**Demo on AWS (Amazon Web Services) / Cloud Computing**

**<https://github.com/kumarawsbit/awsbit-1.git> - AWS tutor’s GitHub link**

**AWS is Iaas – Infrastructure as service**

Company called " **Sales force** " first started cloud computing which offers " **Software as a service** " through cloud. AWS – a place where we can define and manage our own virtual network.

**Cloud computing**:

Refers to the delivery of computing services—such as storage, applications, servers, databases, and more—over the internet (the "cloud") instead of relying on local physical hardware or servers.

It allows users to access and manage data and applications remotely, with greater flexibility and cost-effectiveness.

**USES:**

- Scalability: Resources can be scaled up or down based on demand.

- Cost Efficiency: Reduces expenses on physical hardware and maintenance

- Accessibility: Services can be accessed from anywhere in the world with an internet connection.

**Types of Cloud Computing:**

1. Public Cloud: The cloud services are managed by third-party providers like AWS,

Microsoft Azure, or Google Cloud and are shared among multiple

users.

Examples: Ideal for general-purpose applications like Gmail, Google Drive,

Dropbox.

2. Private Cloud: A cloud environment dedicated to a single organization for

exclusive use. Managed internally or by a third party.

Examples: Used by banks, government agencies, or enterprises handling sensitive

data.

1. Hybrid Cloud: Combines private and public clouds, enabling data and

applications to be shared between them.

Sensitive data can be stored in a private cloud, while less critical

data can reside in the public cloud.

Examples: Commonly used by multinational corporations.

**4 ways to communicate with Aws:**

* Using web-based Application i.e., AWS Management console – it’s is a visual user interface
* Using CLI (Linux/mac/windows) - users who prefer scripting or automating tasks.
* By Programmatically access AWS services (Software Development Kit available in multiple programming languages - c++, c#, Go, Python, Java, Nodejs etc.,)
* Using cloud Formation Template (Json / yaml)

**Terminologies used in AWS and Networking**

Virtual Private Cloud (VPC)

* + VPC is used to provide a **secure and customizable networking environment** for our AWS resources.
  + Creating VPC is the foundational step when setting up our network infrastructure.
  + When we want our servers not be exposed to public then we create them in our private cloud (called VPC) in AWS.
  + It functions as our private network within the public AWS cloud.
  + **VPC** is private Network which also supports Public sub-Network with in it.
  + You create a VPC to have control over the virtual network environment, including IP address ranges, subnets, routing tables, network gateways, and security settings.

Network – one or more servers grouped together to communicate each other – Local Area Network (LAN).

Intranet Application – Application (example SAP) that is accessed by other servers within the network.

Internet Application – Application within a Network can be accessible by other servers from other Networks over internet.

CIDR (classless inter-domain routing) – Defines the size of a network.

**CIDR Notation Syntax: IP Address/N ;** The starting address of the network / Number of bits in Network mask (e.g., 192.168.1.0/24)

CIDR allows creating custom network sizes instead of being restricted to fixed classes (Class A, B, C).

You can use CIDR notation to divide a network into smaller subnets by adjusting the network mask (/N value) to create custom subnets.

Subnet – Dividing the servers into groups within the Network.

Each group of servers is called subnet.

DNS (Domain Name System) - is a fundamental part of the internet.

Every device or server on the internet has a unique IP address.

Example: Google’s IP address might be 142.250.183.238.

Remembering these IP addresses is difficult for humans.  
Instead, we use domain names like **www.google.com**, and

DNS translates these **human-readable domain names** (like www.google.com) into **machine-readable IP addresses** (like 142.250.183.238).

Steps in Mapping domain name into specific IP address:

* + - Request to www.google.com in browser
    - our Internet Service Provider's (ISP) DNS server or a third-party DNS server handles this request
    - TLD (Top-Level Domain) servers helps in locating the IP address of servers (like .com, .org etc.,)
    - TLD points to Authoritative DNS Server which holds the exact IP address for the domain and sends it back to the DNS resolver
    - Once our browser gets the IP address, it connects to the web server and loads the website.

Admin Steps to Add a windows PC to an Organisation Domain (let say infosys.com):

* + - Open the computer’s System settings.
    - At ‘About’ option
      * select “join to domain’ or
      * Select ‘Advance System Settings’, under ‘Computer Name’ click on ‘change’ button
    - Enter the domain name (e.g., infosys.com).
    - When our PC tries to join infosys.com, it sends a query to the DNS server.
    - The DNS server resolves the domain name to the IP address of the **Domain Controller (DC) that** manage user authentication and Active Directory.
    - Provide administrative credentials to complete the process.
    - our PC communicates with the **Domain Controller (DC)** of the organization.
    - The DC verifies your PC and creates a unique computer account in Active Directory.

Active Directory: It acts as a database and a set of services to help administrators manage permissions, resources, and user access efficiently within a network.

Uses:

* Login Management
* ensures users have access only to the resources they are permitted to use (e.g., printers, files, applications).
* Group Policies set by the IT team, like password complexity requirements or session timeout, are enforced automatically.

NTP (Network Time Protocol): is a protocol used to synchronize the clocks of computers and devices across a network.

It ensures that all systems maintain accurate time.

**IP address (Internet Protocol address)** is a unique numerical label assigned to each device connected to a network that uses the Internet Protocol for communication.

* **public IP addresses** in AWS are generally dynamic in nature
  + When you launch an instance in a public subnet with the **"Auto-assign public IP"** option enabled, AWS assigns a public IP dynamically from its pool of public IP addresses.
  + These public IP addresses Change if the instance is stopped and restarted.
  + These public IPs **are not persistent** because they are not associated with your AWS account.
* **Elastic IPs** in AWS are static in nature
  + If you need a **static public IP**, AWS provides **Elastic IPs which are** **persistent** public IPs allocated to your account.
  + They can be associated with an instance, and the IP will not change even if the instance is stopped and restarted.

**Structure of IP Addresses:** In 10.0.0.1

10: Represents the Network ID.

0.0.1: Represents the Host ID i.e., specific device in

the network.

* **IPv4 (Internet Protocol version 4)** uses 32 bits size, divided into four 8-bit octets, separated by dots.

 Each octet can range from 0-255

**CIDR Notation - Specifies the IP address range in a Network:**

Let say Network should have IP address range - 10.0.0.0/24

The **number** after the slash (/) indicates how many bits in the IP address are reserved for the ***Network part***.

IPv4 addresses are 32 bits long, which splits into **network bits** and **host bits**

/24 means **24 bits** are reserved for the network, and the remaining **8 bits** are for hosts.

So, in each IP address of range 10.0.0.0/24,

The portion 10.0.0 is reserved for Network i.e., three 8-bits octet–8+8+8=24

The next portion 0 for Host/devices i.e., one 8-bit octet which ranges from 0-255,

which means for 10.0.0.0/24 the IP addresses rage will start from

10.0.0.0, 10.0.0.1,10.0.0.2... to 10.0.0.255 = Total 256 IP addresses

0.0.0.0/8 -> 0.0.0.0 to 0.255.255.255 = Total 16,777,216 IP address

However, Usable IP addresses are 254 since 2 addresses are reserved: one for the network ID (i.e., starting address 10.0.0.0) and one for the broadcast address (i.e., Ending address 10.0.0.255)

Simple Formula to calculate the total Ip addresses in any Range:

Say, 10.0.0.0/32 --- 2 ^ (32-32) = 2^0 = 1 ---- Mini range

10.0.0.0/16

* + - * Total bits in an IPv4 address: 32
      * Network bits: 16
      * Host bits: 32−16 => 16 --- 2 ^ 16 = 65536

10.0.0.0/8 --- 2 ^ (32-8) =2^24 = 1678Cr --- Last range in IPv4.

To have higher range than 10.0.0.0/8 then, we have IPv6.

IP Range for 10.0.0.0/28

Network Portion 28 bits i.e.,

00001010.00000000.00000000.0000

Host Portion 4 bits i.e., 0000 to 1111

**IP Address Standards:**

### **RFC 1918 , a** standardthat defines how to assign private IP addresses to devices **on a TCP/IP network** such as using private IP addresses for local networks and public IP addresses for internet-facing devices.

### RFC 1918, was created by **the Internet Engineering Task Force (IETF).**

### IP address standards were created to ensure seamless communication between devices worldwide, provide unique device identification, and support the scalability of the growing internet.

### The transition from IPv4 to IPv6 highlights the adaptability of these standards to accommodate technological advancements and the increasing number of devices.

**IPv4’s** **32-bit addressing scheme** could only support approximately **4.3 billion unique addresses**, which became insufficient due to the explosion of internet-connected devices.

**IPv6** introduced **128-bit addresses**, offering **3.4×10³⁸ unique addresses** to meet current and future needs.

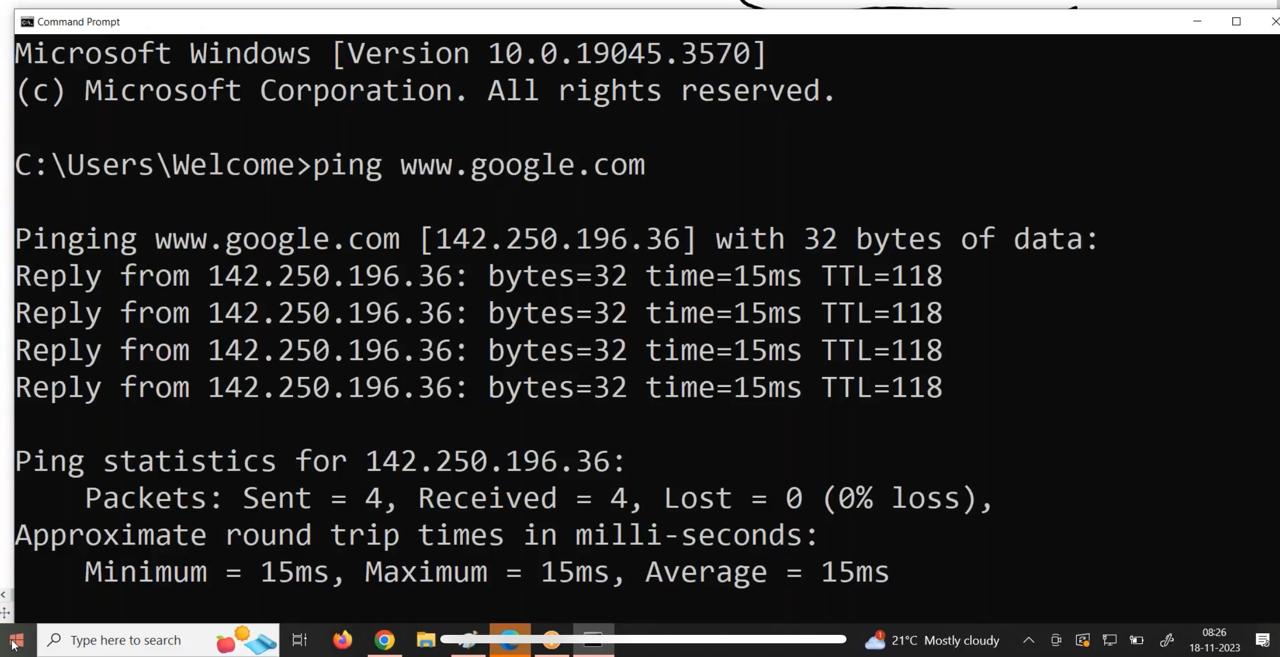
* **IPv4 (I**nternet Protocol version 4 standards established in **1981):**
  + 32-bit addressing (e.g., 192.168.1.1).
  + Address classes (A, B, C, D, E).
  + Subnetting to organize networks.

#### https://en.wikipedia.org/wiki/Reserved\_IP\_addresses

### **Private IP Addresses**: Used within private networks, not routable on the internet.

* + Class A: 10.0.0.0 to 10.255.255.255 – for large Networks
  + Class B: 172.16.0.0 to 172.31.255.255 – for medium Networks
  + Class C: 192.168.0.0 to 192.168.255.255 – for small Network
* **Loopback Private Addresses to Local Host**: 127.0.0.0 to 127.255.255.255 - Used for testing (e.g., 127.0.0.1).
* **Link-Local Private addresses between two hosts:** 169.254.0.0 to 169.254.255.255 (comes under Subnet scope)
* **Public IP addresses**: **Any IP address out of above-mentioned ranges.**
* **IPv6 (1998):**
  + 128-bit addressing (e.g., **2001:0db8:85a3:0000:0000:8a2e:0370:7334**).
  + Eliminates the need for NAT (Network Address Translation).
  + Built-in security features (IPsec).

**How to get an IP address of a Domain Name:**



DHCP **(Dynamic Host Configuration Protocol)** is a network management protocol used to automatically/dynamically assign IP addresses and other network configuration settings to devices (clients) within a network.

* **AWS installs and manages the DHCP functionality automatically** for every VPC.
* When you create a VPC, AWS automatically sets up a DHCP server and assign IP address to it - for managing the assignment of IP addresses to the instances launched within that VPC.

Edge Locations in AWS are part of the 'Amazon **CloudFront Content Delivery Network** (CDN)', designed to deliver frequently accessed content (like web pages, videos, or software) to users with

- low latency(delay) by minimizing the distance data has to travel and,

- high transfer speeds.

CloudFront uses these edge locations to store cached copies of content from the

origin servers (e.g., an S3 bucket or an HTTP server).

When a user requests content, CloudFront delivers it from the nearest edge

location instead of the origin, ensuring faster delivery.

Gateway: A **gateway** is a network node that acts as a bridge between two different networks.

* A **default gateway** is the device (often a router) that connects a local network to external networks like the internet.
  + - The gateway/router receives the data and forwards it to the internet (via your ISP).
    - Incoming data from the internet is routed back to your device by the gateway.
  + A Network Gateway Connects two or more different networks and allows communication between them.
    - Many gateways include firewall features to filter traffic for security purposes.

**Port Forwarding** : In a private network (behind a router), devices have **private IPs**, which are not directly accessible on the public internet.

If you want to access a private network device (like a PC, camera, or game server) from the internet, **port forwarding** technique that allows external devices on the internet to communicate with correct private IP device inside a private network.

**In port forwarding**, the router uses the **Unique** **public IP + port number** combination to decide which device inside the network should receive the request.

**How the Internet Request processed from PC:**

* Nuvu PC browser lo google.com or 142.250.190.46 (Google IP address) ani type chesthe, request start ayyedi PC lo unna TCP/IP stack nunchi.
* Router lo **DHCP (Dynamic Host Configuration Protocol)** service untundi. dhani dwara WiFi Router devices ki unique Private IP addresses ni dynamically allocate chesthundi.
* All device’s private IPs are only valid within your local network and not visible on the internet. Private IPs can be reused in different homes or networks because they are not visible on the public internet.
* PC has a Routing table used to determine where to send network packets. Routing Table ensures that network traffic is correctly routed within a local network or across the internet.
* PC ki already router private IP address telusu (192.168.x.x range lo) through a process called Default Gateway.
  + **Router Private IP Address** నెట్‌వర్క్ యొక్క **Default Gateway** గా పని చేస్తుంది.
  + PC లు Router తో మాట్లాడేందుకు **Gateway** **IP** ని ఉపయోగిస్తాయి. Ee router private IP address PC lo settings lo set avutundi, so data router ki vellali ani telustundi.
* PC Request ni Packets ga divide chestundi (small data units).

Ee Packets lo ,

1. Source IP: PC private IP (192.168.1.x)
2. Destination IP: Google public IP (142.250.190.46)

* PC nunchi packets Ethernet (LAN cable) or Wi-Fi router ki vellutayi.
* Router packets ni chustundi, and Google public IP ki pampali ani decide chestundi.

• Router ki Public IP address untundi (ISP ichindi).

* <https://whatismyipaddress.com/> or <https://www.whatismyip.com/> - Which shows your ISP provided Public IP address
* Internet lo data exchange cheyyataniki **Public IP** avasaram untundi
* Ee Public IP address use chesi router packets ni ISP ki forward chestundi.
* **In context of home internet,** the public IP assigned to each home depends on how the ISP manages their IP address allocation.
  + As IPv4 addresses are limited, so many ISPs use **CGNAT Carrier-Grade NAT (CGNAT)**, to save public IPs by assigning same public IP to multiple customers.
  + In this case, inside each house, the router assigns private IPs (e.g., 192.168.x.x or 10.x.x.x) to the devices within the local network.
* When devices in both houses access the internet, their traffic is routed through the same shared public IP address.
* When multiple customers share the same public IP, the ISP uses source port numbers to identify which device (and home) sent the request.

Example: Every internet request from a device includes a

unique **source port number** in addition to the **public IP**.

* + - * Private IP + Port (e.g., 192.168.1.2:50000)
      * Corresponding public IP + Port (e.g., 203.0.113.1:50000)
  + Some ISPs still have enough IPv4 addresses or use IPv6 to provide unique public IPs to each customer then each home has its own distinct public IP.

• Router lo NAT (Network Address Translation) untundi. Idi private IP (PC IP) ni

public IP ga convert chestundi, so Google server ki understand avutundi.

* All devices (PCs, mobiles, etc.) in your local network have private IP addresses (e.g., 192.168.1.2, 192.168.1.3).
* When these devices send requests to the internet, the router's NAT (Network Address Translation) mechanism **converts** each device's private IP into the router's public IP (**assigned by the ISP**).
* Router uses Network Address Translation (NAT) to keep track of which local device (e.g., your PC) made the request.
* ISP nunchi, packets routers and switches dvara Google server ki reach avutayi.

**Simple ga cheppali ante:**

PC request router ki velladaniki Default Gateway address ni use chestundi. Router packets ni internet lo forward chestundi public IP address dvara.

Internet request from our PC -> Router -> ISP -> Internet (Google server ki) -> Google response -> PC

AWS **VPC (Virtual Private Cloud) ki internet connectivity provide cheyyataniki specific resources and configurations:**

AWS itself acts as the Internet Service Provider (ISP) for all its resources.

Internet Gateway (IGW):

* **IGW Used for public subnets.**
* If you want your VPC resources to access internet, you need to attach an **Internet Gateway** to your VPC**.**
* AWS lo Internet Gateway (IGW) ane resource internet access provide chestundi.
* IGW is a bridge between VPC and public internet. All traffic from public and private subnet must pass through IGW to communicate on internet.
* IGW ni VPC ki attach chesthe, IGW allows **direct** inbound and outbound

communication between the internet and the instances in your VPC that

**have public IPs.**

* Hence IGW **does not perform** Network **Address Translation; as it directly routes traffic.**

**NOTE:** Elastic IP (Public IP): **Instances must have public IP or Elastic IP to use IGW.**

**IGW** is used byPublic subnet instances and NAT Gateway for direct internet

access.

AWS Internet gateways(IGW) --------- are for IPv4

AWS Egress-only internet gateways ------- are for IPv6

**Use Case:**

When you want public-facing instances (e.g., web servers) in a public subnet to

communicate with the internet directly.

NAT Gateway (Private Instances):

* **NAT Used for private subnets but is reside in Public Subnet**, as it needs direct access to **IGW** (An internet gate way which only allows instances with Public IPs or Elastic IPs to communicate with Internet) **to forward traffic between the private subnet instances and the internet.**
* If **NAT Gateway** placed in **private subnet**, it would also lack direct internet access since Only public subnet **have a route** to the **IGW** while Private subnet **have no route** to the **IGW** Therefore **NAT must reside in Public Subnet**,
* In general, **private network** assigned with **private IP addresses** (like 192.168.x.x, 10.x.x.x, or 172.16.x.x), which are **not routable** on the public internet.
* To communicate on the internet, each instance must need a unique public IP address.
* Private instances do not need public IPs as they don’t want to expose on internet.
* VPC lo unna private instances ki direct ga internet access undadu as they assigned with Private IP addresses not with Public.
* Ee private instances ki internet access kavali ante, NAT Gateway or NAT Instance configure chestaru.
* While creating the NAT Gateway, you specify the **public subnet**, and AWS automatically **assigns a private IP from the public subnet's range** to the NAT Gateway.
* This **private IP** is used by instances in the private subnet to communicate with the NAT Gateway.
* When instances in Private Subnet send internet-bound traffic , the traffic is routed to **NAT Gateway** using the **private subnet’s route table** then **NAT Gateway** translates the **private IPs** **of instances** in the **private subnet** to the NAT Gateway’s **public** **IP** (**Elastic IP**) and forward requests to **IGW** **(**Since the **NAT Gateway** resides in the **public subnet**, its traffic is automatically routed to the **IGW** because the **public subnet's route table** has a route directing **0.0.0.0/0** to the **Internet Gateway**.**)** to communicate with internet.
* While creating NAT Gateway explicitly assigned with Elastic IP (**public IP**).

This setup ensures a **secure and controlled flow of internet-bound traffic** from **private subnets** without exposing their instances directly to the internet.

**NOTE:**

* NAT Gateway allows instances in private subnets to access **the internet outbound only**, but **prevents inbound traffic from the internet**.
* A NAT Gateway has a **private IP address** assigned to it **from the private subnet** it is located in. After internet request processed the assigned **NAT** **private IP** helps to connect back from NAT to private subnet instances.
* **NAT** is only needed to communicate with the internet, not for internal

instance-to-instance communication.

* **NAT Gateway** is an AWS-managed service. While it likely **runs on a Linux-based**

**system in the backend**, AWS abstracts all the underlying details.

* **NAT Instance** - A self-managed Amazon EC2 instance configured to perform NAT.

It essentially functions as a NAT server.

If you set up a NAT Instance using an EC2 instance, you will typically

choose a Linux-based AMI (such as Amazon Linux, Ubuntu, or CentOS)

and configure it as a NAT server manually.

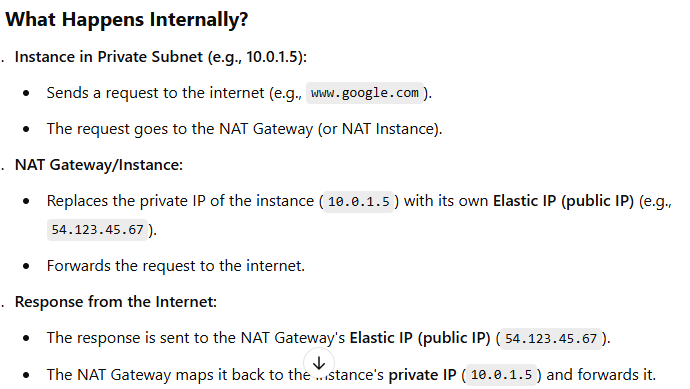
**Use Case:**

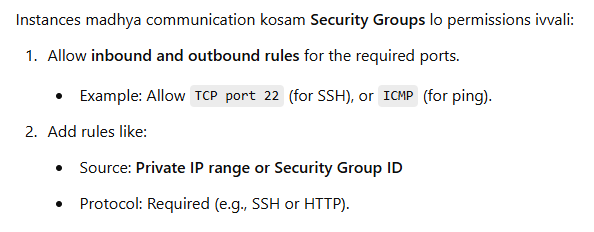
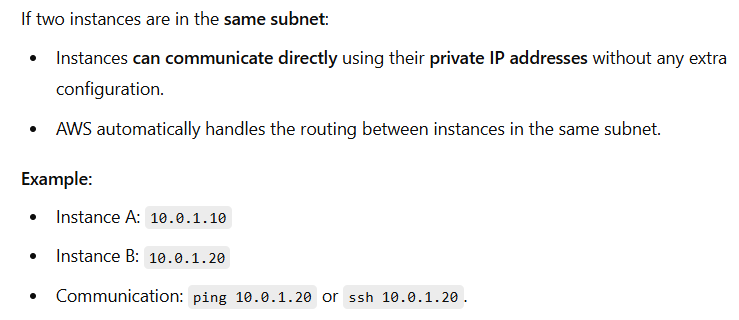
When outbound internet access is required without exposing the instances to

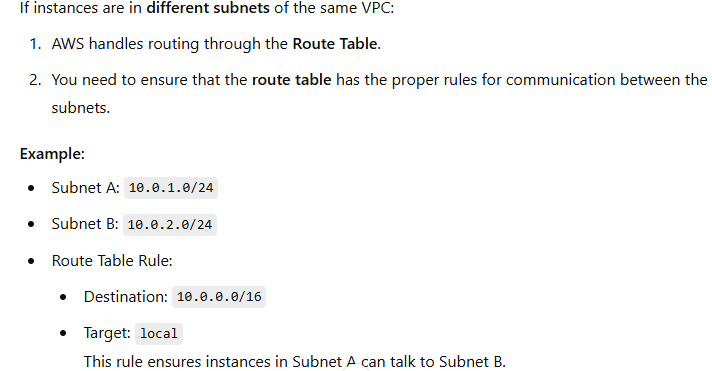
the internet.

When private instances (e.g., database servers) need to download

updates/patches.





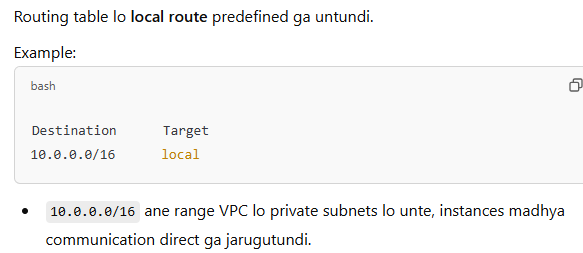


Routing Table:

VPC lo request anedi Private instance to NAT or local instance to instance vellali

anedi **Routing table** lo entry cheyyali.

1. Instances within the same private subnet can communicate with each other **directly** without additional routing table configuration
2. configuration at Route Table to enable communication between private subnets within same VPC



Similarly,

1. Private Subnet lo unna instances via NAT outbound internet requests cheyyali ante

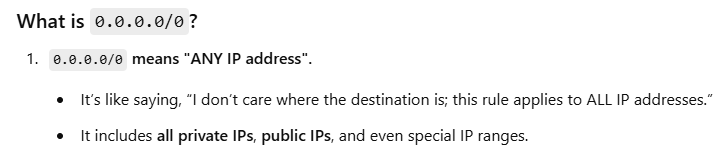
**Destination** **0.0.0.0/0 (any IP)** ni **Target** **nat-gateway-id** kiroute cheyyali.

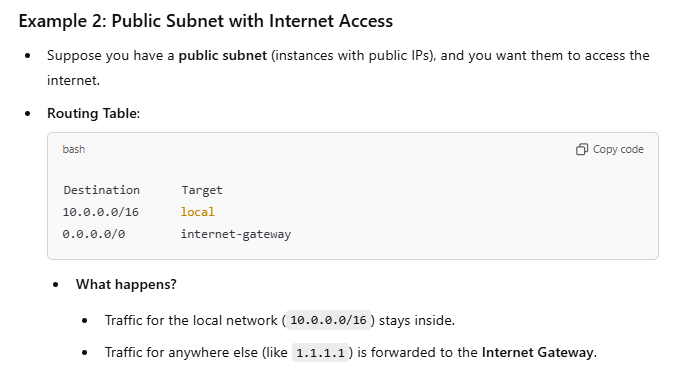
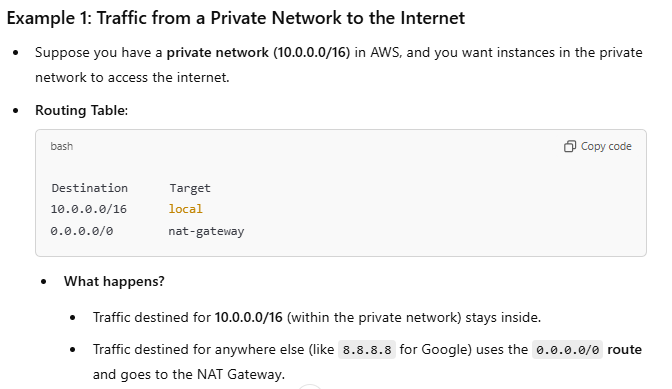
1. Routing Table configuration for Public Subnet

**Destination 0.0.0.0/0 (any IP)** ni **Target** **Internet Gateway** ki route cheyyali.

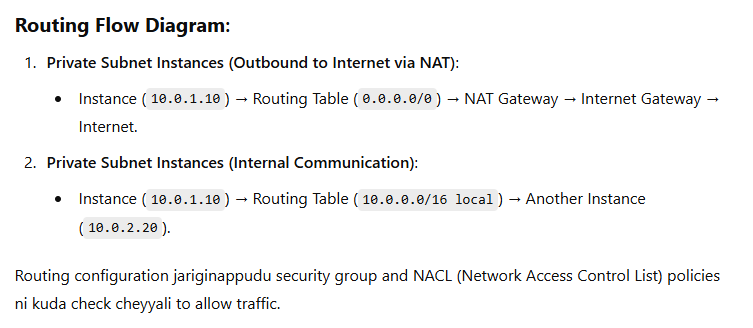
**NOTE:**

* **above routes ni route table lo configure cheyyakapothe**, instances ki internet access undadu.
* When we create **VPC** in **AWS**, Network routing will auto define by AWS
  + **Public subnet** will associate with different Route Table which has route to `**Internet gateway i.e., igw**`
  + **Private Subnet** will associate with different Route Table which has route to **NAT**
* In practice in a routing table, 0.0.0.0/0 is typically for internet-bound traffic.
* A subnet is considered "**public**" when its **route table** has a route pointing to the **IGW**.





Traffic IGW dvara internet ki vellutundi.



**Private Networks having private IP** are designed to operate in a controlled and secure environment, such as within a company or a cloud provider's infrastructure (like AWS VPC).

* Private networks typically don't have internet access by default for **security and isolation** purposes.
* Devices in a private network are **not routable on the public internet as they** are assigned with **private IP addresses**.
* Internet routers do not recognize private IP ranges which are not designed for use on the public internet.
* Private IP addresses are reserved for use only within the local private network. They are not unique globally – different private networks can reuse the same private IP address.
* **Private** networks are intentionally isolated to protect devices from unauthorized access or attacks originating from the internet.
* Without internet access, internal systems like databases, application servers, or backend services are shielded from external threats.
* If devices in a private network need internet access, they require **NAT (Network Address Translation)** to convert their private IP addresses into public IP addresses. Without NAT, private IPs can't communicate with the internet.

Dividing a VPC network into **public and private subnets** within a Virtual Private Cloud (VPC) so that

public subnet servers can be exposed to the world and private subnet servers can be exposed within company

* In General, when we divide **Private network** into **Public** & **Private** Subnets, the servers created inside these two subnets are assigned with private IP address only.

NOTE: Private IPs can be reused in different homes or networks because they

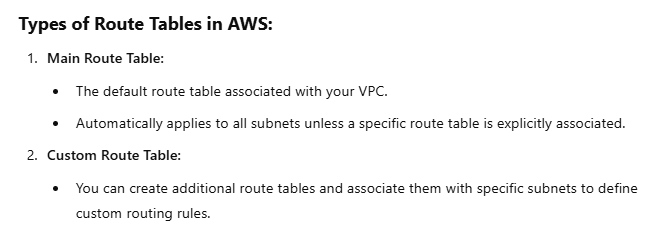
are not visible on the public internet.

* In order to expose instances from public subnet to internet then those each instance must be assigned with **unique** **Public IP address** along Private IP address**.**
  + - Whenever Public IP address assigned to a server then that server will get direct access to internet.
    - AWS assigns a public IP from its pool of IP addresses to each instance when instance is in public subnet and Auto-assign Public IP is enabled during instance launch.

NOTE:

* + - Instead of using the automatically assigned public IP, you can associate an **Elastic IP** with the instance.
      * A public IP is released when the instance is stopped or terminated.
      * Elastic IPs are static public IPs that you can retain even if the instance is stopped or terminated.
    - To have internet access simply assigning a **public IP** to an instance isn't enough because traffic must pass through the AWS’s **IGW** to reach the internet.
      * **Without an IGW,** the AWS infrastructure will not recognize the path to route internet-bound traffic from the VPC.

Defining a route table to Subnets in Virtual Private Cloud (VPC) ensures that traffic inside the VPC is properly routed to its destination, whether that's another instance, another subnet, or the internet.



Without Route Table, Instances wouldn’t know where to send traffic.

In a VPC, you may have **public subnets** (with internet access) and **private subnets** (without direct internet access). The route table helps define

* + - Routing public subnet traffic to the **Internet Gateway (IGW)** for direct internet access
    - Routing private subnet traffic to a **NAT Gateway/Instance** for controlled internet access.

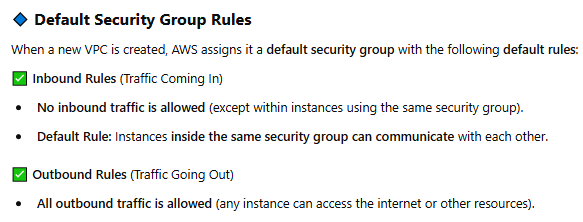
**Two Types of securities are provided by AWS**

1. A **Network Access Control List (NACL)** is a virtual firewall in AWS that controls inbound and outbound traffic for **one or more subnets** in a **VPC (Virtual Private Cloud)**. It operates at the **subnet level**, providing an additional layer of security.

* **Stateless:** NACLs are stateless, meaning each request and response is evaluated separately. For example, if an inbound rule allows traffic, you must explicitly create an outbound rule to allow the response.
* Default behaviour of NACL **ALLOWS** all inbound and outbound traffic.
* A **custom NACL** denies all traffic by default unless rules are explicitly added.
* NACLs use numbered rules (in ascending order) to allow or deny traffic. The evaluation stops as soon as a rule match.

1. **Security Group (SG)** is a virtual firewall at the **instance level** in AWS that controls **inbound and outbound traffic** for Amazon **EC2** **instances**. It allows or denies specific types of traffic based on defined rules.

* Unlike NACLs, Security Groups are **stateful**, meaning if an inbound rule allows traffic, the response traffic is automatically allowed without an explicit outbound rule (and vice versa).
* Default behaviour - All inbound traffic is **DENIED** by default and All outbound traffic is **ALLOWED** by default.



**AWS in Practice**

**Creation of VPC ( AWS online class-6 - date 27/11/2023)**

* Remember
  + - VPC is Region based.
    - Plan CIDR allocation based on future scaling needs.
      * To attach a **Load Balancer(**distributes incoming traffic across multiple **EC2 instances in diff AZs** to improve **availability, scalability, and fault tolerance**) to a VPC, each subnet has a CIDR block with at least a /27 bitmask and at least 8 free IP addresses.
* At dashboard select VPC -> Create VPC
  1. **Select** VPC and more which create Private Network with additional network resources.
  2. **Give a name to** Name tag auto-generation which can be auto-generated for all resources in VPC in order to identify them easily.
  3. **Select** available CIDR notation in IPv4 CIDR block which determine the starting IP and size of our VPC (e.g., 10.0.0.0/24)
     + From assigned CIDR block to VCP,AWS reserves **five IP addresses** from each subnet within the VPC that are used for –
       - 1. **First IP Address: 10.0.0.0** from 10.0.0.0/24 **used to** Identify the subnet itself.
         2. **Second IP Address: 10.0.0.1** from 10.0.0.0/24 **-** Reserved for the subnet's default gateway to route traffic.
         3. **Third IP Address: 10.0.0.2** from 10.0.0.0/24 – Reserved by AWS for AWS DNS Server
         4. **Fourth IP Address: 10.0.0.3** from 10.0.0.0/24 -Reserved by AWS for Future Use
         5. **Last IP Address: 10.0.0.255** from 10.0.0.0/24 - Reserved by AWS as the broadcast address (However broadcast communication is not supported in VPCs).
  4. **Under** IPv6 CIDR block – Leave this with auto selected option (i.e., No IPv6 CIDR

Block)

* 1. **Select** Tenancy as Default – This Option refers to how the underlying hardware (physical servers) is allocated to the instances running in your VPC
     + Default Tenancy - Instances are launched on shared hardware, where multiple customers' instances may run on the same physical host
     + Dedicated Tenancy - Instances are launched on hardware that is physically isolated from instances of other AWS customers
  2. **Select** Number of Availability Zones (AZs) - at least 2 or more
     + - An **Availability Zone** is a **physically isolated data centre** within an AWS Region that is designed for high availability.
       - Each AZ has **independent** power, cooling, and networking but is connected to other AZs in the same Region with **low-latency, high-speed network links**.
       - By deploying resources (e.g., EC2 instances, RDS databases) across multiple AZs, you can ensure that your application continues to operate even if one AZ fails.
  3. **Select** Number of public subnets - Use public subnets for web applications that need to be publicly accessible over the internet.
  4. **Select** Number of private subnets - Use private subnets to secure backend resources that don't need public access.
     + **At Customize subnets CIDR blocks –** we can increase or decrease the subnet CIDR range as per our IP address requirement.

**NOTE**: Subnet CIDR is **fixed** within the VPC and **cannot be changed after creation**.

* 1. **NAT** gateways --- Choose the number of Availability Zones (AZs) in which to create NAT gateways. **Note** that there is a charge for each NAT gateway.
     + - **NAT gateways** enable resources in private subnets to reach the internet. External services, however, cannot initiate a connection with the resources in the private subnets.
       - If None selected no internet access to private subnets in VPC.
       - **NAT** is configured and managed manually by us whereas, **NAT Gateways** are fully managed by AWS
       - **How to delete a NAT** - There are **2 steps** to delete a NAT

1. Under VPC dashboard – go to NAT gateways
   1. Select the NAT which is created within your VPC
   2. Now go to "Actions" and select "Delete NAT gateway" now type "delete" in confirmation textbox to delete the NAT.

**NOTE:** Refresh the NAT gateways template **until**

status of NAT gateway ID must turn to "**Deleted**"

1. Now, **as NAT is auto assigned with Elastic IP address** by AWS which is also chargeable.
   1. **Go to** "Elastic IPs" **under** VPC dashboard refresh the template to avoid conflicts and **select** the

Elastic IP address within your VPC

* 1. Now **go to** "Actions" and **select** "Release Elastic IP addresses" & **click on** "Release" button.
  2. **VPC** **endpoints** – Leave this with auto selected option( i.e., S3 Gateway)
  3. **DNS Options** – Leave this with auto selected options(i.e., Enable DNS hostnames &

Enable DNS resolution)

* 1. **Click** on “Create VPC” - VPC is created with a **Unique ID**

**NOTE:**

**AWS creates a** **default VPC** to simplify networking for users, especially for beginners or those who don't need complex custom VPC setups.

When a new AWS account is created, AWS automatically sets up a default VPC in each region.

* Go to Your VPCs under VPC dashboard and then check under Default VPC yes refers the Default VPC auto created by AWS.

**How to know the number of subnets created under your VPC**

* + - VPC is created with a **Unique ID**
    - By copying and searching VPC’s **ID** at Subnet console under VPC dashboard , we get see our VPC subnets.

**To** **determine the subnet is Public or Private, check the subnet associated Route Table**

* + - Clickonany **subnet ID** and then go to Route table
      * If there is route to Internet Gateway(IGW) then it is a **Public Subnet**
      * If there is route to NAT Gateway(NAT) then it is a **Private Subnet**
      * Click on **NAT Gateway ID** to know **Public** and **Private IP** assigned by **AWS** to it.

**To know the auto associated Network ACL of subnet**

* + - Clickonany **subnet ID** and then go to Network ACL – default **allow** all inbound and outbound traffic to subnet

**To explicitly define Network ACL rule to a subnet**

* + - Clickonany **subnet ID** and then go to Network ACL
    - Click on Edit network ACL association
    - Click on Current network ACL – which takes us to Network ACL dashboard
    - Select **Network ACL’s ID**
    - Under Inboundrules, click on “Edit inbound rules”
    - Click onAdd new rule
      * Rule number– 101
      * Type- Custom TCP
      * Port range– 80
      * Source– 70.0.0.1
      * Allow/Deny– Deny

**To check Elastic IP address auto assigned to NAT**

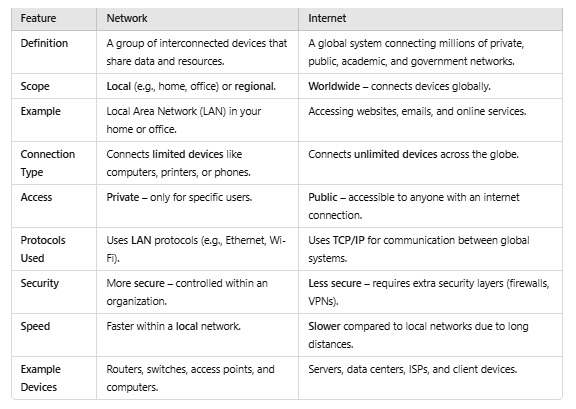
* + - Go to Elastic IPs under VPC dashboard
    - Click on Allocated IPv4 address of your VPC

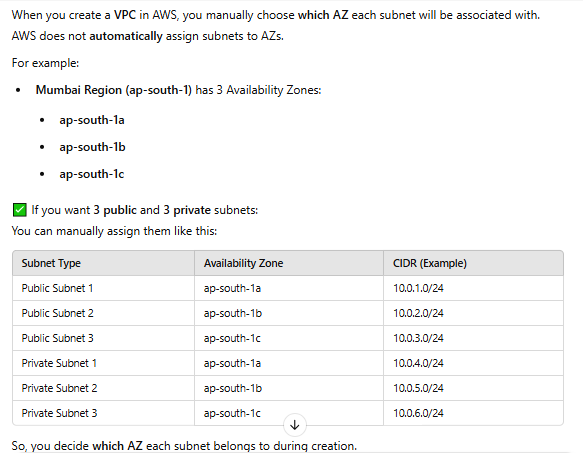
**To create another subnet under our VPC**

* + - Go to Subnet under VPC dashboard
    - **Click** on Create Subnet
      * **At** VPC ID – select under which VPC you want to create subnet
      * At Subnet 1 of 1
        1. Subnet name – give a name to subnet
        2. Availability Zone **–** let you can or amazon can decide
        3. IPv4 VPC CIDR block –
        4. IPv4 subnet CIDR block **–** choose size of subnet (say 10.0.1.0/2)
        5. Click on create subnet button
        6. Click on newly created **Subnet ID** and select Edit route table association

Under Route table ID – select the **route table ID** which you want to associate to new subnet.

Click Save then a route to IGW or NAT will be defined in new Subnet’s Route Table

****

****

