**VPC –TASK1**

**1.Create VPC with 2 private and 2 public subnets.**

## **Steps to Create VPC (with 2 Public + 2 Private Subnets)**

### ****1. Create VPC****

1. Go to **VPC service** in AWS Console.
2. Click **Create VPC** → select **VPC only** (or VPC + subnets wizard if available).
3. Enter details:
   * Name: My-VPC
   * CIDR block: 10.0.0.0/16 (gives you lots of IPs).
   * Tenancy: Default.
4. Click **Create VPC**.

### ****2. Create Subnets****

Now we will divide the CIDR 10.0.0.0/16 into **4 subnets**.

* **Public Subnet 1 (AZ1)** → 10.0.1.0/24
* **Public Subnet 2 (AZ2)** → 10.0.2.0/24
* **Private Subnet 1 (AZ1)** → 10.0.3.0/24
* **Private Subnet 2 (AZ2)** → 10.0.4.0/24

Steps:

1. Go to **Subnets** → Create subnet.
2. Select **My-VPC**.
3. Add each subnet one by one with correct CIDR + Availability Zone (AZ).
4. For public subnets → check the box **Auto-assign public IPv4** = enabled.

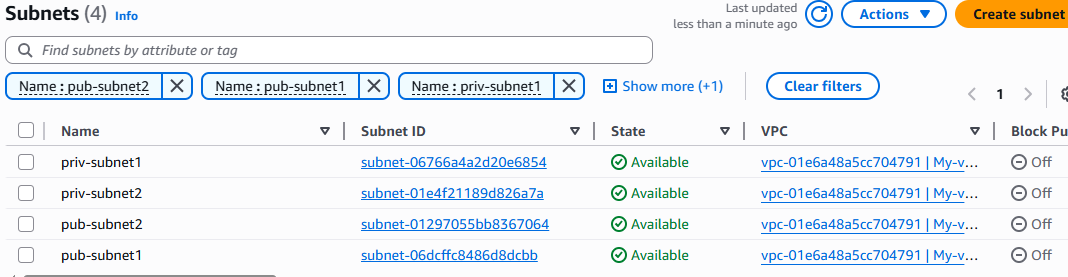
### ****3. Create Internet Gateway (IGW)****

1. Go to **Internet Gateways** → Create IGW.
2. Attach it to **My-VPC**.

### ****4. Create Route Tables****

We need **2 route tables**:

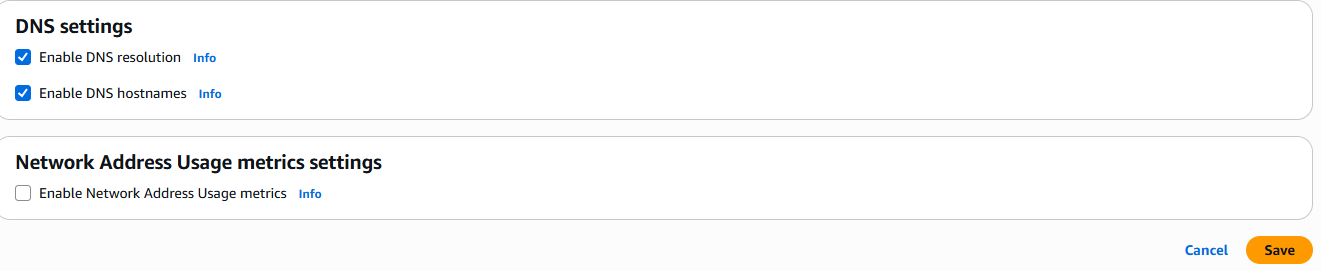
1. **Public Route Table** →
   * Routes:
     + 10.0.0.0/16 → local (default)
     + 0.0.0.0/0 → IGW (for internet access)
   * Associate **both Public Subnets**.
2. **Private Route Table** →
   * Routes:
     + 10.0.0.0/16 → local
     + (Optional) 0.0.0.0/0 → NAT Gateway (if you want private subnets to access internet).
   * Associate **both Private Subnets**.



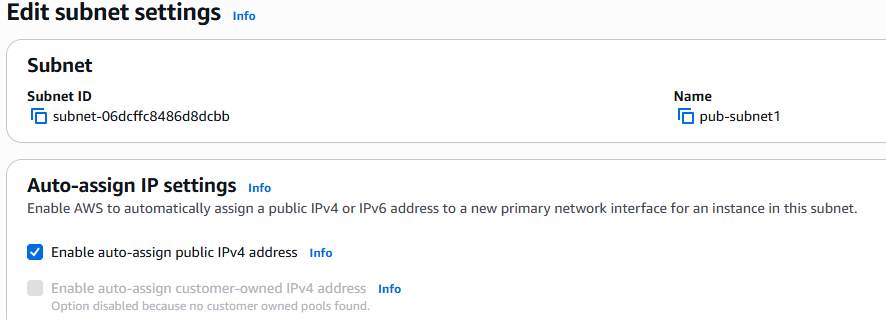
**2 .Enable DNS Hostname in VPC.**

**Steps to Enable DNS Hostnames in a VPC**

1. Go to **AWS Management Console** → **VPC**.
2. Select your VPC (My-VPC).
3. In the left panel, click **Your VPCs**.
4. Select your VPC → choose **Actions** → **Edit VPC settings**.
5. Check the box:
   * ✅ **Enable DNS resolution** (usually enabled by default).
   * ✅ **Enable DNS hostnames** (this is the one you need).
6. Click **Save changes**.



**3.Enable Auto Assign Public IP in 2 public subnets.**

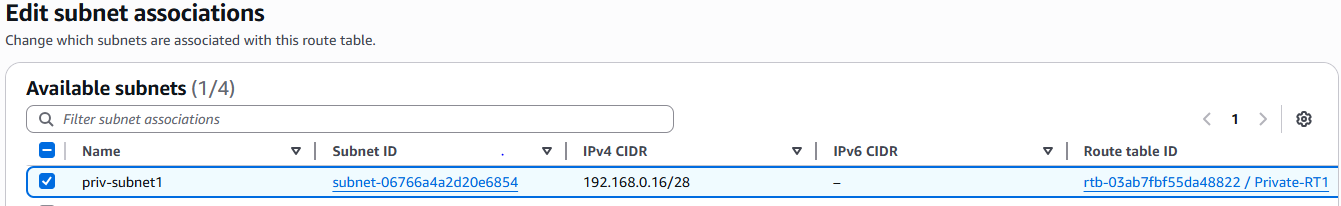


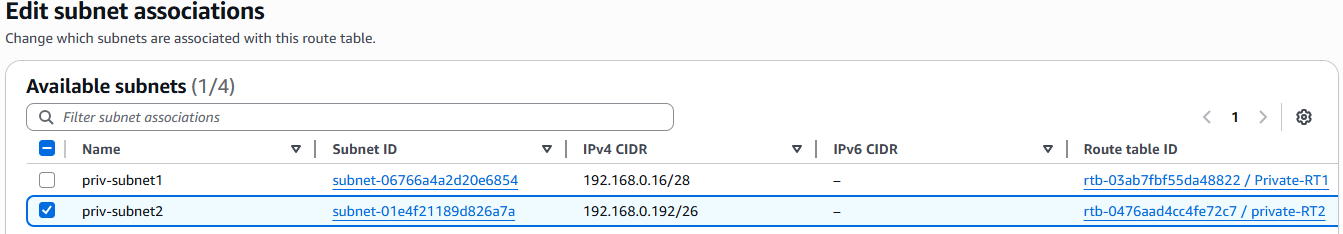


**4. Add 2 private subnets in private route table.**

### ****Steps in AWS Console****

1. Go to **VPC** service → **Route Tables**.
2. Select your **Private Route Table** (created earlier).
3. Go to the **Subnet Associations** tab.
4. Click **Edit subnet associations**.
5. Select your **2 Private Subnets** (example: 10.0.3.0/24 and 10.0.4.0/24).
6. Save associations ✅.

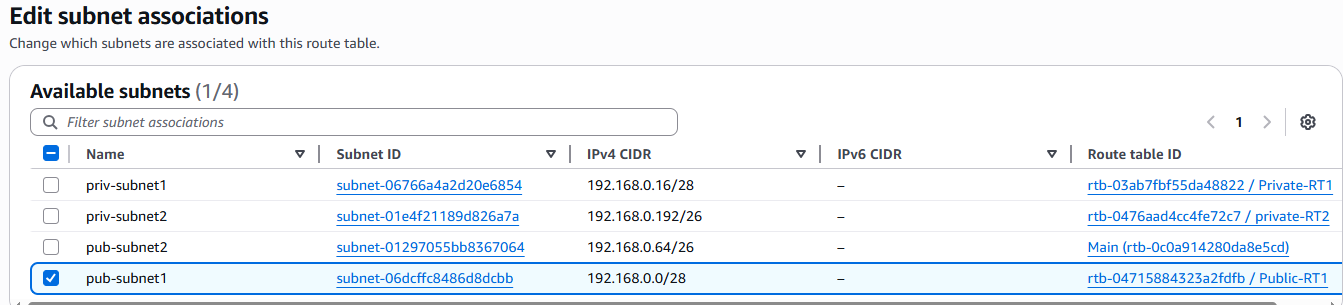


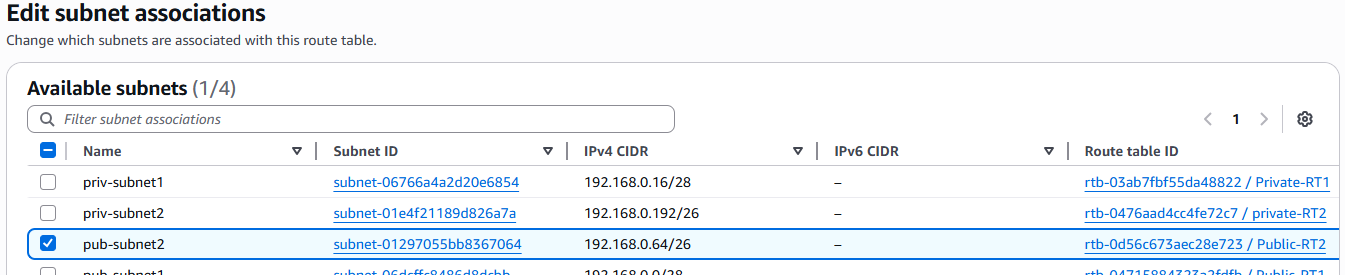


**5 .Add 2 public subnets in public route table.**

### ****Steps in AWS Console****

1. Go to **VPC service** → **Route Tables**.
2. Select your **Public Route Table**.
3. Open the **Subnet Associations** tab.
4. Click **Edit subnet associations**.
5. Select your **2 Public Subnets** (example: 10.0.1.0/24 and 10.0.2.0/24).
6. Save associations ✅

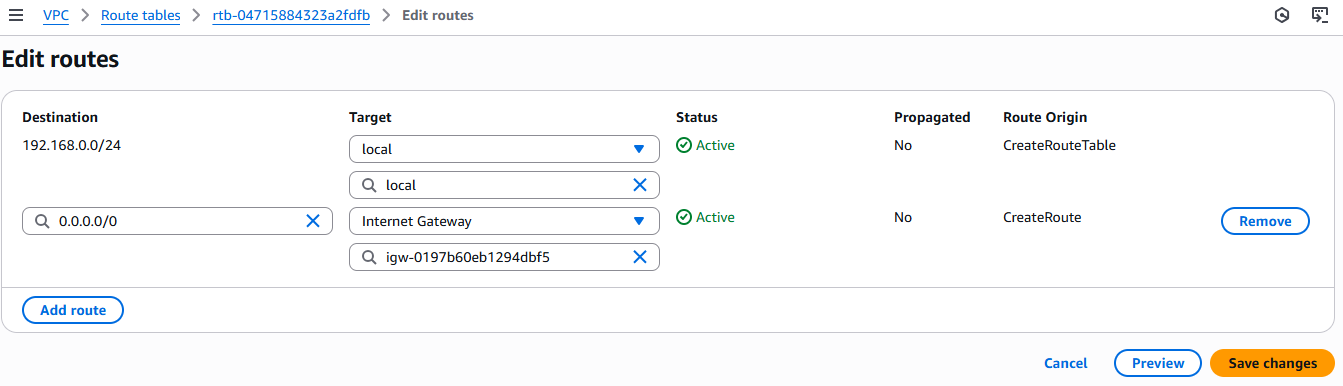




**6 .Public route table will have the routes to internet and local.**

A **Public Route Table** always has **two key routes**:

1. **Local Route**
   * Destination: 10.0.0.0/16 (or your VPC CIDR block)
   * Target: **local**
   * Purpose: Allows communication between all subnets inside the VPC (private ↔ public).
2. **Internet Route**
   * Destination: 0.0.0.0/0
   * Target: **Internet Gateway (IGW)**
   * Purpose: Allows resources in public subnets to send/receive traffic from the internet.



**7. Create EC2 in public subnet with t2.micro and install PHP.**

## **Step 1: Launch EC2 in Public Subnet**

1. Go to **AWS Console → EC2** → **Launch instance**.
2. Enter details:
   * Name: Public-EC2-PHP
   * AMI: **Amazon Linux 2 (or Amazon Linux 2023)**
   * Instance type: **t2.micro** (Free tier eligible).
3. Key pair: Create or choose existing key pair (for SSH access).
4. Network settings:
   * VPC: Select your **My-VPC**.
   * Subnet: Select **Public Subnet 1**.
   * Auto-assign public IP: **Enable**.
   * Security group: Create new → allow:
     + SSH (22) from your IP
     + HTTP (80) from anywhere (0.0.0.0/0)
5. Storage: Default 8GB is fine.
6. Click **Launch instance**.

## **Step 2: Connect to EC2**

From your terminal (Git Bash / Linux / Mac):

ssh -i my-key.pem ec2-user@<public-ip>

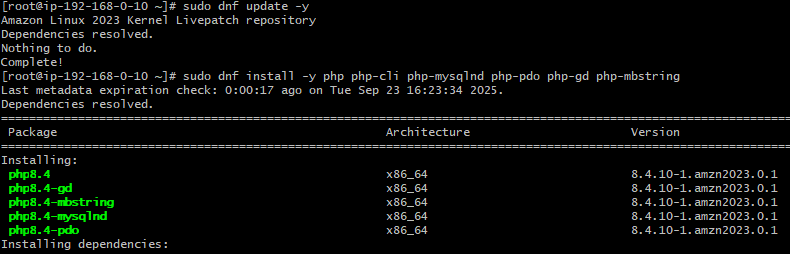
## **Step 3: Install PHP**

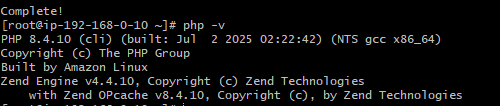
Once inside EC2:

For **Amazon Linux 2023**:

sudo dnf update -y

sudo dnf install -y php php-cli php-mysqlnd php-pdo php-gd php-mbstring





**8.Configure NAT gateway in public subnet and connect to private instance.**

## **Step 1: Create a NAT Gateway**

1. Go to **VPC service** → **NAT Gateways**.
2. Click **Create NAT Gateway**.
   * Name: My-NAT-GW.
   * Subnet: Choose a **Public Subnet** (must have a route to IGW).
   * Elastic IP: Allocate a new Elastic IP and attach.
3. Click **Create NAT Gateway**.

## **Step 2: Update Route Table for Private Subnets**

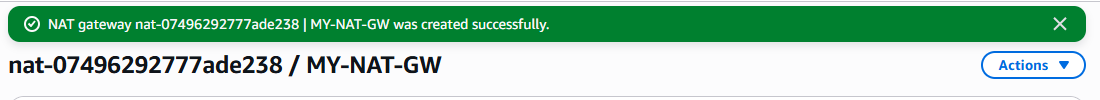
1. Go to **Route Tables** in VPC.
2. Select your **Private Route Table**.
3. Edit routes → Add new route:
   * Destination: 0.0.0.0/0
   * Target: nat-xxxxxxxx (your NAT Gateway ID).
4. Save changes.

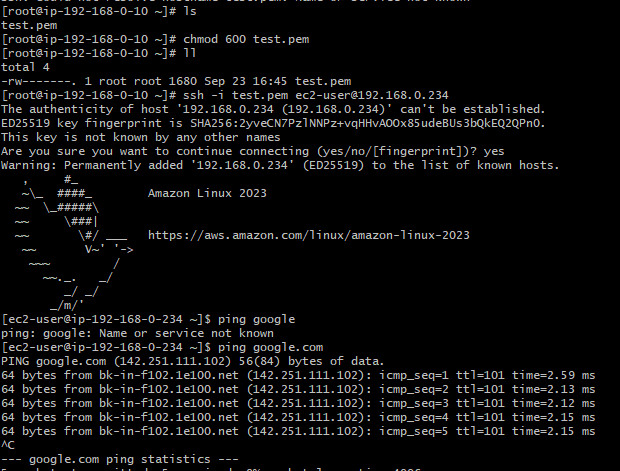
✅ Now, Private Subnets have internet access **through NAT Gateway**.

## **Step 3: Test Internet Access from Private EC2**

1. Launch an EC2 instance in **Private Subnet**.
   * Security group: allow SSH only from your **Public EC2 (Bastion Host)**.
2. SSH into **Public EC2** first:
3. ssh -i my-key.pem ec2-user@<public-ec2-public-ip>
4. From inside Public EC2, SSH into Private EC2 (using private IP):
5. ssh ec2-user@10.0.3.25
6. From Private EC2, test internet:

ping google.com





**9.Install Apache Tomcat in private EC2 and deploy a sample app.**

## **Step 1: Connect to Private EC2**

Use the **Public EC2 (bastion host)** to SSH into the private EC2:

ssh -i my-key.pem ec2-user@<private-ec2-private-ip>

## **Step 2: Update and Install Java**

Tomcat requires Java, so install it:

sudo yum update -y

sudo yum install -y java-11-amazon-corretto

Check version:

java -version

## **Step 3: Download and Install Apache Tomcat**

1. Go to /opt directory:

cd /opt

sudo wget https://downloads.apache.org/tomcat/tomcat-9/v9.0.91/bin/apache-tomcat-9.0.91.tar.gz

1. Extract:

sudo tar -xvzf apache-tomcat-9.0.91.tar.gz

sudo mv apache-tomcat-9.0.91 tomcat9

1. Make scripts executable:

sudo chmod +x /opt/tomcat9/bin/\*.sh

## **Step 4: Start Tomcat**

cd /opt/tomcat9/bin

sudo ./startup.sh

Check logs:

tail -f /opt/tomcat9/logs/catalina.out

## **Step 5: Deploy a Sample App**

1. Download a sample .war file (Java web app):

cd /opt/tomcat9/webapps

sudo wget https://tomcat.apache.org/tomcat-9.0-doc/appdev/sample/sample.war

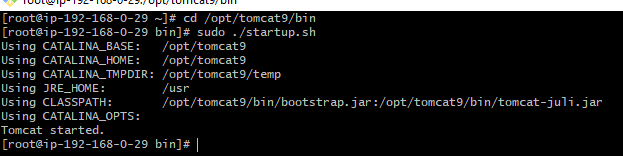
1. Tomcat will **automatically deploy** it (extracts to sample/ folder).

## **Step 6: Verify Deployment**

Since this is a **private EC2**, you cannot access it directly from the internet.

✅ Options:

* Use **curl** inside the private EC2:



**10 . Configure VPC flow logs and store the logs in S3 and CloudWatch.**

