

AWS Reference

Main Reasons for Using AWS

1. Scalability

- Auto Scaling and Elastic Load Balancing let you scale your applications up or down based on demand.
- Perfect for startups, growing businesses, and enterprise-level workloads.

2. Cost-Effectiveness

- Pay-as-you-go pricing model: you only pay for what you use.
- Avoids the need for upfront hardware investments.

3. High Availability & Reliability

- AWS offers multiple Availability Zones (AZs) and Regions worldwide.
- Redundant systems ensure minimal downtime

4. Security

- Strong security infrastructure, including:
 - Data encryption
 - Identity and Access Management (IAM)
 - Compliance with major certifications (ISO, SOC, HIPAA, etc.)

5. Wide Range of Services

- Compute (EC2, Lambda)
- Storage (S3, EBS, Glacier)
- Databases (RDS, DynamoDB)
- AI/ML, IoT, DevOps, Big Data tools, and more

6. Global Infrastructure

- Deploy applications anywhere in the world.
- Low-latency access for users across different regions.

7. Fast Deployment

- You can spin up servers, databases, and entire environments in minutes.
- Great for DevOps and CI/CD practices.

8. Managed Services

- AWS handles backups, updates, and maintenance with services like:
 - RDS (managed databases)
 - Lambda (serverless computing)
 - Fargate (serverless containers)

What is a Region

- A **physical location** in the world (e.g. ap-south-1 = Asia Pacific (Mumbai),)
- Comprised of **multiple, isolated Availability Zones**.
- Each Region is **isolated** from others for fault tolerance and latency control.

Why Are Regions Important

- **Latency:** Choose a region closest to your users to reduce response time.
- **Data residency:** Some organizations must store data in a specific country for compliance.
- **Disaster recovery:** You can replicate systems across regions for fault tolerance.
- **Cost differences:** Some regions may be cheaper than others for the same services.

What is an Availability Zone

- It is **one or more discrete data centers** with **redundant power, networking, and connectivity**.
- Each **Region** contains **at least two Availability Zones** (usually 2 to 6).
- AZs in a Region are **separate from each other**, so if one AZ fails (due to power outage, natural disaster, etc.), the others can keep running.
- All AZs in a Region are connected via **low-latency, high-bandwidth networking**.

Example :- Mumbai Region (ap-south-1)

- **Region Code** : ap-south-1
- **Number of AZs** : 3
- **AZ Names** : ap-south-1a, ap-south-1b, and ap-south-1c.

Why are AZs important

- **High availability:** If one AZ goes down, your application can still run from another.
- **Disaster recovery:** You can replicate data and systems across AZs.
- **Scalability:** You can spread workloads across multiple AZs.
- **Fault isolation:** Issues in one AZ won't affect the others.

Regions



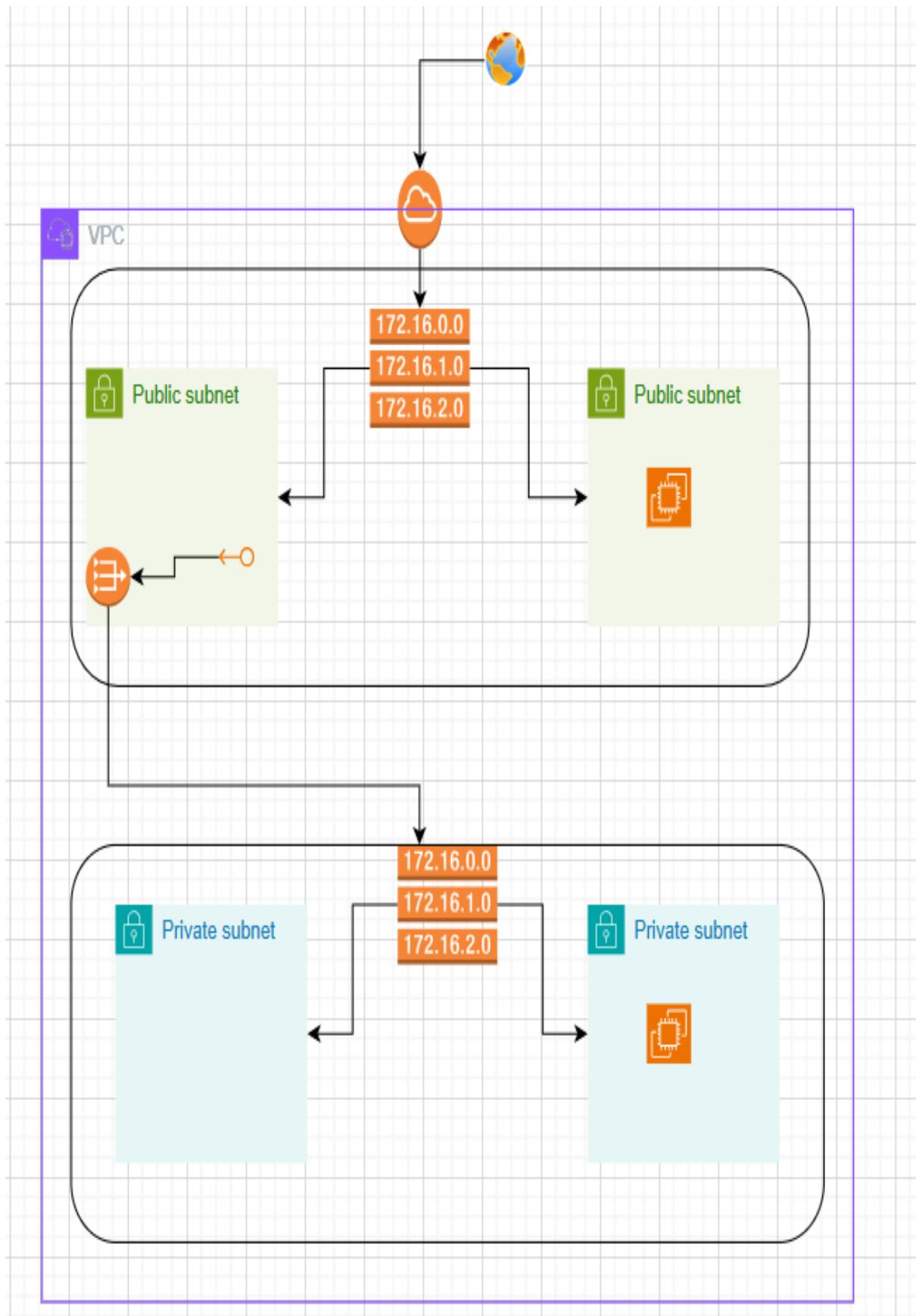
What is a VPC (Virtual Private Cloud)

A VPC is like our **own private data center in the cloud**. It gives you **full control** over our networking environment, including:

- IP address ranges (CIDR blocks)
- Subnets
- Route tables
- Internet gateways
- NAT gateways
- Security groups and network ACLs

Why do we use a VPC

- **Security** :- Keep our resources isolated from other customers. Use firewalls and routing rules to control traffic.
- **Custom Networking** :- Define our own private IP ranges, subnets, and control over routing.
- **Internet Access Control** :- Decide which resources are public (internet-facing) and which are private.
- **VPN / Direct Connect** :- Connect our AWS VPC to our on-premise data center for hybrid cloud setups.
- **Fine-grained Access** : Use security groups and NACLs (Network ACLs) to tightly control inbound/outbound traffic.
- **Multi-Tier Architectures** :- We can separate application layers (e.g., web, app, DB) using public and private subnets.



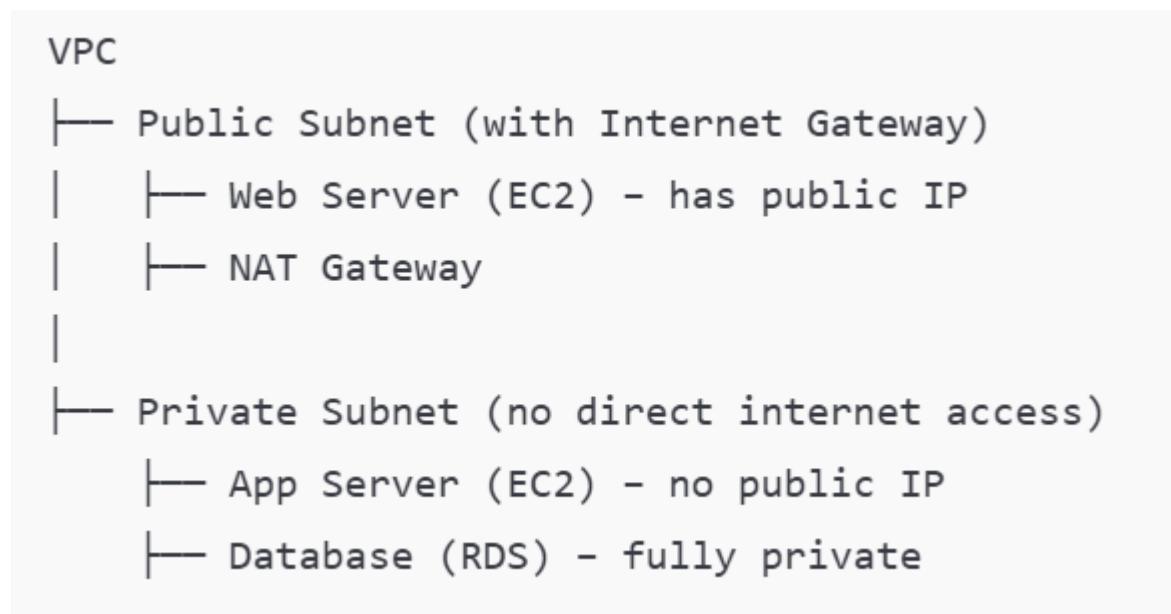
What is a Public Subnet :- A Public Subnet is a subnet inside a VPC that is configured to allow direct communication between its resources and the internet.

We use a Public Subnet in AWS to host resources that need direct access to the internet — such as web servers, bastion hosts, or load balancers.

To be a public subnet, it must meet both of these conditions:

1. It is associated with a route table that has a route to an **Internet Gateway (IGW)**.
2. Resources (like EC2 instances) in that subnet have a **public IPv4 address or Elastic IP**.

Example VPC Architecture



In this setup:

- The **public subnet** contains things that **must talk to the internet**.
- The **private subnet** contains sensitive components that should **not be directly reachable** from the internet.
-

Public Route Table:- A **public route table** is associated with a **public subnet**, which is defined by the presence of a route to the internet through an **Internet Gateway (IGW)**.

- Give internet access to specific resources
- Host public-facing services
- Allow private subnets to connect outward via a NAT Gateway

Example Scenario in AWS:

- **Public Subnet** (associated with a public route table):
 - Contains route: $0 \cdot 0 \cdot 0 \cdot 0 / 0 \rightarrow \text{Internet Gateway}$
 - Can host:
 - Load Balancers
 - NAT Gateways
 - Bastion Hosts
 - Public-facing EC2 instances
- **Private Subnet** (associated with private route table):
 - No direct IGW route.
 - May access internet **indirectly via NAT Gateway** in public subnet.

To **route internet-bound traffic** from a **public subnet** through an **Internet Gateway (IGW)**. Without it, even if an instance has a **public IP**, it **cannot send or receive traffic** from the internet.

Private Route Table:- A **private route table** is associated with a **private subnet**—a subnet that **does not have a direct route to the internet** through an Internet Gateway (IGW). This ensures that resources like databases or internal services are **not exposed publicly**.

- We use **private route tables** in a cloud network (like AWS, Azure, GCP) to control and restrict traffic **within private subnets**.

Why Use a Private Route Table

- **Security:-** Keeps internal resources like databases, application servers, or cache instances **isolated from the public internet**.

- **Access Control**:- Enables fine-grained routing rules—e.g., traffic to the internet only goes through a **NAT Gateway**, or inter-VPC peering is allowed.
- **Internal Communication**:- Allows routing within private subnets or to internal services (e.g., within the same VPC or peered VPCs).
- **Controlled Internet Access**:- If outbound internet access is needed, it's routed through a **NAT Gateway** or **NAT instance**, which is placed in a public subnet.
- **Compliance & Auditing**:- Helps meet compliance requirements by ensuring that sensitive systems are **not internet-accessible**.

Private route tables are used to:

- Protect internal resources
- Avoid direct exposure to the internet
- Control routing policies for private networks

They are essential for **secure, scalable, and compliant network architectures**.

Example Scenario in AWS:

- **Public Subnet** (with public route table):
 - Has a route to $0.0.0.0/0$ via an **Internet Gateway**
 - Used for web servers or load balancers
- **Private Subnet** (with private route table):
 - No direct route to IGW
 - Route to $0.0.0.0/0$ via a **NAT Gateway** (in a public subnet)
 - Used for DB servers, backend apps, etc.

VPC :- I created a custom VPC named **Ganga-VPC** and assigned it a Class A IPv4 CIDR block of **10.0.0.0/16**.

The screenshot shows the AWS VPC dashboard. In the left sidebar, under 'Virtual private cloud', 'Your VPCs' is selected. The main area displays 'Your VPCs (1/2)' with a table showing two entries:

Name	VPC ID	State	Block Public...	IPv4 CIDR
vpc-02dfa10bfec23fec8	Available	Off		172.31.0.0/16
Ganga-VPC	Available	Off		10.0.0.0/16

In the right panel, the details for 'vpc-0014aacb9e6f1314b / Ganga-VPC' are shown:

- DNS resolution: Enabled
- Tenancy: default
- DHCP option set: dopt-04bd2772451334009
- Main route table: rtb-0dad232b75d6a86b3
- Main network ACL: acl-0598c1dada0df9bba
- Default VPC: No
- IPv4 CIDR: 10.0.0.0/16
- IPv6 pool: -

Internet gateway (IGW):- It's allows communication between instances in our **VPC** and the **internet**. Need to **attached** from our **VPC**.

The screenshot shows the AWS Internet gateways dashboard. In the left sidebar, under 'Virtual private cloud', 'Internet gateways' is selected. The main area displays 'Internet gateways (1/1)' with a table showing one entry:

Name	Internet gateway ID	State	VPC ID
Ganga-IGW	igw-07561fe706240401d	Attached	vpc-0014aacb9e6f1314b

In the right panel, the details for 'igw-07561fe706240401d / Ganga-IGW' are shown:

- Internet gateway ID: igw-07561fe706240401d
- State: Attached
- VPC ID: vpc-0014aacb9e6f1314b
- Owner: 427566523194

Subnets:- a **subnet** is a segment of a **VPC's IP address range** where we can place groups of **resources** like **EC2 instances**. I Given name as Public-1-subnet, Public-2-subnet, with range of 10.0.1.0/24 , 10.0.2.0/24. And Private-3-subnet, private-4-subnet with range of 10.0.3.0/24,10.0.4.0/24 Respectively.

Name	Subnet ID	State	VPC
Public2-subnet	subnet-005c5e68c8d12b59c	Available	vpc-0014aabb9e6f1314b Gang...
Public1-subnet	subnet-0dc5fa0aa014c9196	Available	vpc-0014aabb9e6f1314b Gang...
Private3-subnet	subnet-0f58048c533ac549a	Available	vpc-0014aabb9e6f1314b Gang...
Private4-subnet	subnet-0f971315a4b497dbb	Available	vpc-0014aabb9e6f1314b Gang...

Route Table :- a **Route Table** is a set of rules, called **routes**, that control how network traffic is directed within a **VPC**. I created custom Route tables within **Ganga-vpc** named **Public-RT** and **Private-RT** and associated with **subnet association** and **Routes** each othes.

Name	Route table ID	Explicit subnet associations	Edge associations
Private-RT	rtb-066c81495583032e8	2 subnets	-
Public-RT	rtb-006aad57505f2ba91	2 subnets	-

Public subnet associations:- It's a subnet association, establish the communication between public subnets to public route table. We need to associations the respective subnets.

The screenshot shows the AWS VPC Route Tables page. The left sidebar under 'Virtual private cloud' has 'Route tables' selected. The main table lists four route tables: Private-RT, Public-RT (selected), and two others. The Public-RT table details show it has two subnets associated: Public2-subnet (CIDR 10.0.2.0/24) and Public1-subnet (CIDR 10.0.1.0/24).

Name	Route table ID	Explicit subnet associ...	Edge associations
Private-RT	rtb-066c81495583032e8	2 subnets	-
Public-RT	rtb-006aad57505f2ba91	2 subnets	-
-	rtb-0dad232b75d6a86b3	-	-

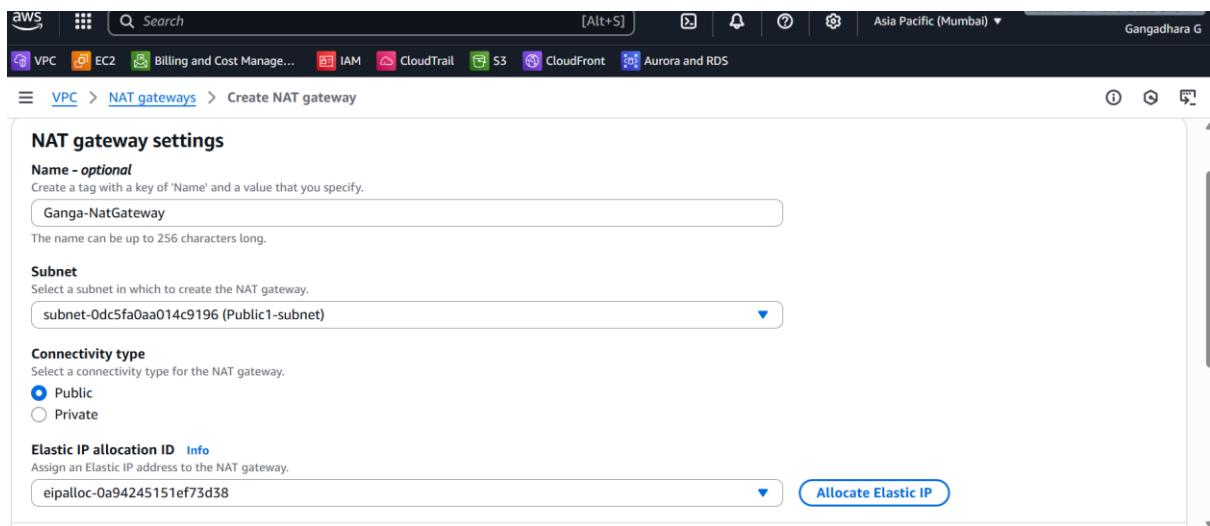
Name	Subnet ID	IPv4 CIDR	IPv6 CIDR
Public2-subnet	subnet-005c5e68c8d12b59c	10.0.2.0/24	-
Public1-subnet	subnet-0dc5fa0aa014c9196	10.0.1.0/24	-

Routes :- Its allows the commination between the internet and Public EC2 instance through the Public Route table.

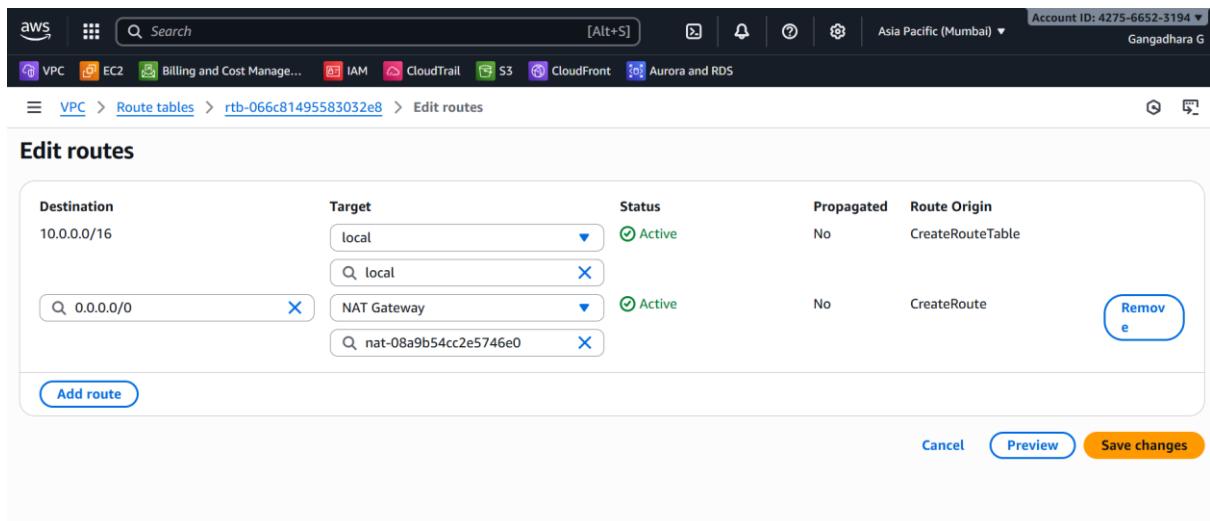
The screenshot shows the AWS VPC Route Tables page. The left sidebar under 'Virtual private cloud' has 'Route tables' selected. The main table lists four route tables: Private-RT, Public-RT (selected), and two others. The Public-RT table details show it has two routes: one to the internet gateway (igw-07561fe70624...) and one to the local subnet.

Destination	Target	Status	Propagated	Route Origin
0.0.0.0/0	igw-07561fe70624...	Active	No	Create Route
10.0.0.0/16	local	Active	No	Create Route Table

Nat Gateway :- is a **network device** that allows **instances in a private subnet** to access the **internet**, when we create the Nat gateway need to allocate the Elastic IP Because An **Elastic IP** in AWS is a **static**.



- After creating Nat gateway need to Routes with in the Public subnets



EC2 :- EC2 is a virtual machine (VM) running on AWS infrastructure that you can use to host applications, websites, databases, or anything else you'd run on a computer or server.

Amazon EC2 provides flexible, resizable, and secure virtual servers in the AWS cloud. You can use it to run anything from small personal apps to large-scale enterprise workloads.

Features

- Scalable Compute
- Wide Range of Types
- Pay-as-You-Go
- Secure
- Flexible OS Choices

Common Use Cases:

- Hosting **web servers** or **APIs**
- Running **databases** or **backend services**
- Performing **data processing** or **batch jobs**
- Hosting **applications** in different environments (dev, test, prod)

Key Components of EC2

Instance :- A virtual server running in AWS

AMI (Amazon Machine Image);- A preconfigured image used to launch an instance (includes OS, software, etc.)

Instance Type:- Defines hardware (CPU, RAM, etc.) for the instance (e.g., t2.micro, m5.large)

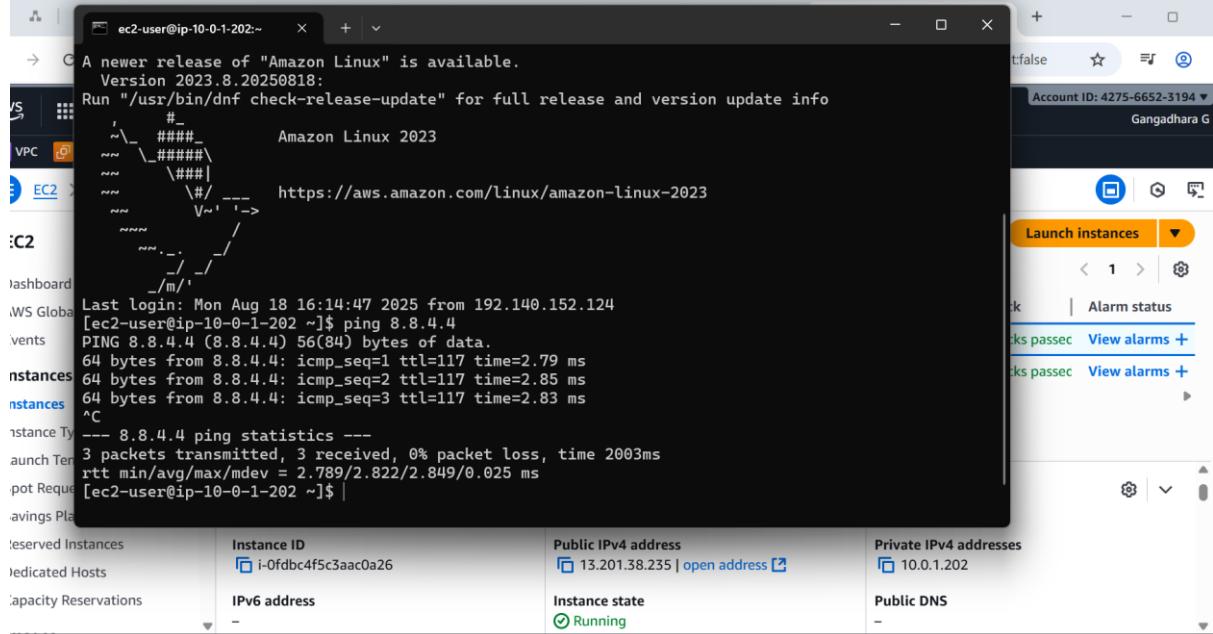
Key Pair:- SSH key used to securely connect to your instance

Security Group:- Acts like a virtual firewall to control inbound/outbound traffic

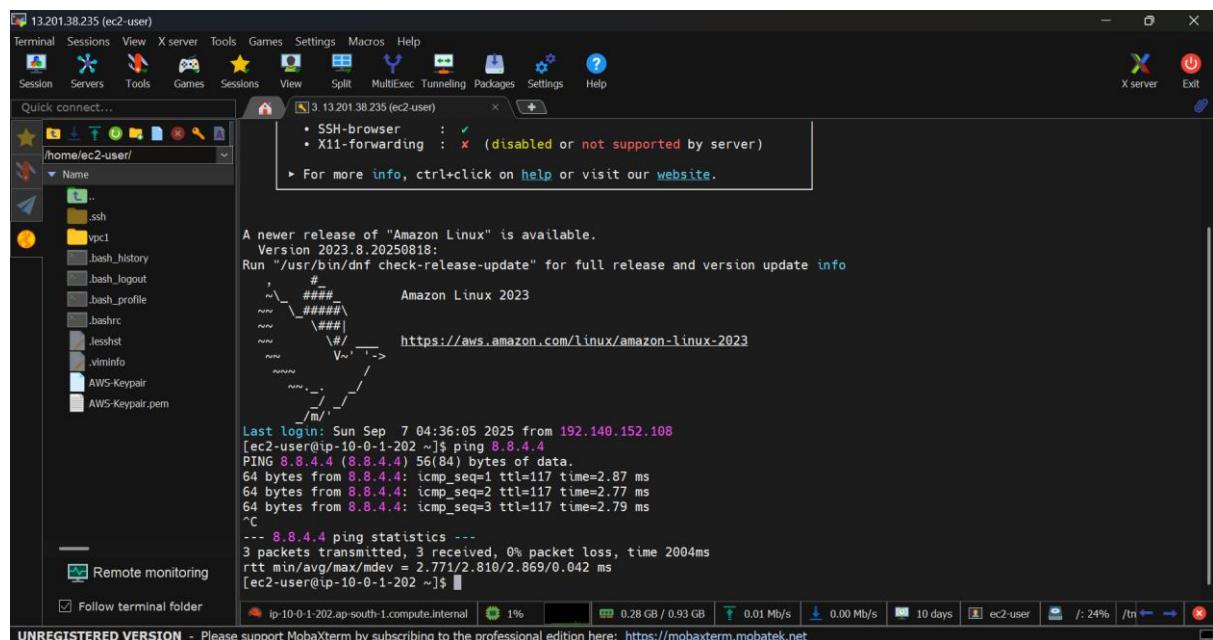
Elastic IP (optional):- A static public IP address for the instance

The screenshot shows the AWS EC2 Instances page. At the top, there's a navigation bar with links for VPC, EC2, Billing and Cost Management, IAM, CloudTrail, S3, CloudFront, and Aurora and RDS. The EC2 link is highlighted. Below the navigation is a search bar and a dashboard summary. On the left, a sidebar lists EC2 navigation options: Dashboard, AWS Global View, Events, Instances (selected), Instances Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, and Capacity Reservations. The main content area displays the 'Instances (1/2)' section. It shows two instances: 'Public-EC2' (Instance ID: i-0fdbcb4f5c3aac0a26, Status: Running, Type: t2.micro) and 'private-EC2' (Instance ID: i-018267fd0446e81e1, Status: Running, Type: t2.micro). Each instance has a 'Details' tab open, showing its Public and Private IPv4 addresses. The Public-EC2 instance also has a 'Status and alarms' tab open.

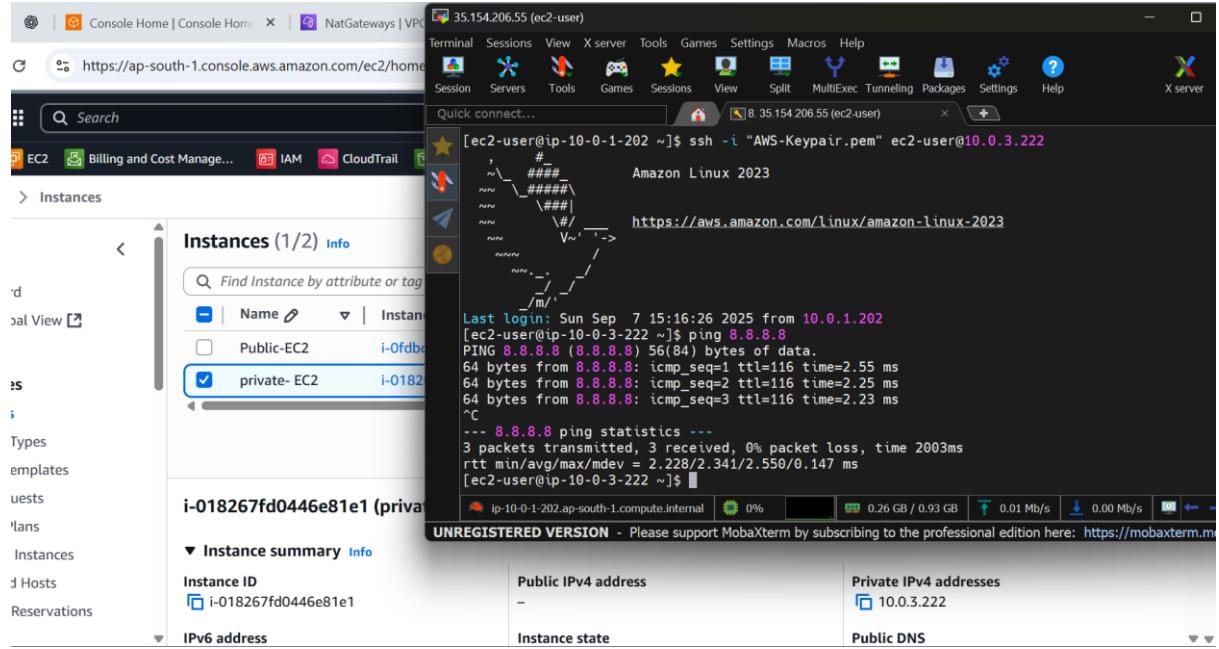
Here, I used the **command line**. Now, communication happens between the internet and the public subnet instance



Here, I used the MobaXterm, communication happens between the internet and the public EC2 instance. 10.0.1.202 → private IPv4 under Public subnet instance.



Here, communication happens between internet and Private EC2 instance through the public EC2 instance. 10.0.3.222 → this IPv4 belongs to private subnet.



The screenshot shows a MobaXterm window titled "35.154.206.55 (ec2-user)". The terminal session displays the following command and output:

```
[ec2-user@ip-10-0-1-202 ~]$ ssh -i "AWS-Keypair.pem" ec2-user@10.0.3.222
,
# 
Amazon Linux 2023
#
https://aws.amazon.com/linux/amazon-linux-2023

Last login: Sun Sep 7 15:16:26 2025 from 10.0.1.202
[ec2-user@ip-10-0-3-222 ~]$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=116 time=2.55 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=116 time=2.25 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=116 time=2.23 ms
^C
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 2.228/2.341/2.550/0.147 ms
[ec2-user@ip-10-0-3-222 ~]$
```

The terminal also shows system status information at the bottom:

```
ip-10-0-1-202.ap-south-1.compute.internal 0% 0.26 GB / 0.93 GB 0.01 Mb/s 0.00 Mb/s
```

The left sidebar of the MobaXterm interface shows the AWS CloudWatch Metrics dashboard, specifically the Instances section, where the "private- EC2" instance is selected.