A **VPC (Virtual Private Cloud)** in AWS is a logically isolated network within the AWS cloud where you can launch and manage your resources, such as servers (EC2), databases, and other services.

Key Points:

- **Isolation:** A VPC provides a private, customizable network for your AWS resources, isolated from other users.
- **Customization:** You can define IP address ranges (using CIDR blocks), create subnets, set up route tables, and configure internet or VPN access.
- **Connectivity:** VPCs allow you to connect securely to the internet, other VPCs, or your on-premises data center.
- **Security:** You can control inbound and outbound traffic using **security groups** and **network access control lists (NACLs)**.

How to Build AWS VPC using Terraform – Step by Step

- Step 1: Create a VPC
- Step 2: Create Subnets
- Step 3: Set up Internet Gateway
- Step 4: Create a Route Table
- Step 5: Associate Public Subnets with the Second Route Table

Creating a VPC module usually involves at least the following components:

- **VPC(s):** A logically isolated network in AWS for launching cloud resources.
- **Subnet(s):** Smaller sections within a VPC to organize and isolate resources.
- **Internet Gateway(s):** A gateway that enables VPC resources to connect to the internet.
- **Route Tables:** Rules that control how traffic flows within a VPC and outside of it.
- **NAT Gateway(s) – Optional:** Allows private subnet resources to access the internet without being directly exposed.
- **Security Group(s):** Stateful firewalls that control inbound and outbound traffic to resources.
- **Network Access Control List(s) – Optional:** Stateless firewalls that control traffic at the subnet level.
- **Peering – Optional:** Connects two VPCs for secure communication without using the internet.

This is the minimum structure of a module:

- **main.tf**: Contains the core resource declarations and configurations for the module.
- **variables.tf**: Defines input variables that allow users to customize the module's behavior.
- **outputs.tf**: Provides information about the created resources.
- **providers.tf**: Defines the versions used for the providers and terraform
- **README.md**: Documentation on how to use the module, including descriptions of input variables and outputs.

After this execute all these terraform files using the below commands one by one .

- terraform init

- terraform fmt
- terraform validate
- terraform plan
- terraform apply -auto-approve

Now if you want to delete all the resources created through terraform, then write this command.

- terraform destroy

1. VPC (Virtual Private Cloud)

- **Purpose:** A VPC is like your own private network within AWS. It allows you to isolate and control your cloud resources (like servers, databases, etc.) as if they are running in a private data center.
- **Analogy:** Think of a VPC as a house where you can decide who gets access to each room, what furniture goes inside, and what the layout looks like.

2. Subnet(s)

- **Purpose:** Subnets divide your VPC into smaller sections. Each subnet can host specific resources (like EC2 instances).
- **Key Detail:** Subnets are either **public** (connected to the internet) or **private** (isolated, with no direct internet access).
- **Analogy:** Imagine a VPC (your house) with rooms (subnets). One room has a window to the outside (public subnet), while another is sealed off for privacy (private subnet).

3. Internet Gateway

- **Purpose:** The Internet Gateway enables resources inside your VPC (like EC2 instances) to access the internet and be accessed from the internet. It's attached to your VPC.
- **Key Detail:** Without an Internet Gateway, resources inside a VPC cannot communicate with the internet.
- **Analogy:** It's like the front door of your house—allowing you to go outside (access the internet) and letting others in (incoming traffic).

4. Route Table(s)

- **Purpose:** Route tables control how traffic moves within your VPC and outside of it. They contain rules (routes) that direct traffic to the correct destination (e.g., to an internet gateway or a NAT Gateway).
- **Key Detail:** Each subnet must be associated with a route table.
- **Analogy:** Think of route tables as road maps for your network traffic, telling data which "road" to take to reach its destination.

5. NAT Gateway (Optional)

- **Purpose:** NAT (Network Address Translation) Gateway allows resources in a **private subnet** to access the internet (e.g., to download updates) without being directly exposed to incoming traffic.
- **Key Detail:** It's used for outbound traffic only (no inbound internet access).
- **Analogy:** It's like a one-way window—resources inside the private room can look outside and access the internet, but no one outside can look in.

6. Security Group(s)

- **Purpose:** Security Groups act as firewalls that control inbound and outbound traffic to AWS resources, like EC2 instances. They are **stateful**, meaning they remember connections and automatically allow responses to inbound traffic.
- **Key Detail:** Security groups are tied to specific resources (like EC2).
- **Analogy:** Imagine security guards outside specific rooms (EC2 instances) deciding who can enter and leave based on a list of allowed visitors.

7. Network Access Control Lists (NACLs) – Optional

- **Purpose:** NACLs are additional firewalls that control traffic at the subnet level. Unlike Security Groups, NACLs are **stateless**, meaning they don't remember connections—you need to specify both inbound and outbound rules.
- **Key Detail:** NACLs act as a first layer of defense before traffic reaches a subnet.
- **Analogy:** Think of NACLs as the security gate at the neighborhood entrance that checks all cars (traffic) before they enter any house (subnet).

8. Peering (Optional)

- **Purpose:** VPC Peering connects two VPCs so that resources in one VPC can communicate with resources in another, as if they're on the same network.
- **Key Detail:** VPC Peering doesn't use the internet, so it's secure and fast.
- **Analogy:** Imagine you want to connect two houses in the same city with a private tunnel so that people (data) can move between them without going through public roads (the internet).