

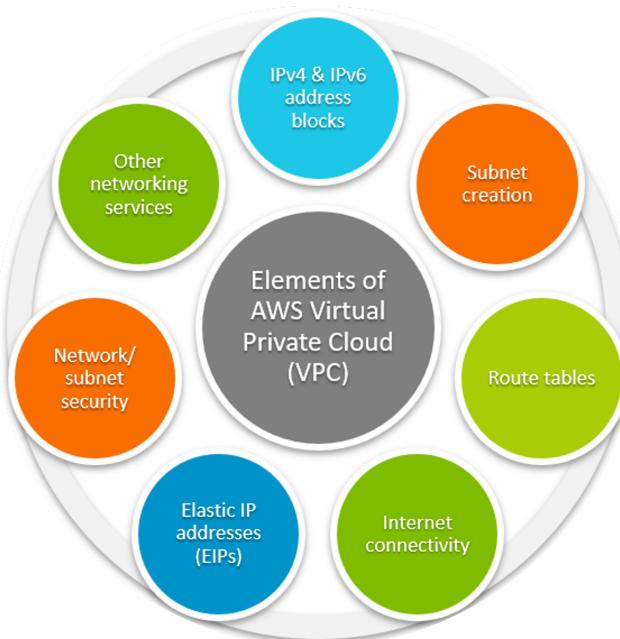
VPC Creation and setup.

👋 Welcome to my-blog!

VPC

VPC stands for **Virtual Private Cloud**. It's a virtual network dedicated to your AWS account. Just like a traditional network in a data center, you can **control** the **virtual networking environment**, including your **own IP address range, subnets, route tables, and network gateways**.

in Simple ways Imagine a secure, isolated section of the cloud where you can run your AWS resources. It's like having your own private piece of the internet where you decide who can access it and how they can interact with your services.

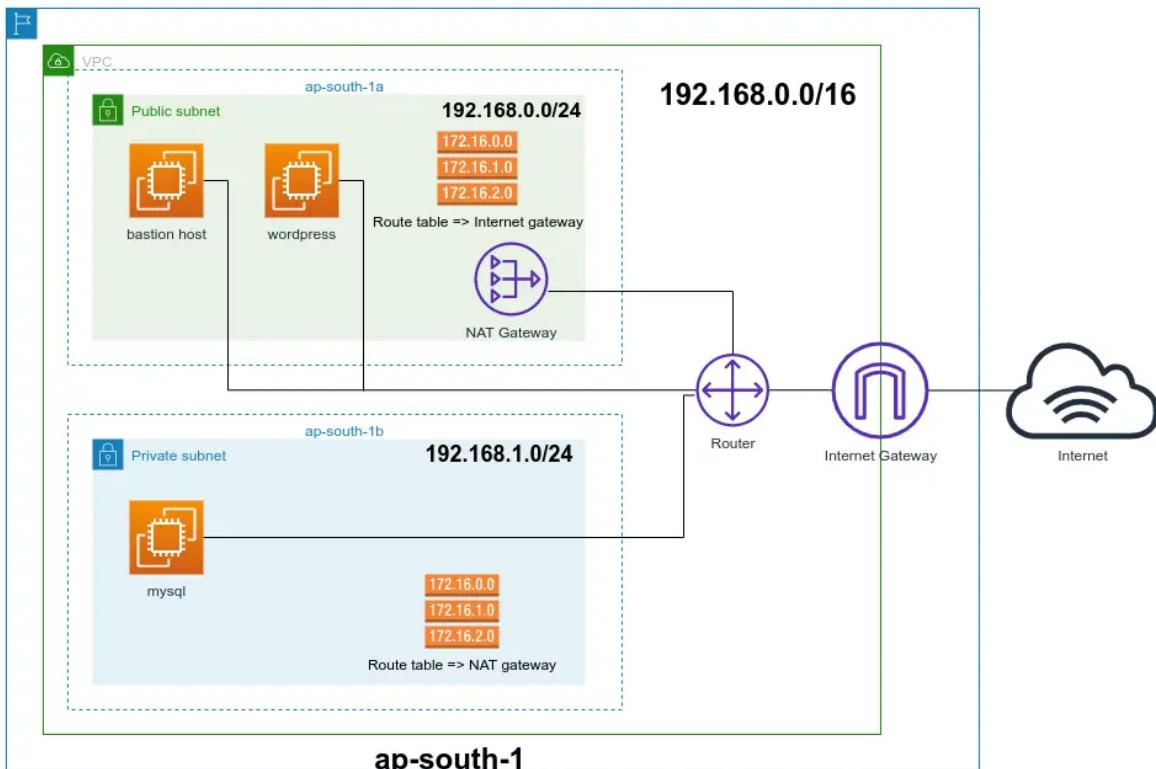


Use Case's

Imagine you're setting up a VPC for your company's cloud infrastructure. Here's how it might look:

TIP : (this descriptions are just for imagination).

- VPC Layout:** Your VPC is like a digital office building. It has different floors (subnets) for different departments, such as HR, Finance, and Development.
- Subnets:** Each subnet is like a room on a floor. You might have one subnet for HR, one for Finance, and one for Development. Each subnet has its own IP address range.
- Security Groups:** These act like security guards at the entrance of each room. You decide who can enter (access rules) and what they can do inside (inbound/outbound traffic rules).
- Internet Gateway:** This is like the main entrance to your office building. It allows resources in your VPC to communicate with the internet.
- Route Tables:** These are like maps that tell data packets where to go. You create routes to direct traffic within your VPC and to external destinations.
- NAT Gateway:** If your resources need to access the internet but you want to keep them hidden from inbound traffic, you can use a NAT gateway. It's like a proxy server that acts on behalf of your resources.



STEPS TO CREATE OUR OWN VPC NETWORK.

VPC network is basically our isolated networks created by AWS for increasing security with respect to the client.

to create VPC network in AWS we follows the process as below's

1. Sign in to the AWS Management Console:

- Log in to your AWS account using your credentials.

2. Navigate to the VPC Dashboard:

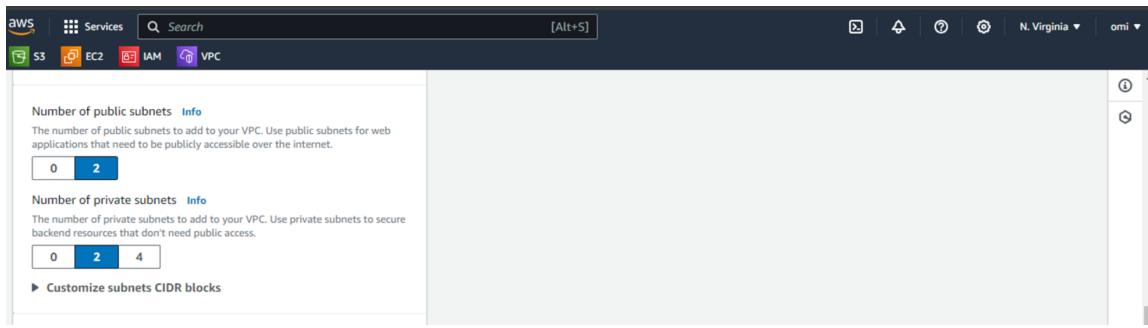
- In the AWS Management Console, go to the "Services" menu and select "VPC" under the "Networking & Content Delivery" section.

3. Create a VPC:

- start creating a VPC. Specify the name tag to your VPC. the IPv4 CIDR block for your VPC. This is the primary IP address range for your VPC. For example, **10.0.0.0/16**.

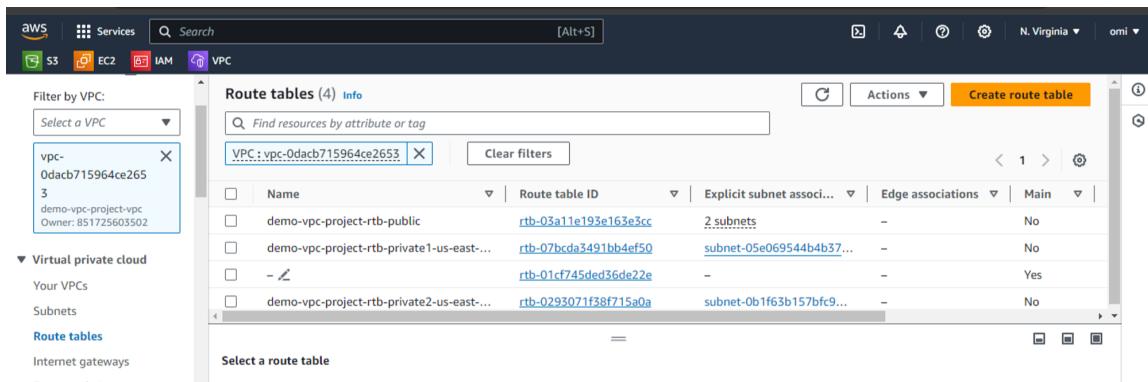
4. Configure Subnets:

- Within your VPC, create subnets. Subnets are segments of your VPC where you deploy your resources. Define each subnet's IPv4 CIDR block and the Availability Zone it belongs to. For example, **10.0.1.0/24** in **us-east-1a**. here we are keeping it default.



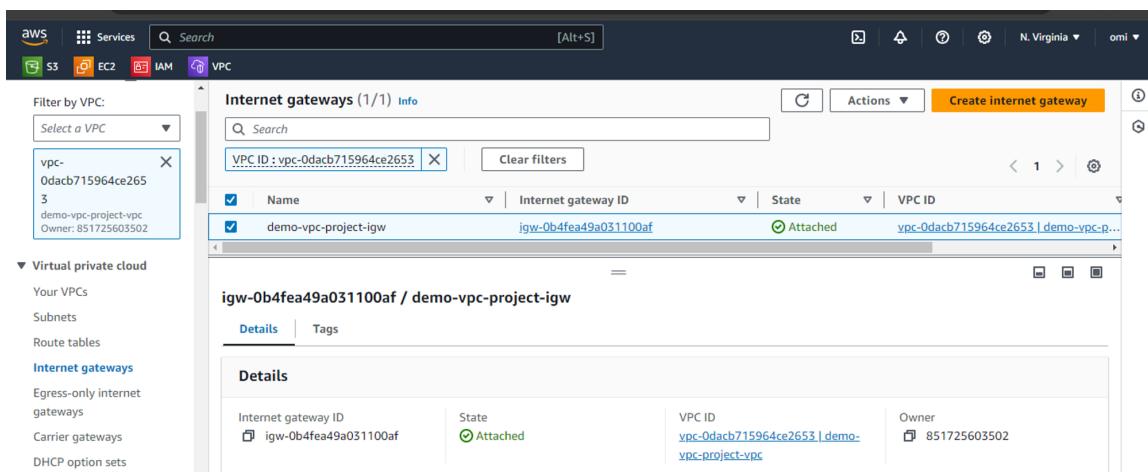
5. Create Route Tables:

- Route tables determine where network traffic is directed. By default, a main route table is created with your VPC. You may need to create additional route tables for specific purposes, such as public and private subnets.



6. Configure Internet Gateway (IGW) (for Public Subnets):

- If your VPC needs to communicate with the internet, create an internet gateway here I given it name as **demo-vpc-project-igw** and then attach it to your VPC. Associate the internet gateway with the route table of your public subnets.



7. Configure NAT Gateway or NAT Instance (for Private Subnets):

- If resources in your private subnets need internet access, you can set up a NAT gateway or NAT instance. This allows private subnet resources to initiate outbound connections to the internet while remaining inaccessible from the outside.

The screenshot shows the 'Create NAT gateway' page in the AWS VPC console. It includes fields for Name (optional), Subnet (selected), Connectivity type (Public), and a 'Create security group' button.

Name - optional
Create a tag with a key of 'Name' and a value that you specify. demo-vpc-NAT

Subnet
Select a subnet in which to create the NAT gateway. subnet-05e069544b4b37ccf (demo-vpc-project-subnet-private1-us-east-1a)

Connectivity type
<input checked="" type="radio"/> Public <input type="radio"/> Private

8. Set Up Security Groups:

- Security groups act as virtual firewalls for your EC2 instances and other resources. Define inbound and outbound traffic rules to control access to your resources. Associate security groups with your instances.

The screenshot shows the 'Security Groups (1/2)' page in the AWS VPC console. It lists two security groups: 'demo-vpc-sg' and 'default'. The 'demo-vpc-sg' group is selected.

Name	Security group ID	Security group name	VPC ID
sg-01cdafdd2e9ef83c2	sg-01cdafdd2e9ef83c2	demo-vpc-sg	vpc-0dacb715964ce2653
sg-0cf9f6edb92beef9d	sg-0cf9f6edb92beef9d	default	vpc-0dacb715964ce2653

9. (Optional) Configure Network ACLs:

- Network Access Control Lists (ACLs)** are stateless firewalls that control traffic at the subnet level. You can define rules to allow or deny traffic

based on IP addresses, ports, and protocols.

The screenshot shows the AWS VPC Details page. At the top, there's a search bar and navigation links for S3, EC2, IAM, and VPC. On the left, a sidebar lists various VPC-related options like Carrier gateways, DHCP option sets, and Network ACLs. The main panel displays the 'Details' tab for a Network ACL named 'act-05b0f67831d95d751'. It shows it's associated with 4 Subnets, has a Default Yes status, and a VPC ID of 'vpc-0dacb715964ce2653 / demo-vpc-project-vpc'. Below this, the 'Inbound rules' tab is selected, showing three rules:

Rule number	Type	Protocol	Port range	Source	Allow/Deny
50	HTTP* (8080)	TCP (6)	8080	0.0.0.0/0	✖ Deny
100	All traffic	All	All	0.0.0.0/0	✓ Allow
*	All traffic	All	All	0.0.0.0/0	✖ Deny

10. Launch Instances and Resources:

- Now that your VPC is set up, you can launch EC2 instances, ELB load balancers, and other AWS resources as per your requirements within your VPC.

The screenshot shows the AWS EC2 Instances page. The sidebar includes options like EC2 Dashboard, Global View, Events, and various instance-related tabs like Instances, Instance Types, Launch Templates, and AMIs. The main area shows one instance named 'demo-vpc-ec2' with ID 'i-0d3ae96f2b700b0a5', which is 'Running' and has an 't2.micro' instance type. A detailed view of this instance is shown below, including its platform (Amazon Linux), AMI ID ('ami-0bb84b8ffd87024d8'), and monitoring status (disabled). The bottom of the screen includes standard AWS footer links for CloudShell, Feedback, Privacy, Terms, and Cookie preferences.

11. Test Connectivity:

- After launching resources, verify connectivity both within your VPC and to external networks to ensure that **routing and security configurations are working** as expected.

Here I checked Network ACLs by **customizing default configurations** and **changing the priorities** by changing the **Rule numbers** of Inbound rules.

The screenshot shows the AWS Management Console interface for managing Network ACLs. The left sidebar lists various VPC-related services and security options. The main content area displays the details of a specific Network ACL, including its ID, association with four subnets, and status as the default. Below this, the 'Inbound rules' section is shown, containing three entries:

Rule number	Type	Protocol	Port range	Source	Allow/Deny
50	HTTP* (8080)	TCP (6)	8080	0.0.0.0/0	Deny
100	All traffic	All	All	0.0.0.0/0	Allow
*	All traffic	All	All	0.0.0.0/0	Deny

At the bottom of the page, there are links for CloudShell, Feedback, and legal notices.

THANK YOU !