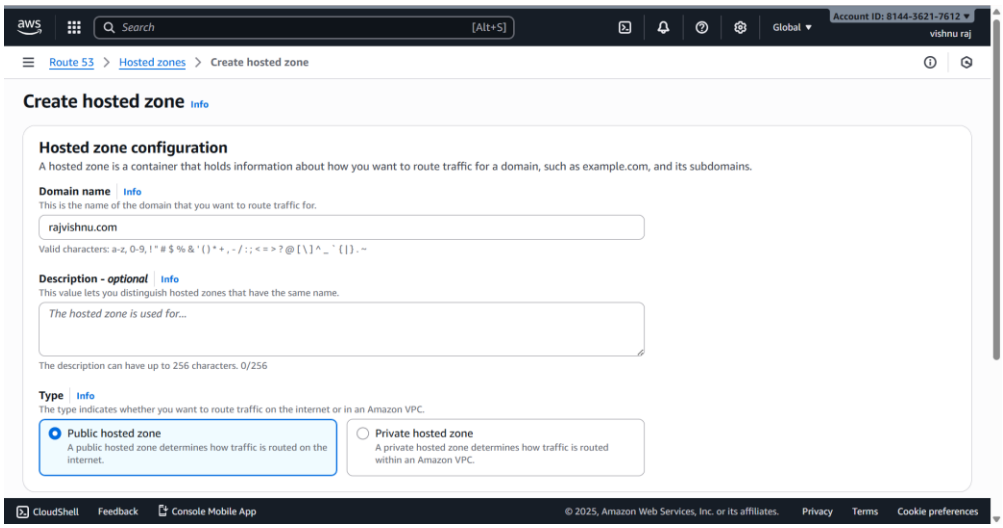
    - Read access to a specific **S3 bucket** in Account A
        - IAM → Roles → CrossAccountS3ReadRole → Permissions → Add Inline Policy

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "s3:ListBucket"
   ],
   "Resource": "arn:aws:s3:::your-bucket-name"
  },
```

```
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": "arn:aws:s3:::your-bucket-name/*"
    }
  ]
}
```

Route53

1. **Register a domain**
    - Register a domain through an external registrar such as
        - GoDaddy
        - Namecheap
        - Google Domain
    - If you already own a domain, you may use it.
2. **Create a Public Hosted Zone**
    - Go to AWS Console → Route 53 → Hosted zones
    - Click **Create hosted zone**.
    - Enter your root domain name
    - Choose **Public hosted zone**
    - Click **Create hosted zone**



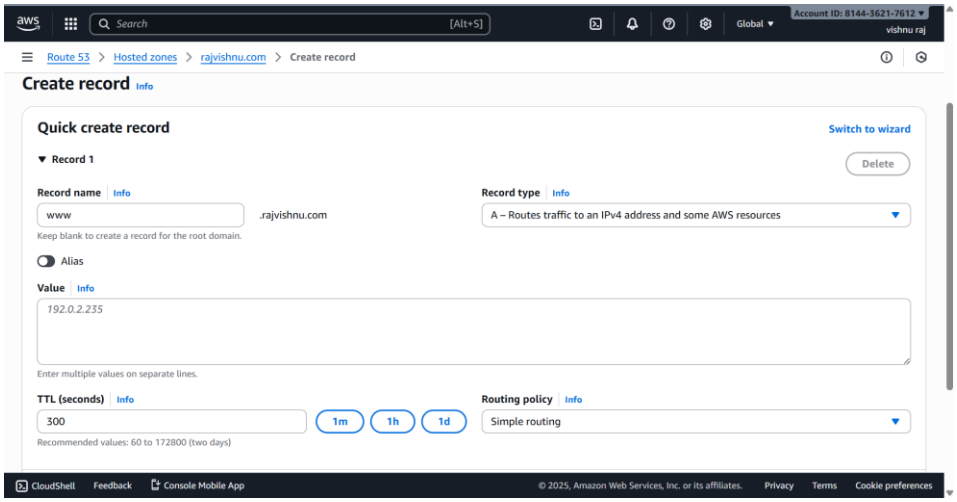
3. **Update Nameserver Settings at the Registrar**

   At Namecheap

   - Go to **Domain List → Manage → Nameservers**
   - Select **Custom DNS**
   - Paste the **Route 53 nameservers**
   - Save

4. **Create DNS records**

   - Create an **A Record → EC2 Public IP**
   - Select **Create record**
   - Name: blank (root) or “www”
   - Type: **A – IPv4 address**
   - Value: **EC2 public IP address**
   - TTL: leave default (300 seconds)
   - Save



5. **Verify the configuration**

   Running nslookup or dig to confirm the domain resolves correctly

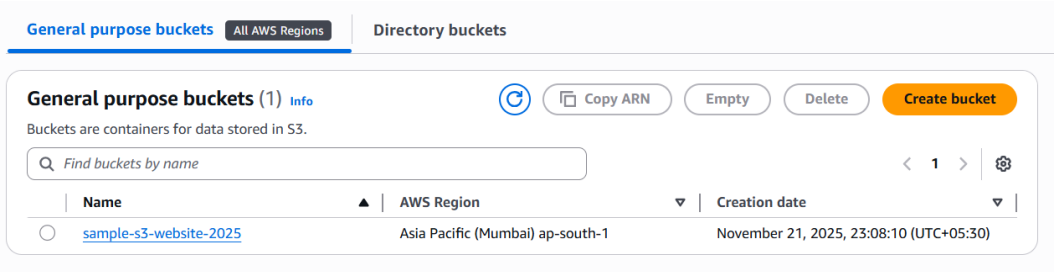
   - Using nslookup
     - nslookup mydomain.com
   - Using dig
     - dig mydomain.com +trace

Browser Test

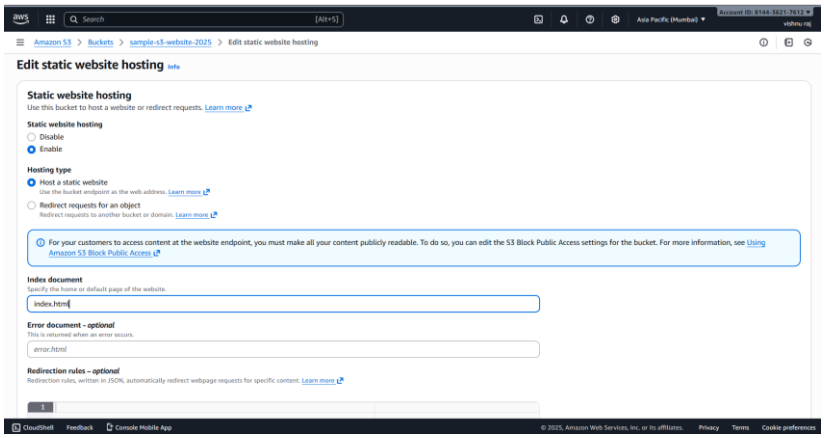
- Enter your domain in a web browser:
  - https://mydomain.com

S3

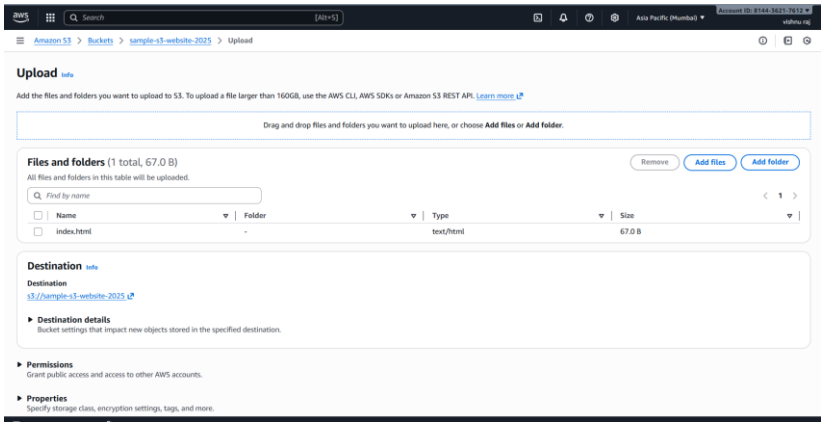
1. **Create and Configure S3 Bucket**
   - **Create S3 Bucket**
     - Go to **Amazon S3 → Create Bucket**
     - Enter a globally unique bucket name
     - Choose a region
     - Uncheck “Block all public access”
     - Acknowledge the warning and create the bucket.



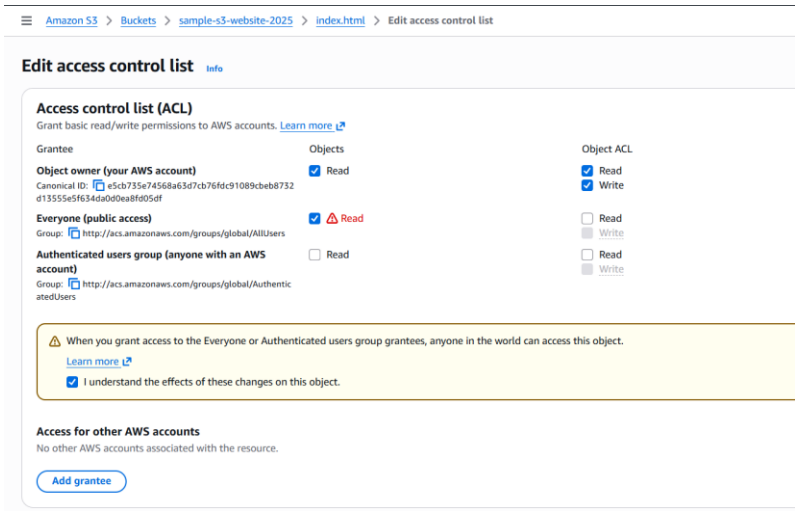
   - **Enable Static Website Hosting**
     - Open your bucket → Properties tab
     - Scroll to **Static Website Hosting**
     - Choose:
       1. Enable
       2. Hosting type*: Host a static website*
       3. Index document: index.html
     - Save changes



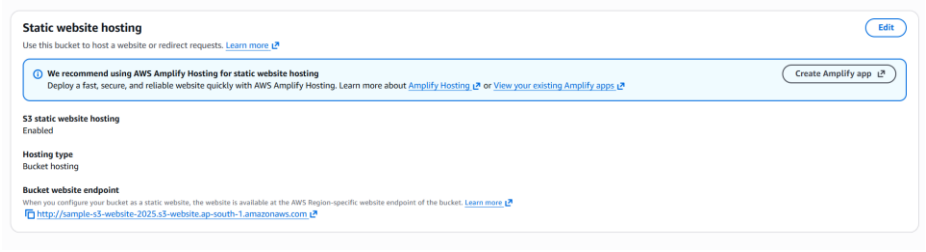
- **Upload Index Content**
  - Go to the Objects tab → Upload
  - Upload your file index.html

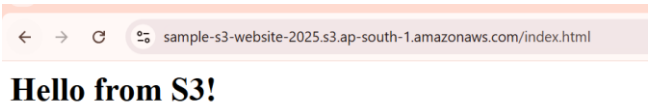


- **Configure the bucket permissions**
  - Ensure index.html is public (S3 → Object → Permissions → Public Access)



- **Access the Website**
    - Go to Properties → Static website hosting and open the Website endpoint:



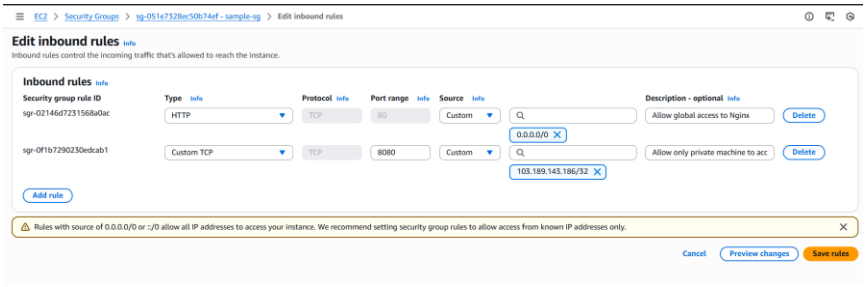


2. High-Level Storage Comparison

Feature	Amazon S3	Amazon EFS	Amazon EBS
Type	Object storage	File storage	Block storage
Use Case	Static websites, backups, data lakes	Shared file system for multiple EC2	File system for a single EC2
Access	HTTP(S), API-based	NFS protocol	Attached as a disk
Performance	High throughput, not low-latency	Good throughput & parallel access	Lowest latency, high IOPS
Scalability	Virtually unlimited	Automatically scales	Fixed size volume
Cost Model	Pay per GB stored + requests	Pay per GB + throughput	Pay provisioned size + IOPS
Availability & Durability	99.999999999% (11 9's) durability	Multi-AZ by default	AZ-specific unless snapshot
Best For	Static content, backups, large data	Shared storage for multiple EC2 in same VPC	Databases, OS disks, low-latency apps

SG

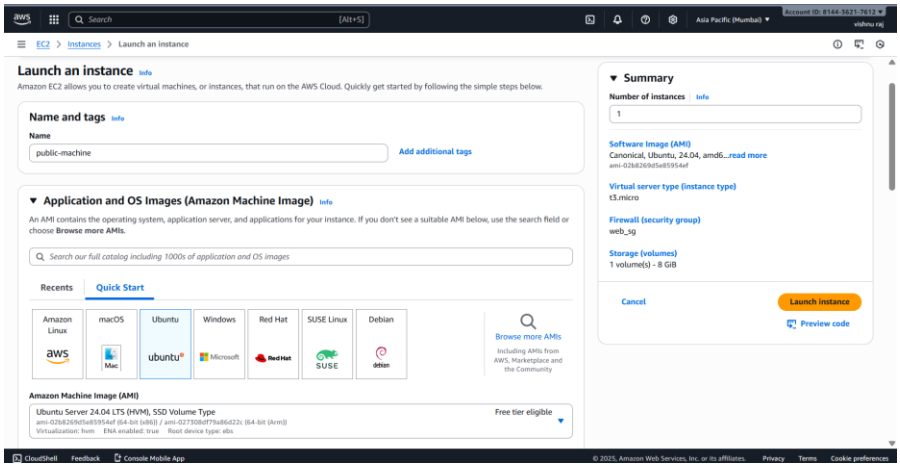
1. **Security Group**
    - Go to EC2 Console → Network & Security → Security Groups → Create security group
    - Provide a name for **Security Groups**
    - VPC: Select your custom VPC
    - Add **Inbound Rules**:
        - Inbound HTTP traffic on **port 80** from anywhere (0.0.0.0/0)
        - Inbound HTTP traffic on **port 8080** from another EC2 instance's private IP.
        - Outbound Rules: **Allow all traffic**
    - Click **Create security group**

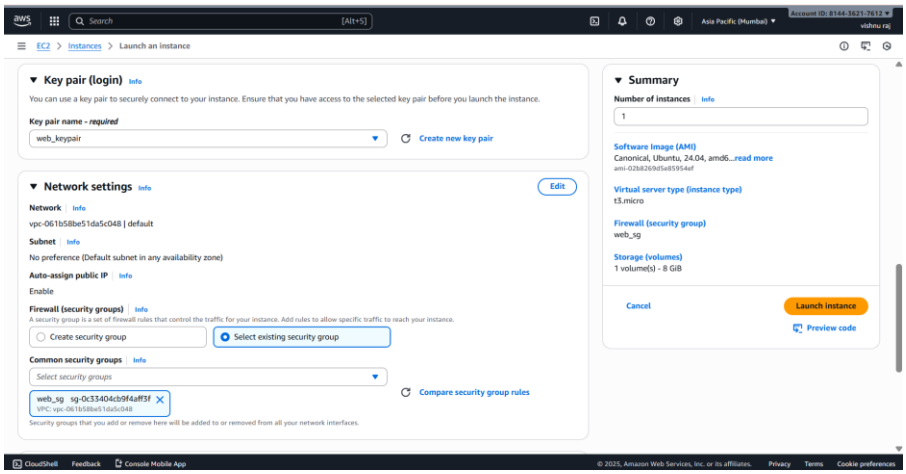


2. **EC2 Instance**

    For public EC2 instance

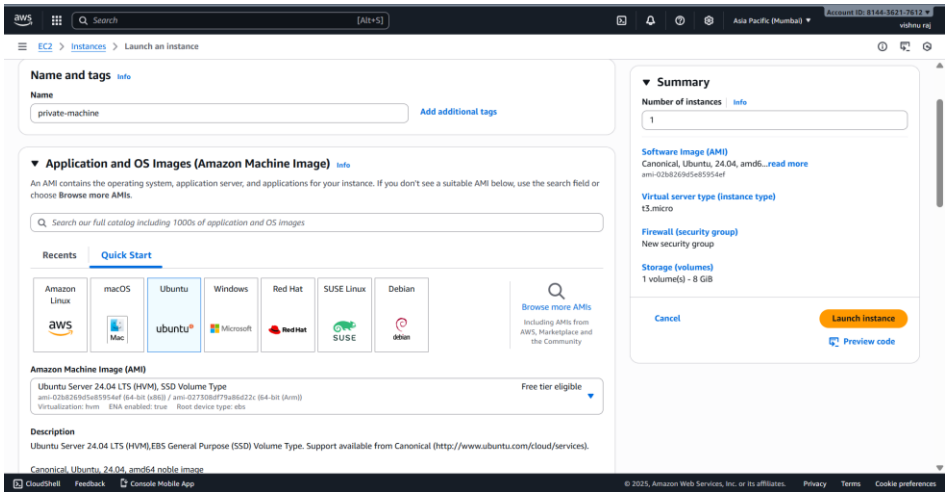
    - EC2 → Instances → **Launch Instance**
    - Provide a name for **EC2 Launch**
    - AMI: **Ubuntu**
    - Instance type: **t3.small**
    - Provide a name for **Key Pair**
    - Network Settings:
        - Add VPC
        - Auto-assign Public IP: **Enable**
        - Security Group: **Choose existing → web-sg**
    - Click Launch

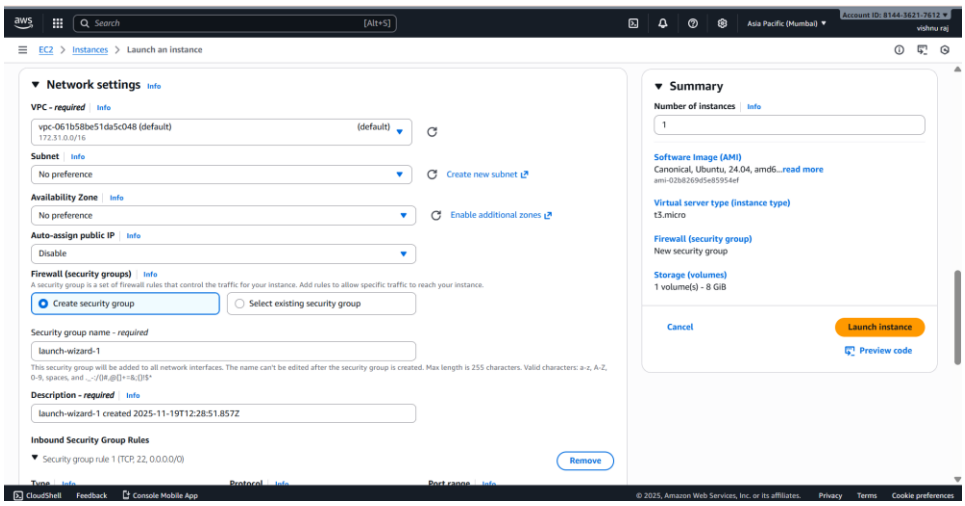




For private EC2 instance

- EC2 → Instances → **Launch Instance**
- Provide a name for **EC2 Launch**
- AMI: **Ubuntu**
- Instance type: **t3.small**
- Provide a name for **Key Pair**
- Network Settings:
    - Add VPC
    - Auto-assign Public IP: **Disable**
- Click Launch





3. Web Servers Installation

- SSH into **public-machine**
  - ssh -i yourkey.pem ubuntu@<public-ip>
- Install and Configure Nginx (Port 80)

```
sudo apt update

sudo apt install -y nginx

sudo systemctl enable nginx

sudo systemctl start nginx
```

Test locally:

```
curl http://localhost
```

- Install Tomcat (Port 8080)

```
sudo apt install -y default-jdk
cd /opt
sudo wget https://downloads.apache.org/tomcat/tomcat-10/v10.1.26/bin/apache-tomcat-10.1.26.tar.gz
sudo tar -xvzf apache-tomcat-10.1.26.tar.gz
sudo mv apache-tomcat-10.1.26 tomcat
sudo chmod +x /opt/tomcat/bin/*.sh
sudo /opt/tomcat/bin/startup.sh
```

Check locally:

```
curl http://localhost:8080
```

4. **Testing**

- Test Nginx from Browser
  - Open browser →

    http://<public-ip>
  - You should see:

- Test Tomcat from Private Machine
  - SSH into **private-machine from Public-machine:**

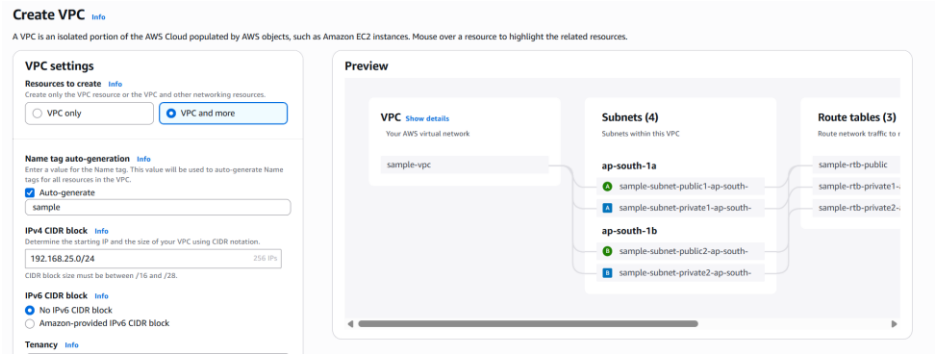
    ssh -i yourkey.pem ubuntu@<private-machine-private-ip>
  - Now curl the public machine's private IP on 8080:

    curl http://<public-machine-private-ip>:8080

VPC

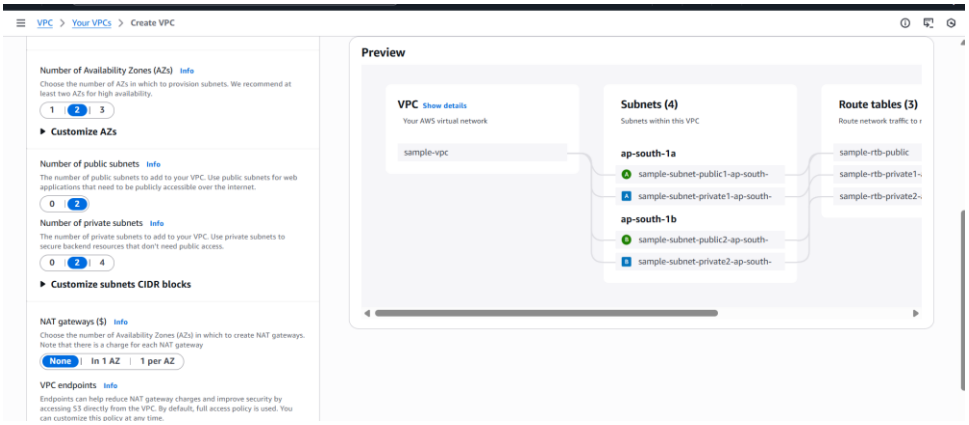
1. VPC Configuration

- VPC → Your VPCs → Create VPC
- Select VPC and more from **Resources to create**
- Provide VPCs Name
- Give **192.168.25.0/24** on IPv4 CIDR block
- Click on **Create VPC**



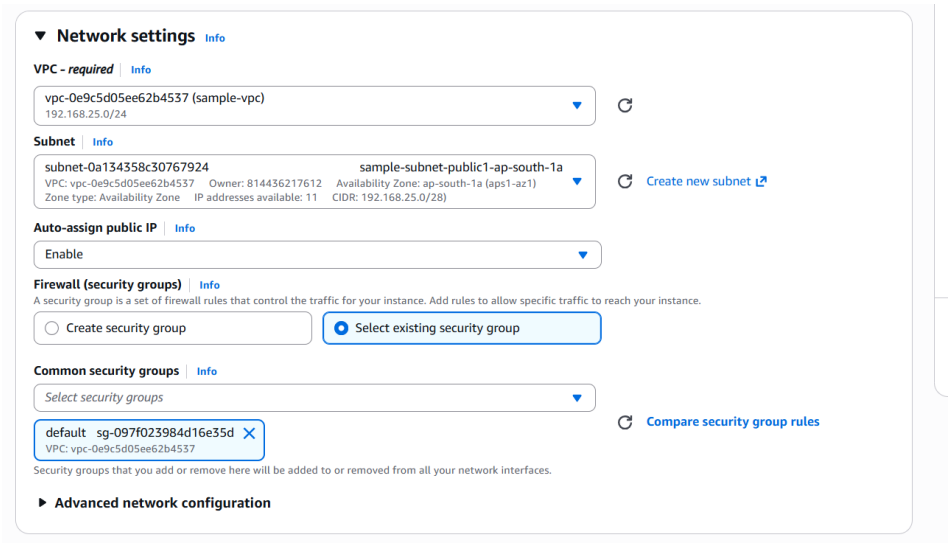
2. Subnets

- Select **2 Public Subnets** in two different Availability Zones
- Select **2 Private Subnets** in the same two Availability Zones.



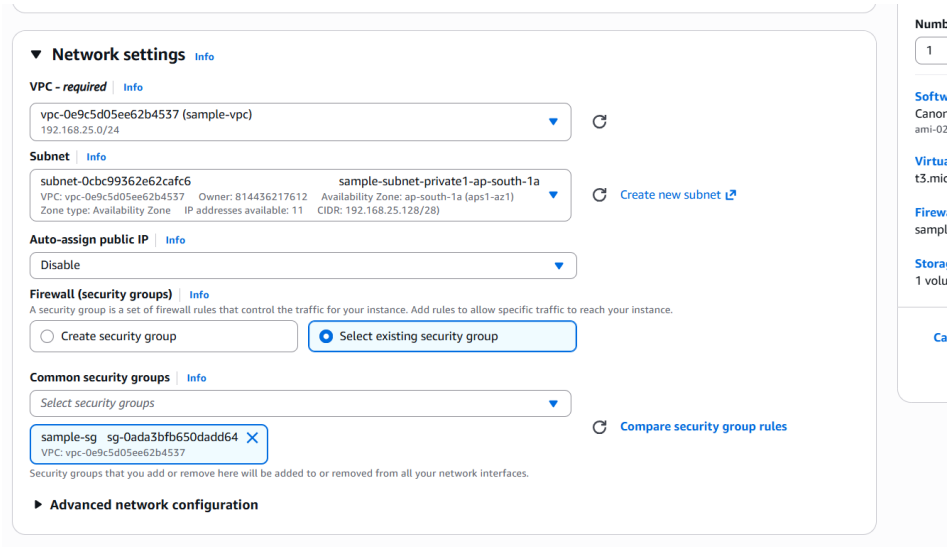
- Create the Security Group and add two inbound rules one for ssh and one for All ICMP – IPv4
- Create a EC2 instances for public → Network → adds the VPC and public subnet we created
- Select the **Auto-assign public IP** as Enable

- Launch instance
- Open **Git Bash** → Go to the **Key Pair** downloaded path
- Connect to your instance using its Public DNS




```
Vishnu Raj@LAPTOP-KSAOBH37 MINGW64 ~/Downloads
$ ssh -i "web_keypair.pem" ubuntu@43.205.119.100
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1015-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Wed Nov 19 17:22:19 UTC 2025

  System load:  0.03              Temperature:           -273.1 C
  Usage of /:   26.3% of 6.71GB   Processes:             114
  Memory usage: 25%               Users logged in:       1
  Swap usage:   0%                IPv4 address for ens5: 192.168.25.8


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Wed Nov 19 17:15:25 2025 from 152.58.217.70
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-192-168-25-8:~$
```

- Create a EC2 instances for private → Network settings → adds the VPC and private subnet we created
- Launch instance
- Using the ping command, we connect the private machine from public machine




```
ubuntu@ip-192-168-25-8:~$ ping 192.168.25.134
PING 192.168.25.134 (192.168.25.134) 56(84) bytes of data.
64 bytes from 192.168.25.134: icmp_seq=1 ttl=64 time=0.175 ms
64 bytes from 192.168.25.134: icmp_seq=2 ttl=64 time=0.173 ms
64 bytes from 192.168.25.134: icmp_seq=3 ttl=64 time=0.202 ms
64 bytes from 192.168.25.134: icmp_seq=4 ttl=64 time=0.157 ms
64 bytes from 192.168.25.134: icmp_seq=5 ttl=64 time=0.161 ms
64 bytes from 192.168.25.134: icmp_seq=6 ttl=64 time=0.159 ms
64 bytes from 192.168.25.134: icmp_seq=7 ttl=64 time=0.163 ms
64 bytes from 192.168.25.134: icmp_seq=8 ttl=64 time=0.163 ms
^C
--- 192.168.25.134 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7169ms
rtt min/avg/max/mdev = 0.157/0.169/0.202/0.013 ms
ubuntu@ip-192-168-25-8:~$
```

3. Internet Connectivity

- While creating the VPC the **Internet Gateway** and **route tables** configure for public subnets to internet access given.