**Lecture Notes: IP Addressing, Subnetting & AWS VPC (Hands-On Guide)**

**🔹 Section 1: Understanding IP Addressing and Subnetting**

Before diving into AWS VPC, let’s quickly revise how IP addressing and subnetting work.

**1.1 IP Addressing Basics**

* An **IP Address** is a unique identifier assigned to each device in a network.
* It consists of **Network Portion** and **Host Portion** (based on IP Class & Subnet Mask).
* IP Address Types: **Public IP, Private IP, Static IP, Dynamic IP**

**1.2 Private IP Ranges (Used in AWS VPC)**

AWS uses **private IPs** for VPC networking based on **RFC 1918**:

| **Class** | **Private IP Range** | **Default Subnet Mask** | **Total IPs** |
| --- | --- | --- | --- |
| **Class A** | 10.0.0.0 – 10.255.255.255 | 255.0.0.0 (/8) | **16 Million+ IPs** |
| **Class B** | 172.16.0.0 – 172.31.255.255 | 255.255.0.0 (/16) | **1 Million+ IPs** |
| **Class C** | 192.168.0.0 – 192.168.255.255 | 255.255.255.0 (/24) | **65,536 IPs** |

💡 **AWS VPC generally uses Class A or Class B IP ranges to allow scalable subnetting.**

**🔹 Section 2: What is AWS VPC?**

**2.1 AWS VPC Overview**

* **Amazon Virtual Private Cloud (VPC)** allows users to create an **isolated** network in AWS.
* You can create **subnets, route tables, internet gateways, and security groups**.
* Supports **IPv4 and IPv6** addressing.

**2.2 Components of AWS VPC**

| **Component** | **Description** |
| --- | --- |
| **VPC** | The overall **network container** where subnets and resources reside. |
| **Subnets** | A smaller division of a VPC where resources (EC2, RDS) are deployed. |
| **Internet Gateway (IGW)** | Allows public subnets to communicate with the internet. |
| **Route Tables** | Define how traffic flows between subnets and the internet. |
| **Security Groups** | Acts as a **firewall** controlling inbound/outbound traffic for resources. |
| **Network ACLs** | Optional **stateless firewall** at the subnet level. |

**🔹 Section 3: Subnetting in AWS VPC**

**3.1 Subnetting Concept**

* AWS **subnets** allow dividing a **VPC’s CIDR block** into multiple smaller **subnets**.
* Each subnet belongs to **one availability zone (AZ)**.
* Example of a **VPC CIDR Block**:  
  10.0.0.0/16 → Allows **65,536** IPs.  
  We can divide it into **smaller subnets** for different applications.

**3.2 Example of AWS Subnet Allocation**

| **Subnet Type** | **CIDR Block** | **Total IPs** | **Usable IPs (AWS Reserves 5)** |
| --- | --- | --- | --- |
| **Public Subnet** | 10.0.1.0/24 | 256 | 251 |
| **Private Subnet** | 10.0.2.0/24 | 256 | 251 |
| **Database Subnet** | 10.0.3.0/24 | 256 | 251 |

💡 **AWS reserves 5 IPs per subnet**:

* **Network Address** (10.0.1.0)
* **First Usable IP** (10.0.1.1)
* **Last Usable IP** (10.0.1.254)
* **Broadcast Address** (10.0.1.255)
* **AWS Reserved IP for DNS/other services**

**🔹 Section 4: Hands-on AWS VPC Setup**

**4.1 Creating a Custom VPC**

🔹 **Step 1:** Login to AWS Console → Open **VPC Dashboard**  
🔹 **Step 2:** Click **Create VPC**

* Name: MyCustomVPC
* IPv4 CIDR: 10.0.0.0/16
* IPv6 CIDR: **Disable** (optional)
* Tenancy: **Default**
* Click **Create VPC**

**4.2 Creating Subnets**

🔹 **Step 1:** Go to **Subnets** → Click **Create Subnet**

* **Public Subnet**
  + VPC: MyCustomVPC
  + Subnet CIDR: 10.0.1.0/24
  + Availability Zone: us-east-1a
* **Private Subnet**
  + VPC: MyCustomVPC
  + Subnet CIDR: 10.0.2.0/24
  + Availability Zone: us-east-1b
* **Database Subnet**
  + VPC: MyCustomVPC
  + Subnet CIDR: 10.0.3.0/24
  + Availability Zone: us-east-1c
* Click **Create**

**4.3 Creating an Internet Gateway (IGW)**

🔹 **Step 1:** Go to **Internet Gateways** → Click **Create IGW**

* Name: MyIGW
* Click **Create**  
  🔹 **Step 2:** Attach IGW to VPC
* Select MyCustomVPC
* Click **Attach to VPC**

**4.4 Configuring Route Tables**

🔹 **Step 1:** Go to **Route Tables** → Click **Create Route Table**

* Name: PublicRT
* VPC: MyCustomVPC
* Click **Create**  
  🔹 **Step 2:** Edit Routes
* Add route 0.0.0.0/0 → Target: **Internet Gateway (MyIGW)**
* Save

🔹 **Step 3:** Associate Public Subnet

* Go to **Subnet Associations**
* Select **Public Subnet (10.0.1.0/24)**
* Click **Save**

**4.5 Creating Security Groups**

🔹 **Step 1:** Go to **Security Groups** → Click **Create Security Group**

* Name: PublicSG
* VPC: MyCustomVPC 🔹 **Step 2:** Configure Inbound Rules
* Allow **SSH (22)** from **Your IP**
* Allow **HTTP (80)** from **Anywhere**
* Allow **HTTPS (443)** from **Anywhere** 🔹 **Step 3:** Configure Outbound Rules
* Allow **All Traffic**
* Save

**4.6 Launching an EC2 Instance in VPC**

🔹 **Step 1:** Go to **EC2 Dashboard** → Click **Launch Instance**

* Name: MyWebServer
* AMI: Amazon Linux 2
* Instance Type: t2.micro
* VPC: MyCustomVPC
* Subnet: Public Subnet
* Security Group: PublicSG
* Key Pair: Create or use existing
* Click **Launch**

🔹 **Step 2:** Assign Elastic IP (Optional)

* Go to **Elastic IPs** → Allocate
* Associate with **MyWebServer**

**Summary**

✅ IP Addressing and Subnetting are crucial in AWS networking.  
✅ AWS VPC allows **customized private networking** using **subnets, route tables, and security groups**.  
✅ Hands-on AWS VPC setup includes **creating VPC, subnets, IGW, security groups, and launching EC2**.

**Lecture Notes: IP Addressing, Subnetting & AWS VPC Configuration**

**📌 Section 1: Understanding IP Addressing**

**1.1 What is an IP Address?**

An IP (Internet Protocol) address is a **unique identifier** assigned to each device on a network to enable communication. It consists of **two parts:**

* **Network Portion:** Identifies the network
* **Host Portion:** Identifies the specific device (host) within the network

**1.2 IPv4 vs IPv6**

| **Feature** | **IPv4** | **IPv6** |
| --- | --- | --- |
| Address Length | 32-bit | 128-bit |
| Format | Decimal (e.g., 192.168.1.1) | Hexadecimal (e.g., 2001:db8::ff00:42:8329) |
| Address Space | 4.3 billion | Virtually unlimited |
| Example Usage | Local & Internet | IoT, high-scale applications |

**1.3 IP Address Classes**

| **Class** | **Range** | **Network-Host Allocation** | **Usable Hosts** |
| --- | --- | --- | --- |
| **A** | 1.0.0.0 - 126.255.255.255 | N.H.H.H | 16M |
| **B** | 128.0.0.0 - 191.255.255.255 | N.N.H.H | 65K |
| **C** | 192.0.0.0 - 223.255.255.255 | N.N.N.H | 254 |

**1.4 Subnetting Explained**

* **CIDR Notation:** /24 means first **24 bits** are network bits, and the remaining are host bits.
* Example: 192.168.1.0/24
  + Network: 192.168.1.0
  + Usable Hosts: 192.168.1.1 - 192.168.1.254
  + Broadcast: 192.168.1.255

**📌 Section 2: AWS VPC and Subnetting**

**2.1 What is a VPC?**

A **Virtual Private Cloud (VPC)** is an isolated network within AWS that allows users to configure **subnets, route tables, and internet access.**

**2.2 Creating a VPC in AWS**

📌 **Step 1:** Go to **VPC Dashboard** → Click **Create VPC**

* **Name:** MyCustomVPC
* **CIDR Block:** 10.0.0.0/16
* Click **Create**

**2.3 Creating Public & Private Subnets**

| **Subnet Name** | **CIDR Block** | **Type** |
| --- | --- | --- |
| Public-Subnet | 10.0.1.0/24 | Public |
| Private-Subnet | 10.0.2.0/24 | Private |

**📌 Section 3: Configuring Network Access in AWS**

**3.1 Setting Up an Internet Gateway (IGW)**

📌 **Step 1:** Go to **Internet Gateways** → Click **Create IGW** 📌 **Step 2:** Attach it to MyCustomVPC

**3.2 Configuring Route Tables**

| **Route Table Name** | **Routes** |
| --- | --- |
| Public-RT | 0.0.0.0/0 → IGW |
| Private-RT | No internet access initially |

**📌 Section 4: Connecting Private EC2 Using a Bastion Host**

**4.1 Launching a Public EC2 (Bastion Host)**

📌 **Step 1:** Launch EC2 in **Public-Subnet** → Enable **Public IP** 📌 **Step 2:** Configure Security Group → Allow **SSH (22) from your IP**

**4.2 Launching a Private EC2**

📌 **Step 1:** Launch EC2 in **Private-Subnet** → Disable **Public IP** 📌 **Step 2:** Configure Security Group → Allow **SSH from Bastion Host**

**4.3 SSH Access Flow**

Connect to **Public EC2 (Bastion Host)**:

ssh -i mykey.pem ec2-user@<Public-EC2-IP>

From **Bastion Host**, SSH into **Private EC2**:

ssh -i mykey.pem ec2-user@<Private-EC2-Private-IP>

🎉 **Now, you have accessed the private EC2 via the Public Bastion Host!**

**📌 Section 5: Setting Up NAT Gateway**

**5.1 Create a NAT Gateway**

📌 **Step 1:** Go to **NAT Gateways** → Click **Create NAT Gateway**

* **Subnet:** Public-Subnet
* **Elastic IP:** Allocate new one

**5.2 Modify Private Route Table**

📌 **Step 1:** Edit Private-RT 📌 **Step 2:** Add 0.0.0.0/0 → NAT Gateway

🎉 **Now, Private EC2 can access the internet but remains private!**

**📌 Section 6: Configuring VPC Peering**

**6.1 Create a Second VPC**

📌 **Step 1:** Go to **VPC Dashboard** → Click **Create VPC**

* Name: VPC-2
* CIDR: 192.168.0.0/16
* Create Subnet: 192.168.1.0/24

**6.2 Establish VPC Peering**

📌 **Step 1:** Go to **VPC Peering** → Click **Create Peering Connection**

* **Requester VPC:** MyCustomVPC
* **Accepter VPC:** VPC-2 📌 **Step 2:** Accept Peering Request

**6.3 Modify Route Tables**

📌 **Step 1:** Add routes in MyCustomVPC-RT

* 192.168.0.0/16 → VPC Peering 📌 **Step 2:** Add routes in VPC-2-RT
* 10.0.0.0/16 → VPC Peering

🎉 **Now, both VPCs can communicate!**

**🚀 Final Summary**

✅ **IP Addressing & Subnetting Explained**  
✅ **AWS VPC Setup with Public & Private Subnets**  
✅ **SSH into Private EC2 using Bastion Host**  
✅ **NAT Gateway for Internet Access in Private Subnet**  
✅ **VPC Peering for Cross-VPC Communication**

💡 **This setup is ideal for secure backend workloads, multi-tier applications, and cross-VPC networking!**

**📌 Lecture Notes: IP Addressing, Subnetting & AWS VPC with NAT Gateway and VPC Peering (Hands-On Guide)**

**🔹 step-by-step instructions** to:

✅ **Connect a Private EC2 Instance using a Public EC2 (Bastion Host)**  
✅ **Set up NAT Gateway for internet access in a Private Subnet**  
✅ **Configure VPC Peering for cross-VPC communication**

**🔹 Section 2: AWS VPC Setup**

We are going to:  
1️. **Create a Custom VPC with Public and Private Subnets**  
2️. **Set up an Internet Gateway (IGW) for public internet access**  
3️. **Configure Route Tables for Subnet Communication**  
4️. **Launch a Public EC2 (Bastion Host) & Private EC2**  
5️. **Use NAT Gateway to allow private EC2 to access the internet**  
6️. **Configure VPC Peering for cross-VPC communication**

**🔹 Section 3: Hands-on AWS VPC Setup**

**3.1 Creating a Custom VPC**

📌 **Step 1:** Login to AWS Console → Open **VPC Dashboard**  
📌 **Step 2:** Click **Create VPC**

* **Name:** MyCustomVPC
* **IPv4 CIDR:** 10.0.0.0/16
* **IPv6 CIDR:** Disable
* **Tenancy:** Default
* Click **Create VPC**

**3.2 Creating Subnets**

📌 **Step 1:** Go to **Subnets** → Click **Create Subnet**

| **Subnet Name** | **CIDR Block** | **Availability Zone** | **Subnet Type** |
| --- | --- | --- | --- |
| Public-Subnet | 10.0.1.0/24 | us-east-1a | Public |
| Private-Subnet | 10.0.2.0/24 | us-east-1b | Private |

* Click **Create Subnets**

**3.3 Creating an Internet Gateway (IGW)**

📌 **Step 1:** Go to **Internet Gateways** → Click **Create IGW**  
📌 **Step 2:** Name it MyIGW, click **Create**  
📌 **Step 3:** Attach to MyCustomVPC

**3.4 Configuring Route Tables**

📌 **Step 1:** Go to **Route Tables** → Click **Create Route Table**

| **Route Table Name** | **Subnet Association** | **Routes** |
| --- | --- | --- |
| Public-RT | Public Subnet | 0.0.0.0/0 → **IGW** |
| Private-RT | Private Subnet | No internet access initially |

* Click **Create**

**3.5 Launching a Public EC2 (Bastion Host)**

📌 **Step 1:** Go to **EC2 Dashboard** → Click **Launch Instance**  
📌 **Step 2:** Select **Amazon Linux 2 AMI**  
📌 **Step 3:** Choose t2.micro instance type  
📌 **Step 4:**

* **VPC:** MyCustomVPC
* **Subnet:** Public-Subnet
* **Auto-assign Public IP:** **Enable**
* **Security Group:** PublicSG
  + Allow **SSH (22)** from **your IP**
  + Allow **HTTP (80)** from **Anywhere**  
    📌 **Step 5:** Click **Launch**

**3.6 Launching a Private EC2 Instance**

📌 **Step 1:** Go to **EC2 Dashboard** → Click **Launch Instance**  
📌 **Step 2:** Select **Amazon Linux 2 AMI**  
📌 **Step 3:** Choose t2.micro instance type  
📌 **Step 4:**

* **VPC:** MyCustomVPC
* **Subnet:** Private-Subnet
* **Auto-assign Public IP:** **Disable**
* **Security Group:** PrivateSG
  + Allow **SSH (22)** only from **Public EC2 (Bastion Host)**  
    📌 **Step 5:** Click **Launch**

**🔹 Section 4: Connecting Private EC2 using Bastion Host**

📌 **Step 1:** Connect to **Public EC2**

ssh -i mykey.pem ec2-user@<Public-EC2-IP>

📌 **Step 2:** From the Bastion Host, SSH into **Private EC2**

ssh -i mykey.pem ec2-user@<Private-EC2-Private-IP>

🎉 **Now, you have accessed the private EC2 via the Public Bastion Host!**

**🔹 Section 5: Setting Up NAT Gateway (For Private EC2 Internet Access)**

**5.1 Create a NAT Gateway**

📌 **Step 1:** Go to **VPC Dashboard → NAT Gateways → Create NAT Gateway**  
📌 **Step 2:**

* **Subnet:** Public-Subnet
* **Elastic IP:** Allocate a new one  
  📌 **Step 3:** Click **Create**

**5.2 Modify Private Route Table**

📌 **Step 1:** Go to **Route Tables**  
📌 **Step 2:** Select Private-RT, click **Edit Routes**  
📌 **Step 3:** Add 0.0.0.0/0 → **NAT Gateway**  
📌 **Step 4:** Click **Save**

🎉 **Now, the Private EC2 instance can access the internet but is not publicly accessible!**

**🔹 Section 6: VPC Peering (Connecting Two VPCs)**

**6.1 Create Second VPC**

📌 **Step 1:** Go to **VPC Dashboard** → Create **Second VPC**

* Name: VPC-2
* CIDR: 192.168.0.0/16 📌 **Step 2:** Create **Subnet (192.168.1.0/24)**

**6.2 Create VPC Peering Connection**

📌 **Step 1:** Go to **VPC Peering** → Click **Create Peering Connection**  
📌 **Step 2:**

* **VPC 1 (Requester):** MyCustomVPC
* **VPC 2 (Accepter):** VPC-2  
  📌 **Step 3:** Accept the request in VPC-2

**6.3 Modify Route Tables for Communication**

📌 **Step 1:** Edit MyCustomVPC Route Table

* Add 192.168.0.0/16 → **Target: VPC Peering Connection**  
  📌 **Step 2:** Edit VPC-2 Route Table
* Add 10.0.0.0/16 → **Target: VPC Peering Connection**

🎉 **Now, both VPCs can communicate with each other!**

**🚀 Final Summary**

✅ **Public EC2 can SSH into Private EC2 (Bastion Host Approach)**  
✅ **Private EC2 can access the internet via NAT Gateway**  
✅ **VPC Peering allows cross-VPC communication**

💡 **This setup is widely used for private workloads, secure backend processing, and multi-VPC architectures.**