Amazon Virtual Private Cloud (Amazon VPC)

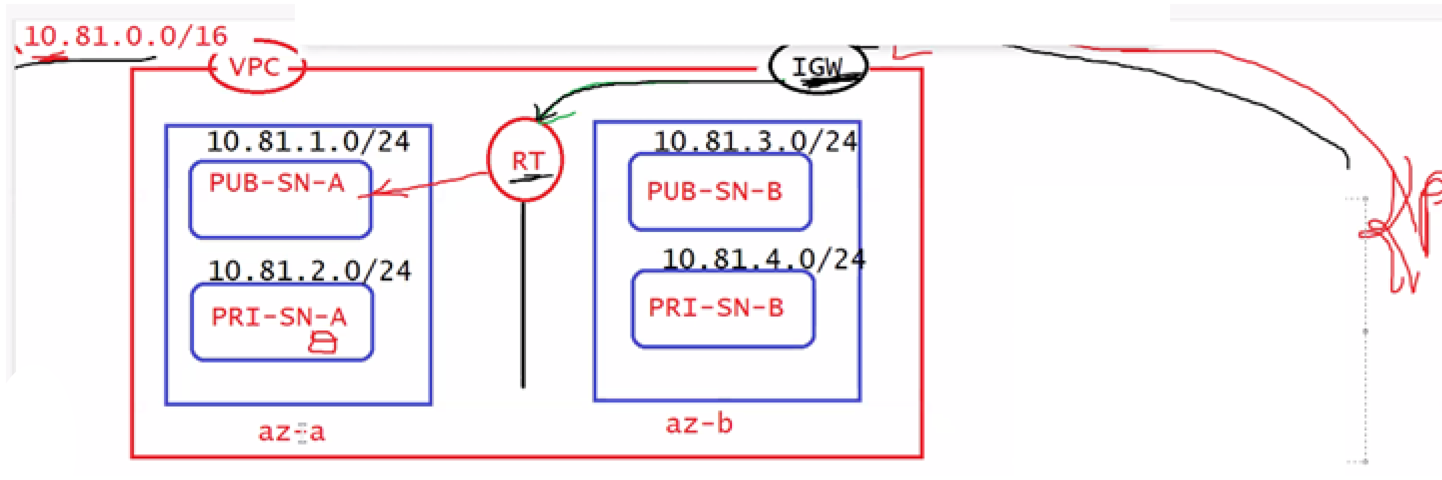
Amazon Virtual Private Cloud (Amazon VPC) is a web service provided by Amazon Web Services (AWS) that allows you to create a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network. You have complete control over your virtual networking environment, including selection of your IP address range, creation of subnets, and configuration of route tables and network gateways.

Key features and components of Amazon VPC include:

1. **Subnets:** A VPC can be divided into multiple subnets, each associated with a specific Availability Zone (AZ). Subnets are used to organize and segment your resources within the VPC.
2. **Route Tables:** A route table contains a set of rules, called routes, that are used to determine where network traffic is directed. Each subnet in a VPC must be associated with a route table, which controls the traffic within the subnet.
3. **Internet Gateway:** An Internet Gateway (IGW) enables communication between instances in a VPC and the Internet. It allows instances with public IP addresses to directly access the Internet and vice versa.
4. **NAT Gateway/NAT Instance:** Network Address Translation (NAT) allows instances in a private subnet to initiate outbound traffic to the Internet while preventing inbound traffic from reaching them. This can be achieved using a NAT gateway or a NAT instance.
5. **Security Groups and Network Access Control Lists (ACLs):** Security groups act as a virtual firewall for your instances to control inbound and outbound traffic. Network ACLs are an optional layer of security that acts at the subnet level, controlling traffic in and out of the subnet.
6. **VPC Peering:** VPC peering allows you to connect one VPC with another over a direct network route, using private IP addresses.
7. **Virtual Private Network (VPN) and Direct Connect:** These options provide secure connections between your on-premises data centers and your VPC, allowing you to extend your network into the cloud.
8. **Elastic Load Balancer (ELB):** ELB distributes incoming application traffic across multiple targets, such as Amazon EC2 instances, in multiple availability zones.
9. **Flow Logs:** Flow logs capture information about IP traffic going to and from network interfaces in your VPC. You can use flow logs for various purposes, such as troubleshooting, monitoring, and security analysis.

10. **VPC Endpoints:** VPC endpoints enable you to privately connect your VPC to supported AWS services without requiring an Internet gateway, NAT device, VPN connection, or Direct Connect connection.

When you create a VPC, you define its IP address range (CIDR block), and you can choose to enable DNS hostnames, DNS resolution, and support for Classic Link, among other options. AWS provides a highly flexible and customizable network environment, allowing you to design and configure your VPC to meet the specific needs of your applications and workloads.



Configuring VPC

1. **Sign in to the AWS Management Console:** Log in to your AWS account at https://aws.amazon.com/.

Step 1: Create a VPC

1. Sign in to the AWS Management Console.
2. Go to the "Services" menu and select "VPC" under "Networking & Content Delivery."
3. In the VPC Dashboard, click on "Your VPCs" in the left navigation pane.
4. Click the "Create VPC" button.
5. Fill in the following details:
    - **Name tag:** Give your VPC a name (e.g., MyVPC).
    - **IPv4 CIDR block:** Enter the IP address range for your VPC in CIDR notation (e.g., 10.0.0.0/16).
6. Click "Create VPC."

Step 2: Create a Subnet

1. In the VPC Dashboard, click on "Subnets" in the left navigation pane.
2. Click the "Create subnet" button.
3. Fill in the following details:
    - **Name tag:** Give your subnet a name.
    - **VPC:** Choose the VPC you created in the previous step.
    - **Availability Zone:** Choose an availability zone.
    - **IPv4 CIDR block:** Enter a subnet IP address range within the VPC CIDR block (e.g., 10.0.0.0/24).
4. Click "Create subnet."

Step 3: Create a Route Table

1. In the VPC Dashboard, click on "Route Tables" in the left navigation pane.
2. Click the "Create route table" button.
3. Fill in the following details:
    - **Name tag:** Give your route table a name.
    - **VPC:** Choose the VPC you created in the first step.
4. Click "Create route table."

Step 4: Associate Subnet with Route Table

1. In the "Route Tables" section, select the route table you just created.
2. In the "Subnet associations" tab, click "Edit subnet associations."
3. Associate the subnet you created in Step 2 with the route table.
4. Click "Save associations."

Step 5: Create an Internet Gateway

1. In the VPC Dashboard, click on "Internet Gateways" in the left navigation pane.
2. Click the "Create internet gateway" button.

3. Give your internet gateway a name and click "Create internet gateway."
4. Select the internet gateway you created and click "Attach to VPC."
5. Choose the VPC you created in the first step and click "Attach internet gateway."

Step 6: Update Route Table for Internet Access

1. In the "Route Tables" section, select the route table associated with your subnet.
2. Click on the "Routes" tab and then click "Edit routes."
3. Add a route with destination **0.0.0.0/0** and target as the internet gateway you created.
4. Click "Save routes."

Now, your VPC is set up with a subnet, a route table, and an internet gateway, allowing instances in your subnet to access the internet.

In AWS, private and public subnets are concepts related to network architecture within a Virtual Private Cloud (VPC). These subnets are used to control access to resources and enhance the security of your applications.

Public Subnet:

A public subnet is a subnet that has a route to the Internet through an Internet Gateway (IGW). Instances launched in a public subnet can communicate directly with the Internet, and they can also receive inbound traffic initiated from the Internet.

Create a Public Subnet:

- When creating a subnet, ensure that you associate it with the VPC you created.
- Configure the subnet with a route table that has a route to the Internet Gateway (**0.0.0.0/0** pointing to the Internet Gateway).

Step 1: Launch a Public Instance in a Public Subnet:

1. In the AWS Management Console, navigate to "Services" > "EC2."
2. Click on "Instances" in the left navigation pane.
3. Click the "Launch Instance" button.
4. Choose an Amazon Machine Image (AMI) and select an instance type.
5. In the "Configure Instance Details" step:

- Select the VPC you created.
- Choose the public subnet from the "Subnet" dropdown.

6. In the "Add Storage" and "Add Tags" steps, configure as needed.
7. In the "Configure Security Group" step:
   - Create a new security group (or use an existing one) for the public instance.
   - Allow inbound traffic on port 22 (SSH) and any other ports your application requires (e.g., port 80 for HTTP).
8. In the "Configure Key Pair" step:
   - Select the key pair you created.
9. Review the configuration and click "Launch."

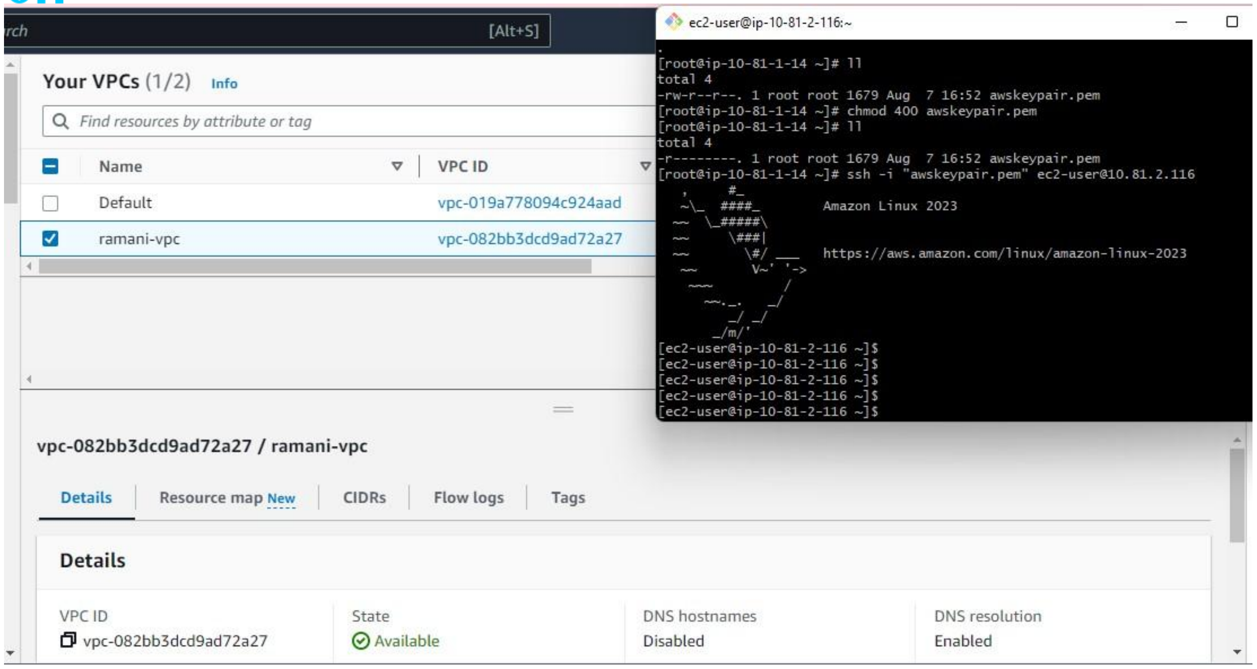
10.Now you can access the public server

Private Subnet:

A private subnet is a subnet that does not have a direct route to the Internet. Instances in a private subnet can communicate with other instances within the VPC and with resources on the Internet, but inbound traffic initiated from the Internet is not allowed.

Create a Private Subnet:

- When creating a subnet, ensure that you associate it with the VPC you created.
- Configure the subnet with a route table that does not have a route to the Internet Gateway
- You don't need to create or attach an Internet Gateway to a private subnet.
- Now to access the private instances in public instances copy and paste the sshkey with gienkey.pem name in public instances
- Now give command chmod 400 givenkey.pem
- Now copy theprivate instances ssh command ssh -i "givenkey.pem" ec2-user@private-instance-private-ip
- Now you can access the private server also



Your VPCs (1/2) Info

Name	VPC ID
Default	vpc-019a778094c924aad
ramani-vpc	vpc-082bb3dcd9ad72a27

vpc-082bb3dcd9ad72a27 / ramani-vpc

Details

VPC ID	State	DNS hostnames	DNS resolution
vpc-082bb3dcd9ad72a27	Available	Disabled	Enabled

ec2-user@ip-10-81-2-116:~

```
.
[root@ip-10-81-1-14 ~]# ll
total 4
-rw-r--r--. 1 root root 1679 Aug  7 16:52 awskeypair.pem
[root@ip-10-81-1-14 ~]# chmod 400 awskeypair.pem
[root@ip-10-81-1-14 ~]# ll
total 4
-r--------. 1 root root 1679 Aug  7 16:52 awskeypair.pem
[root@ip-10-81-1-14 ~]# ssh -i "awskeypair.pem" ec2-user@10.81.2.116
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-10-81-2-116 ~]$
[ec2-user@ip-10-81-2-116 ~]$
[ec2-user@ip-10-81-2-116 ~]$
[ec2-user@ip-10-81-2-116 ~]$
[ec2-user@ip-10-81-2-116 ~]$
```

Amazon VPC Peering

Amazon VPC Peering is a networking connection between two Virtual Private Clouds (VPCs) in the same or different AWS regions. VPC peering allows you to connect VPCs and share resources across them securely using private IP addresses. This enables you to build complex multi-tier applications or move workloads between VPCs.

Here are the key points about VPC peering:

Prerequisites:

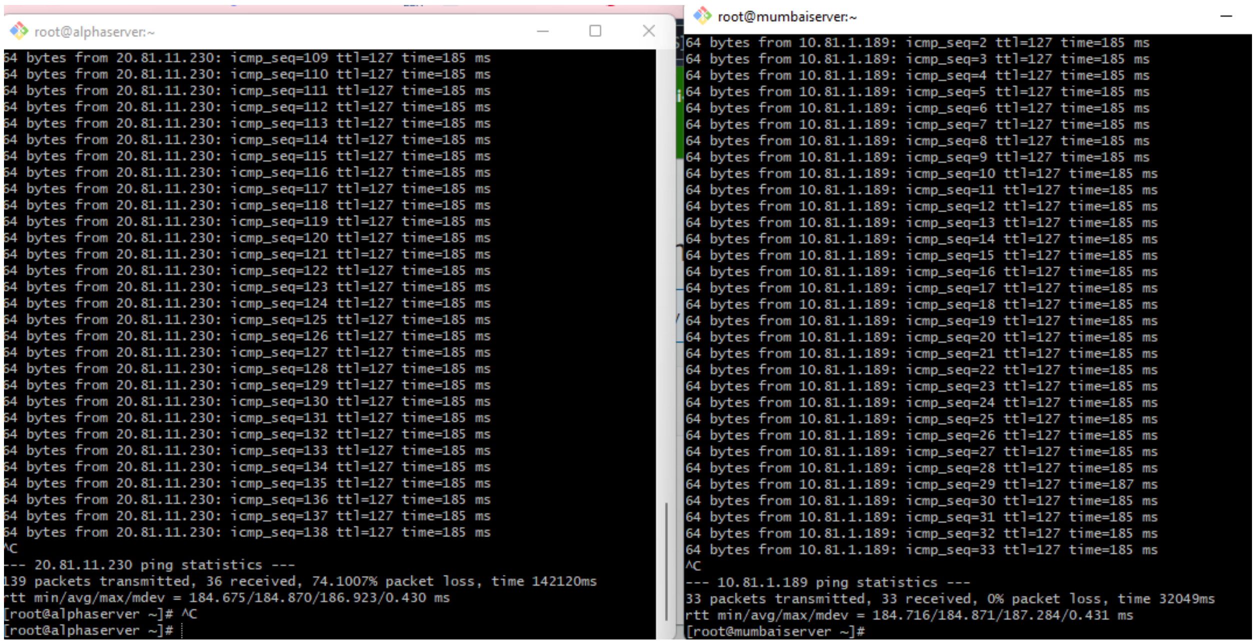
1. **VPCs in the Same or Different Accounts:**
    - VPC peering can be established between VPCs in the same AWS account or different AWS accounts.
2. **Non-overlapping IP Address Ranges:**
    - The IP address ranges of the peered VPCs must not overlap.

Steps to Create VPC Peering:

1. **Navigate to VPC Dashboard:**
    - In the AWS Management Console, go to "Services" > "VPC."
2. **Select "Peering Connections":**

- In the left navigation pane, click on "Peering Connections."

3. **Create Peering Connection:**
   - Click the "Create Peering Connection" button.
4. **Configure Peering Connection:**
   - Specify a unique name for the peering connection.
   - Select the VPC with which you want to peer.
   - Repeat the process in the other VPC's console to request the peering connection.
5. **Accept Peering Connection:**
   - After the peering connection is created, the owner of the other VPC needs to accept the peering connection request.



```
64 bytes from 20.81.11.230: icmp_seq=109 ttl=127 time=185 ms
64 bytes from 20.81.11.230: icmp_seq=110 ttl=127 time=185 ms
64 bytes from 20.81.11.230: icmp_seq=111 ttl=127 time=185 ms
64 bytes from 20.81.11.230: icmp_seq=112 ttl=127 time=185 ms
64 bytes from 20.81.11.230: icmp_seq=113 ttl=127 time=185 ms
64 bytes from 20.81.11.230: icmp_seq=114 ttl=127 time=185 ms
64 bytes from 20.81.11.230: icmp_seq=115 ttl=127 time=185 ms
64 bytes from 20.81.11.230: icmp_seq=116 ttl=127 time=185 ms
64 bytes from 20.81.11.230: icmp_seq=117 ttl=127 time=185 ms
64 bytes from 20.81.11.230: icmp_seq=118 ttl=127 time=185 ms
64 bytes from 20.81.11.230: icmp_seq=119 ttl=127 time=185 ms
64 bytes from 20.81.11.230: icmp_seq=120 ttl=127 time=185 ms
64 bytes from 20.81.11.230: icmp_seq=121 ttl=127 time=185 ms
64 bytes from 20.81.11.230: icmp_seq=122 ttl=127 time=185 ms
64 bytes from 20.81.11.230: icmp_seq=123 ttl=127 time=185 ms
64 bytes from 20.81.11.230: icmp_seq=124 ttl=127 time=185 ms
64 bytes from 20.81.11.230: icmp_seq=125 ttl=127 time=185 ms
64 bytes from 20.81.11.230: icmp_seq=126 ttl=127 time=185 ms
64 bytes from 20.81.11.230: icmp_seq=127 ttl=127 time=185 ms
64 bytes from 20.81.11.230: icmp_seq=128 ttl=127 time=185 ms
64 bytes from 20.81.11.230: icmp_seq=129 ttl=127 time=185 ms
64 bytes from 20.81.11.230: icmp_seq=130 ttl=127 time=185 ms
64 bytes from 20.81.11.230: icmp_seq=131 ttl=127 time=185 ms
64 bytes from 20.81.11.230: icmp_seq=132 ttl=127 time=185 ms
64 bytes from 20.81.11.230: icmp_seq=133 ttl=127 time=185 ms
64 bytes from 20.81.11.230: icmp_seq=134 ttl=127 time=185 ms
64 bytes from 20.81.11.230: icmp_seq=135 ttl=127 time=185 ms
64 bytes from 20.81.11.230: icmp_seq=136 ttl=127 time=185 ms
64 bytes from 20.81.11.230: icmp_seq=137 ttl=127 time=185 ms
64 bytes from 20.81.11.230: icmp_seq=138 ttl=127 time=185 ms
^C
--- 20.81.11.230 ping statistics ---
139 packets transmitted, 36 received, 74.1007% packet loss, time 142120ms
rtt min/avg/max/mdev = 184.675/184.870/186.923/0.430 ms
[root@alphaserver ~]# ^C
[root@alphaserver ~]#
```

```
64 bytes from 10.81.1.189: icmp_seq=2 ttl=127 time=185 ms
64 bytes from 10.81.1.189: icmp_seq=3 ttl=127 time=185 ms
64 bytes from 10.81.1.189: icmp_seq=4 ttl=127 time=185 ms
64 bytes from 10.81.1.189: icmp_seq=5 ttl=127 time=185 ms
64 bytes from 10.81.1.189: icmp_seq=6 ttl=127 time=185 ms
64 bytes from 10.81.1.189: icmp_seq=7 ttl=127 time=185 ms
64 bytes from 10.81.1.189: icmp_seq=8 ttl=127 time=185 ms
64 bytes from 10.81.1.189: icmp_seq=9 ttl=127 time=185 ms
64 bytes from 10.81.1.189: icmp_seq=10 ttl=127 time=185 ms
64 bytes from 10.81.1.189: icmp_seq=11 ttl=127 time=185 ms
64 bytes from 10.81.1.189: icmp_seq=12 ttl=127 time=185 ms
64 bytes from 10.81.1.189: icmp_seq=13 ttl=127 time=185 ms
64 bytes from 10.81.1.189: icmp_seq=14 ttl=127 time=185 ms
64 bytes from 10.81.1.189: icmp_seq=15 ttl=127 time=185 ms
64 bytes from 10.81.1.189: icmp_seq=16 ttl=127 time=185 ms
64 bytes from 10.81.1.189: icmp_seq=17 ttl=127 time=185 ms
64 bytes from 10.81.1.189: icmp_seq=18 ttl=127 time=185 ms
64 bytes from 10.81.1.189: icmp_seq=19 ttl=127 time=185 ms
64 bytes from 10.81.1.189: icmp_seq=20 ttl=127 time=185 ms
64 bytes from 10.81.1.189: icmp_seq=21 ttl=127 time=185 ms
64 bytes from 10.81.1.189: icmp_seq=22 ttl=127 time=185 ms
64 bytes from 10.81.1.189: icmp_seq=23 ttl=127 time=185 ms
64 bytes from 10.81.1.189: icmp_seq=24 ttl=127 time=185 ms
64 bytes from 10.81.1.189: icmp_seq=25 ttl=127 time=185 ms
64 bytes from 10.81.1.189: icmp_seq=26 ttl=127 time=185 ms
64 bytes from 10.81.1.189: icmp_seq=27 ttl=127 time=185 ms
64 bytes from 10.81.1.189: icmp_seq=28 ttl=127 time=185 ms
64 bytes from 10.81.1.189: icmp_seq=29 ttl=127 time=187 ms
64 bytes from 10.81.1.189: icmp_seq=30 ttl=127 time=185 ms
64 bytes from 10.81.1.189: icmp_seq=31 ttl=127 time=185 ms
64 bytes from 10.81.1.189: icmp_seq=32 ttl=127 time=185 ms
64 bytes from 10.81.1.189: icmp_seq=33 ttl=127 time=185 ms
^C
--- 10.81.1.189 ping statistics ---
33 packets transmitted, 33 received, 0% packet loss, time 32049ms
rtt min/avg/max/mdev = 184.716/184.871/187.284/0.431 ms
[root@mumbaiserver ~]#
```

Key Considerations:

1. **Routing:**
   - VPC peering allows private IP communication between instances in the peered VPCs. Ensure that route tables are configured to allow traffic between the VPCs.
2. **Security Groups and Network ACLs:**
   - Adjust security groups and network ACLs to permit the necessary traffic between peered VPCs.

3. **Transitive Peering:**
    - VPC peering is not transitive. If VPC A is peered with VPC B and VPC B is peered with VPC C, VPC A and VPC C are not automatically peered.
4. **CIDR Blocks:**
    - Ensure that the CIDR blocks of the peered VPCs do not overlap.

Benefits of VPC Peering:

1. **Simplified Network Architecture:**
    - VPC peering simplifies the network architecture by allowing direct communication between instances in peered VPCs.
2. **Resource Sharing:**
    - Resources, such as Amazon EC2 instances or Amazon RDS databases, can be shared between VPCs.
3. **Cross-Account Peering:**
    - You can peer VPCs across different AWS accounts, facilitating collaboration between different entities.
4. **Cost-Efficiency:**
    - Instances communicate over the AWS network, avoiding the need for public IPs or NAT gateways.

VPC peering is a powerful feature that enhances the flexibility and scalability of AWS networking, enabling more seamless communication between different parts of your infrastructure.

Elastic IP (EIP)

An Elastic IP (EIP) is a static IPv4 address designed for dynamic cloud computing on the Amazon Web Services (AWS) platform. It is associated with your AWS account, not a particular instance, and you can allocate and release Elastic IP addresses as needed. Here are key points about Elastic IPs:

Characteristics and Use Cases:

1. **Static IP Address:**
   - Elastic IPs provide a static IPv4 address that you can associate with an instance in your VPC.
2. **Persistent:**
   - Once allocated, an Elastic IP address remains associated with your account until you explicitly release it.
3. **Public-Facing Resources:**
   - Elastic IPs are commonly used for public-facing resources like web servers, load balancers, and instances hosting applications that require a fixed public IP address.
4. **Avoiding IP Changes:**
   - If you stop and start an EC2 instance, it may receive a new public IP address. Assigning an Elastic IP to the instance ensures that it retains the same IP address even after stopping and starting.

Key Concepts:

1. **Elastic IP Pool:**
   - Each AWS region has a pool of Elastic IP addresses. You can allocate an Elastic IP from this pool and associate it with an EC2 instance in the same region.
2. **Association and Disassociation:**
   - You can associate an Elastic IP with an EC2 instance, network interface, or a Network Load Balancer. You can also disassociate an Elastic IP from an instance to use it with another one.

Steps to Allocate and Associate an Elastic IP:

1. **Allocate an Elastic IP:**
   - In the AWS Management Console, go to "Services" > "EC2."
   - Under "Network & Security," select "Elastic IPs."
   - Click "Allocate Elastic IP address."
2. **Associate Elastic IP with an Instance:**
   - In the Elastic IPs section, select the newly allocated Elastic IP.
   - Click "Actions" > "Associate Elastic IP address."
   - Choose the EC2 instance from the list or enter the instance ID.

Billing Considerations:

- There is a small charge for Elastic IPs that are not associated with a running instance.
- Ensure to release Elastic IPs when they are not in use to avoid unnecessary charges.

Elastic IPs provide a flexible way to manage IP addresses in the AWS environment, especially when you need a consistent public IP address for your resources. Always consider the best practices and cost implications when using Elastic IPs in your AWS infrastructure.

Network Access Control Lists (NACLs)

Network Access Control Lists (NACLs) are stateless, numbered sets of rules that control traffic at the subnet level in Amazon Virtual Private Cloud (VPC). NACLs are associated with subnets, and each subnet in a VPC must be associated with a NACL. Here are some key points about NACLs in AWS:

Characteristics and Key Concepts:

1. **Stateless Rules:**
   - NACLs are stateless, meaning that responses to allowed inbound traffic are subject to the rules for outbound traffic.
2. **Rule Evaluation Order:**
   - Rules are evaluated based on rule number in ascending order. The first rule that matches the traffic is applied, and subsequent rules are not evaluated.

3. **Numbered Rules:**
   - NACL rules are numbered from 100 to 32766. Lower rule numbers have higher priority.
4. **Default NACL:**
   - Each VPC comes with a default NACL that allows all inbound and outbound traffic. You can create custom NACLs and associate them with your subnets.
5. **Associating with Subnets:**
   - Each subnet in a VPC must be associated with a NACL. If not explicitly associated, a subnet is associated with the default NACL.

Example Use Cases:

1. **Blocking Specific Traffic:**
   - You can use NACLs to block specific types of traffic, such as blocking traffic on certain ports or from specific IP addresses.
2. **Isolating Subnets:**
   - NACLs provide an additional layer of security by allowing you to isolate different subnets within a VPC.

NACL Rules:

1. **Inbound Rules:**
   - Define rules for inbound traffic. For example, allowing inbound traffic on port 80 for web servers.
2. **Outbound Rules:**
   - Define rules for outbound traffic. For example, allowing outbound traffic on port 443 for secure communication.

Steps to Create and Configure a Custom NACL:

1. **Navigate to the VPC Dashboard:**
   - In the AWS Management Console, go to "Services" > "VPC."
2. **Create a Custom NACL:**
   - Under "Security," select "Network ACLs."

- Click the "Create network ACL" button.

3. **Configure the NACL:**
    - Give your NACL a name.
    - Associate it with the desired VPC.
4. **Define Inbound and Outbound Rules:**
    - Add inbound and outbound rules as needed, specifying the rule number, type (Allow/Deny), protocol, port range, and source/destination IP addresses.
5. **Associate the NACL with Subnets:**
    - After creating the NACL, associate it with the desired subnets.
    - In the "Subnet Associations" tab, click "Edit subnet associations" and associate the NACL with the selected subnets.