Author: Madhurima Rawat

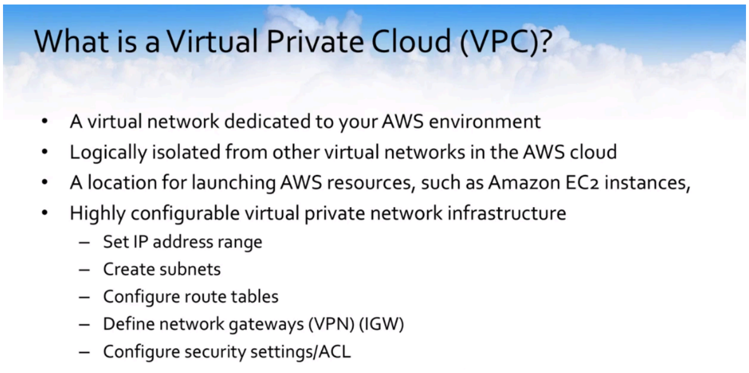
Setting Up and Configuring Cloud Networking

This experiment explores configuring and managing cloud networking services, including setting up Virtual Private Clouds (VPCs) and subnets. It utilizes AWS CLI and LocalStack to simulate cloud networking environments.

1. Overview of Virtual Private Cloud (VPC)

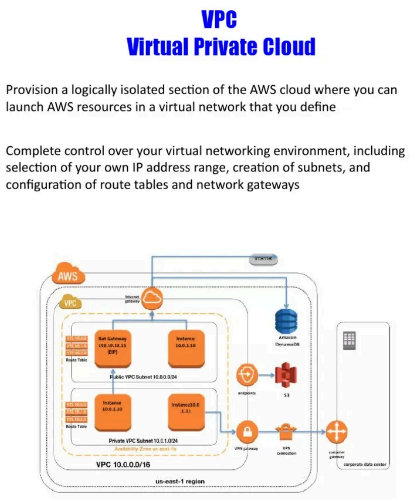
A **VPC** is a customizable virtual network within the AWS cloud. It allows you to manage networking resources securely and privately.

- Allows the creation of isolated networks in the cloud.
- Provides control over IP address ranges and routing.
- Enables creating public and private subnets.
- Facilitates secure, private communication between instances.



What is a Virtual Private Cloud (VPC)?

- A virtual network dedicated to your AWS environment
- Logically isolated from other virtual networks in the AWS cloud
- A location for launching AWS resources, such as Amazon EC2 instances,
- Highly configurable virtual private network infrastructure
  - Set IP address range
  - Create subnets
  - Configure route tables
  - Define network gateways (VPN) (IGW)
  - Configure security settings/ACL



VPC
Virtual Private Cloud

Provision a logically isolated section of the AWS cloud where you can launch AWS resources in a virtual network that you define

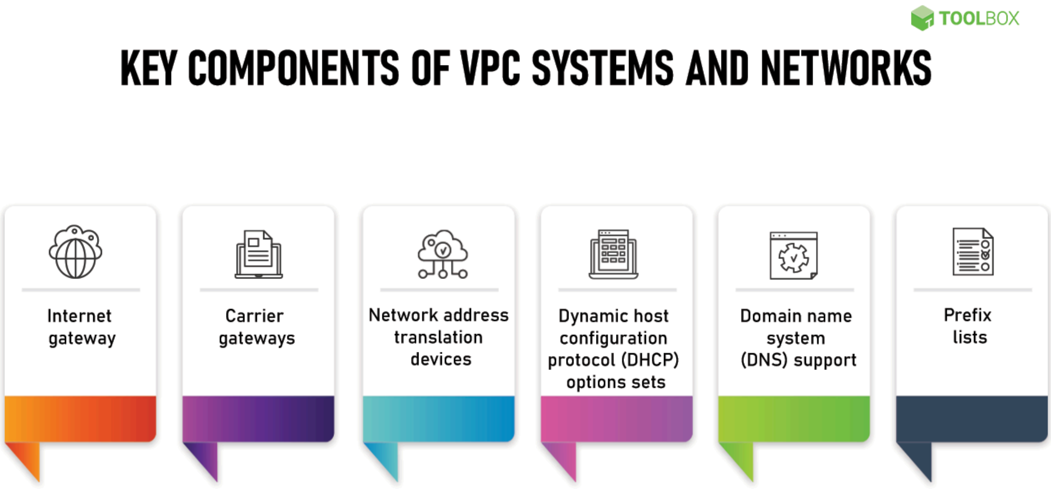
Complete control over your virtual networking environment, including selection of your own IP address range, creation of subnets, and configuration of route tables and network gateways

AWS: Virtual Private Cloud - VPC

2. Key Components of a VPC

The VPC is made up of several essential components for creating a robust networking environment:

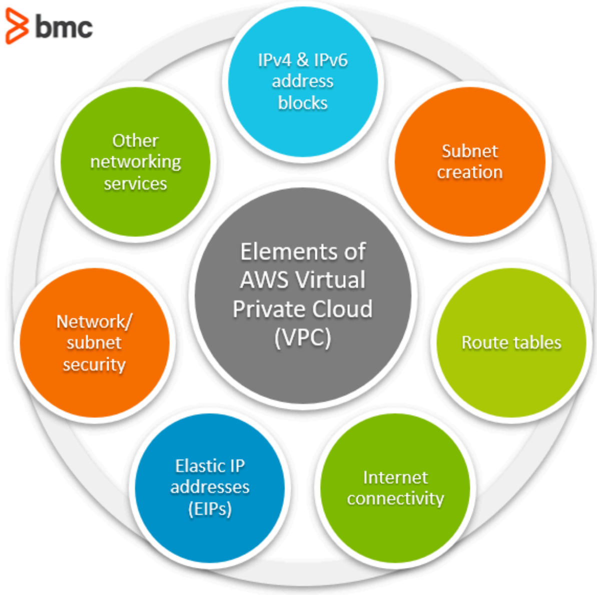
- **Subnets**: Segment the network into smaller, manageable sections.
- **Internet Gateway (IGW)**: Provides internet access for public-facing resources.
- **Route Tables**: Direct traffic within the VPC and between the internet.
- **NAT Gateway**: Allows private subnet instances to access the internet.
- **Security Groups & Network ACLs**: Define inbound and outbound traffic permissions for resources.


TOOLBOX
KEY COMPONENTS OF VPC SYSTEMS AND NETWORKS
Internet gateway
Carrier gateways
Network address translation devices
Dynamic host configuration protocol (DHCP) options sets
Domain name system (DNS) support
Prefix lists

3. Networking Flow and Communication

The VPC enables seamless communication between instances and secure internet access.

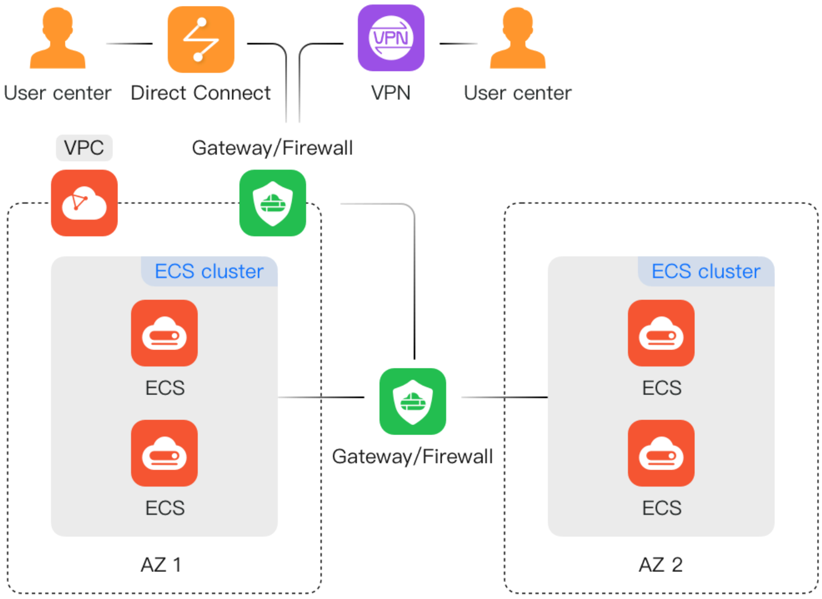
- **Public Subnets**: These subnets have internet access via an IGW, making them suitable for web servers.
- **Private Subnets**: Isolated subnets that don't directly connect to the internet; they access the internet via a NAT Gateway.
- **Route Tables**: Manage the routing of network traffic within the VPC, to/from the internet or between subnets.
- **Security Layers**: Using **Security Groups** and **Network ACLs** to filter traffic and secure resources.

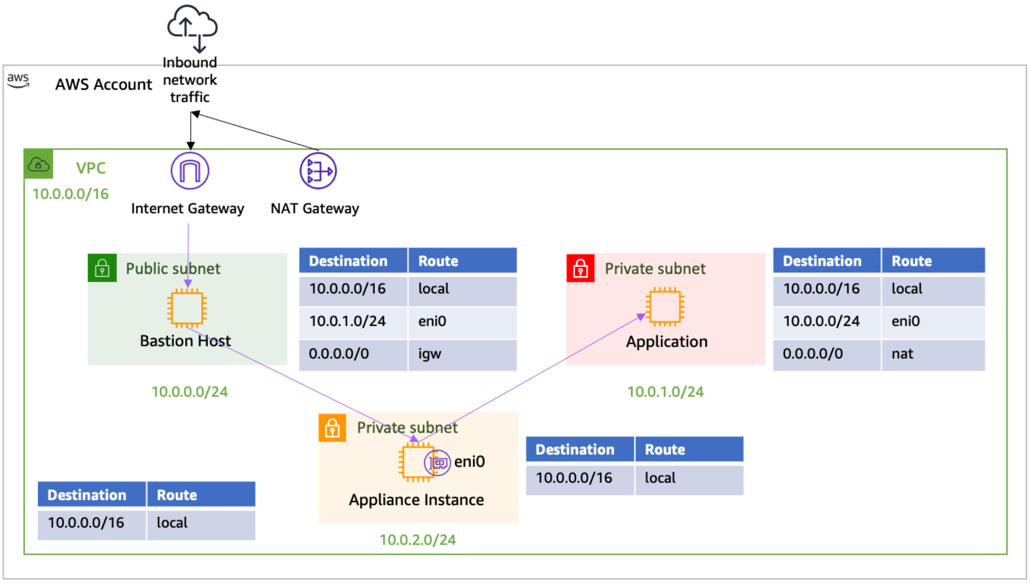

bmc
Elements of AWS Virtual Private Cloud (VPC)
IPv4 & IPv6 address blocks
Subnet creation
Route tables
Internet connectivity
Elastic IP addresses (EIPs)
Network/ subnet security
Other networking services

4. LocalStack for VPC Simulation

LocalStack simulates AWS services locally, allowing you to test cloud configurations without needing actual AWS resources.

- **Simulates AWS Cloud Services**: Enables local testing for cloud networking and services like VPC.
- **Supports Full AWS Environment**: Includes EC2, S3, VPC, and more.
- **Custom Endpoints**: Interact with LocalStack through custom endpoint URLs, mimicking real AWS behavior.
- **Ideal for Development and Testing**: Provides a cost-effective solution for developers to test network setups.





Destination	Route
10.0.0.0/16	local
10.0.1.0/24	eni0
0.0.0.0/0	igw

Destination	Route
10.0.0.0/16	local
10.0.0.0/24	eni0
0.0.0.0/0	nat

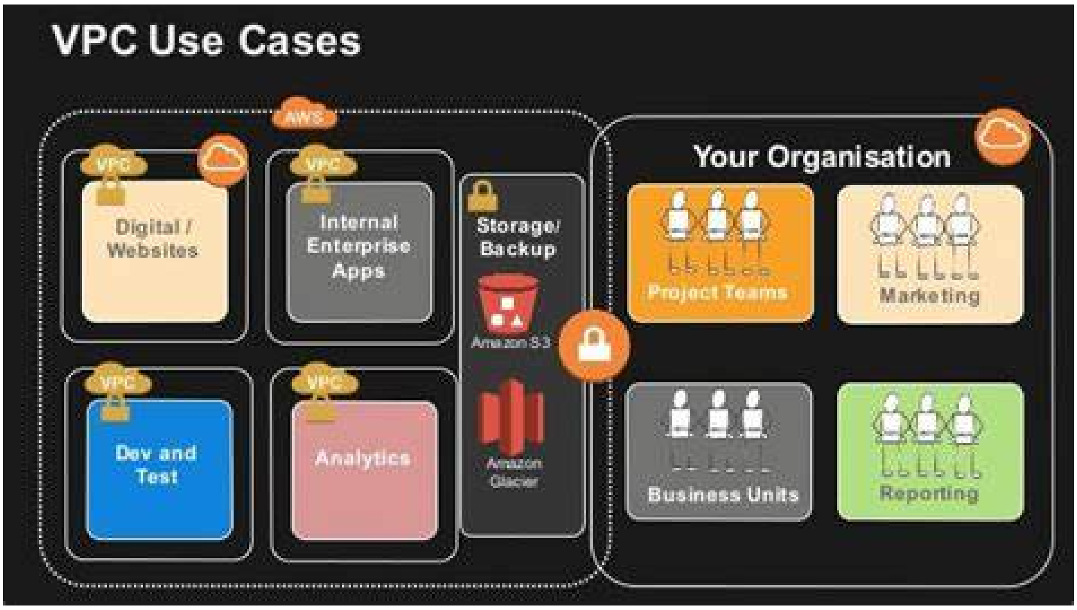
Destination	Route
10.0.0.0/16	local

Destination	Route
10.0.0.0/16	local

5. Use Cases of VPCs

VPCs can be used for various networking purposes to create secure, scalable architectures.

- **Web Hosting**: Deploy secure web applications with public-facing and private backends.
- **Hybrid Cloud**: Extend on-premise networks into the cloud via VPN or Direct Connect.
- **Big Data & Machine Learning**: Isolate workloads for processing and analysis.
- **Multi-Tier Architectures**: Separate frontend, application, and database tiers for better security.



🖥️ All Components of a VPC Implemented in LocalStack

LocalStack is a **local AWS cloud emulator** that allows developers to create and test AWS services, including **Virtual Private Cloud (VPC)**, on their local machines.

🌐 1. Virtual Private Cloud (VPC)

- **Definition**: A logically isolated network where cloud resources like EC2 instances, databases, and load balancers are deployed.
- **Key Attributes**:
  - **VPC ID**: Unique identifier (e.g., `vpc-12345678`).
  - **CIDR Block**: Defines the **IP range** for the VPC (e.g., `10.0.0.0/16`).
  - **Default vs. Custom VPC**:
    - **Default VPC**: Automatically provided by AWS.
    - **Custom VPC**: User-defined network configuration.

🛠️ CIDR (Classless Inter-Domain Routing) & CIDR Block Explained

CIDR stands for **Classless Inter-Domain Routing**, a method used to allocate IP addresses efficiently and manage network traffic. It replaces the **traditional class-based addressing** (Class A, B, C) with a more flexible **prefix-based system**.

📌 What is a CIDR Block?

A **CIDR block** defines a range of IP addresses that can be used within a network, such as a **VPC or Subnet**. It consists of:

1. **Base IP Address** → The starting point of the network (e.g., `192.168.1.0`).
2. **Subnet Mask (Slash Notation `/X`)** → Determines how many IPs are in the range.

🔢 How CIDR Works (Subnet Mask & IP Ranges)

- CIDR notation looks like:

  ```
  192.168.1.0/24
  ```

  - `192.168.1.0` → **Base IP Address**
  - `/24` → **Subnet Mask** (Means first 24 bits are fixed for network, remaining bits for hosts)
- **IP Address Calculation:**
  - `/24` allows **256 total IPs** (`2^(32-24) = 256`), but **usable IPs are 254** because:
    - 1st IP (`192.168.1.0`) → **Network Address**
    - Last IP (`192.168.1.255`) → **Broadcast Address**

📊 Common CIDR Blocks & Their Usages

CIDR Block	Subnet Mask	Total IPs	Usable IPs	Example Usage
`10.0.0.0/8`	255.0.0.0	16,777,216	16,777,214	Large private networks (Enterprise)
`192.168.1.0/24`	255.255.255.0	256	254	Small office/home network
`172.16.0.0/16`	255.255.0.0	65,536	65,534	Medium-sized private networks
`10.0.1.0/28`	255.255.255.240	16	14	Small subnet for internal services

📌 Real-World Example

🏢 Company Network Setup in LocalStack

- A company wants **two separate environments**:
  i. **Public Subnet** for web servers → `192.168.1.0/24`
  ii. **Private Subnet** for databases → `192.168.2.0/24`
- The VPC is created in **LocalStack**, ensuring a controlled environment where services can be tested **without real AWS costs**.

🌟 Summary

- **CIDR Block** defines **IP ranges** within a network (VPC, Subnet).
- The **suffix (`/X`) determines subnet size**, affecting the **number of IPs**.
- **Real-world usage**: Efficiently dividing networks for **different applications (web, database, internal services, etc.)**.
- 📌 **Real-World Example**:
  - A company needs a **private network** for its applications.
  - The **VPC is created in LocalStack** with `192.168.1.0/24` as the IP range, ensuring complete **control over networking and security**.

📌 2. Subnet (Public & Private)

- **Definition**: A segment of the **VPC that organizes resources** and determines how IPs are assigned.
- **Key Attributes**:
  - **Subnet ID**: Unique identifier (e.g., `subnet-98765432`).

  - **CIDR Block**: The subnet's IP range (e.g., `10.0.1.0/24`).
  - **Types of Subnets**:
    - 🌍 **Public Subnet**: Connected to the **Internet Gateway** for external access.
    - 🔒 **Private Subnet**: No direct internet access, used for internal applications.
  - **Availability Zone (AZ)**: Each subnet belongs to a specific AWS **Availability Zone** (e.g., `us-east-1a`).
- 📌 **Real-World Example**:
  - A company **hosts a web application** with two components:
    - **Public Subnet (10.0.1.0/24)** → Hosts a **web server** that users can access.
    - **Private Subnet (10.0.2.0/24)** → Stores a **database** that should not be directly exposed to the internet.

🛣️ 3. Route Table

- **Definition**: A **set of rules** (routes) that direct traffic within the VPC.
- **Key Attributes**:
  - **Route Table ID**: Unique identifier (e.g., `rtb-56789`).
  - **Routes**: Define paths for network traffic (e.g., `0.0.0.0/0 → Internet Gateway`).
  - **Subnet Association**: Subnets are linked to a specific route table.
- 📌 **Real-World Example**:
  - The **public subnet** is associated with a route table that contains:
    - `0.0.0.0/0 → Internet Gateway (igw-12345)`, allowing internet access.
  - The **private subnet** is associated with a route table that contains:
    - `10.0.0.0/16 → Local Traffic`, ensuring internal communication only.

🛜 4. Internet Gateway (IGW) & NAT Gateway

- 🌍 **Internet Gateway (IGW)**:
  - Allows resources in the **public subnet** to connect to the internet.
  - **Real-World Example**: A web server in a public subnet needs to **fetch updates from the internet**.
- 🔒 **NAT Gateway** (for private subnets):
  - Enables **outgoing internet traffic** from private resources without exposing them to inbound requests.
  - **Real-World Example**: A backend server in a **private subnet** needs to **download software updates**, but should not be directly accessible.

⚡ 5. Security Group (SG) & Network ACL (NACL)

- 🔒 **Security Group (SG)**:
  - Acts as a **firewall** that controls inbound/outbound traffic **at the instance level**.
  - **Real-World Example**: A **web server** has an **SG allowing inbound traffic on port 80 (HTTP)** but blocks everything else.
- 📜 **Network ACL (NACL)**:
  - Controls **traffic at the subnet level** and applies **rules in order**.
  - **Real-World Example**: A company sets a **rule to block all traffic** from a **specific IP range** to **prevent malicious access**.

🔗 6. Elastic IP (EIP) & Private IP

- 📍 **Elastic IP (EIP)**:
  - A **static public IP** assigned to an instance, ensuring it remains the same even after reboot.
  - **Real-World Example**: A company assigns an **EIP to a critical web server** to maintain a **consistent IP for users**.
- 🏠 **Private IP**:
  - Assigned **inside a VPC** for internal communication.
  - **Real-World Example**: A database **only allows connections** from a specific **private IP** assigned to an application server.

🎯 7. VPC Peering & VPN Connection

- 🤝 **VPC Peering**:
  - Connects **two VPCs privately** without needing an internet connection.
  - **Real-World Example**: A company **connects its production and development environments** in different VPCs for seamless data sharing.
- 🛡️ **VPN Connection**:
  - Connects **on-premises networks to AWS** securely.
  - **Real-World Example**: A company uses a **VPN to access its AWS resources securely** from its office.

🛠️ 8. Load Balancer & Auto Scaling

- ⚖️ **Load Balancer (LB)**:
  - Distributes traffic across multiple instances to **improve availability**.
  - **Real-World Example**: A **web application with high traffic** uses an **Elastic Load Balancer (ELB)** to route requests efficiently.
- 📈 **Auto Scaling**:
  - **Dynamically adjusts** the number of instances based on demand.

    - **Real-World Example**: An **e-commerce website increases server capacity** automatically during sales events.

✅ Conclusion

LocalStack helps developers **simulate and test AWS VPC components** locally, enabling them to configure networking setups **without using actual AWS resources**. Understanding **CIDR blocks, subnets, route tables, gateways, security groups, and load balancers** ensures efficient **network architecture design** for cloud applications. 🚀

Step-by-Step Guide: Setting Up a VPC in LocalStack

Step 1: Start LocalStack

Start LocalStack using one of the following methods:

Using the CLI

```
localstack start
```

Using Docker

```
docker run --rm -it -p 4566:4566 localstack/localstack
```

Ensure Docker is Running

- Open **Docker Desktop** and wait until it displays **"Docker is running."**
- LocalStack will simulate AWS services on **port 4566**, allowing local development without needing an actual AWS account.

Step 2: Create a Virtual Private Cloud (VPC)

Create a **VPC (Virtual Private Cloud)** to define a private network:

```
aws ec2 create-vpc --cidr-block 10.0.0.0/16
--endpoint-url=%AWS_ENDPOINT_URL%
```

What This Does:

- Defines a **private network (10.0.0.0/16)** for AWS resources.
- Returns a **VPC ID** (e.g., `"VpcId": "vpc-123456"`), which is required for the next steps.

Step 3: Create a Subnet

A **subnet** allows instances to communicate within the VPC.

```
aws ec2 create-subnet --vpc-id vpc-123456
--cidr-block 10.0.1.0/24 --endpoint-url=%AWS_ENDPOINT_URL%
```

What This Does:

- Creates a **smaller network (10.0.1.0/24)** within the VPC for organizing resources.
- A **Subnet ID** (e.g., `"SubnetId": "subnet-123456"`) is returned, which is needed for later steps.

Step 4: Create and Attach an Internet Gateway

An **Internet Gateway (IGW)** allows resources in the subnet to access the internet.

Create the Internet Gateway:

```
aws ec2 create-internet-gateway --endpoint-url=%AWS_ENDPOINT_URL%
```

Attach the Internet Gateway to the VPC:

```
aws ec2 attach-internet-gateway --internet-gateway-id
igw-123456 --vpc-id vpc-123456 --endpoint-url=%AWS_ENDPOINT_URL%
```

What This Does:

- Enables instances in the VPC to communicate with the internet.
- Returns an **Internet Gateway ID** (e.g., `"InternetGatewayId": "igw-123456"`).

Step 5: Configure Routing for Internet Access

To allow traffic to and from the internet, a **Route Table** must be created and updated.

Create a Route Table for the VPC:

```
aws ec2 create-route-table --vpc-id vpc-123456
--endpoint-url=%AWS_ENDPOINT_URL%
```

Add a Default Route to Enable Internet Access:

```
aws ec2 create-route --route-table-id rtb-123456
--destination-cidr-block 0.0.0.0/0 --gateway-id
igw-123456 --endpoint-url=%AWS_ENDPOINT_URL%
```

Associate the Route Table with the Subnet:

```
aws ec2 associate-route-table --route-table-id rtb-123456
--subnet-id subnet-123456 --endpoint-url=%AWS_ENDPOINT_URL%
```

What This Does:

- Defines a **default route (** `0.0.0.0/0` **)**, allowing traffic to flow to the internet.
- Links the route table to the **subnet** so instances can use the IGW.

Step 6: Verify the Configuration

List All VPCs:

```
aws ec2 describe-vpcs --endpoint-url=%AWS_ENDPOINT_URL%
```

List All Subnets:

```
aws ec2 describe-subnets --endpoint-url=%AWS_ENDPOINT_URL%
```

What This Does:

- Confirms the VPC and subnets have been successfully created and configured.
- Ensures all components (VPC, subnet, IGW, and routing) are correctly set up.

Useful Resources for Learning about Virtual Private Cloud (VPC)

1. **Spiceworks - What is Virtual Private Cloud?**
   A comprehensive guide explaining the concept of Virtual Private Cloud (VPC), its benefits, and how it enhances network security and control over cloud resources.
2. **BMC - AWS VPC: Virtual Private Cloud**
   This article provides an in-depth overview of AWS VPC, covering key features, use cases, and how to set up and manage VPCs effectively on AWS.
3. **Medium - Deep Dive into Amazon Virtual Private Cloud (VPC)**
   A detailed exploration of the core features of Amazon VPC, focusing on security, scalability, and flexibility in network configurations.
4. **Medium - Understanding AWS VPC: A Comprehensive Guide from Basics to Best Practices**
   This guide offers an extensive look at AWS VPC, from its basic concepts to best practices for optimal usage in various scenarios, including security and architecture considerations.
5. **Medium - AWS VPC: Virtual Private Cloud**
   A beginner-friendly article that explains the fundamental concepts of AWS VPC, its components, and how to leverage VPC for secure cloud networking.
6. **Dev.to - A Beginner's Guide to AWS Virtual Private Cloud (VPC) Security**
   This beginner-friendly guide introduces AWS VPC and focuses on its security features. It explains the importance of securing VPCs in cloud environments and provides practical tips for setting up secure VPC architectures on AWS.