Guide to Setting Up a VPC with Subnets, Gateways, and EC2 Instances

1. Create the VPC

Step: In the AWS Management Console, navigate to 'VPC' and click 'Create VPC'.

Choose an IP range that supports future scalability. For example, use the CIDR block `10.0.0.0/16`, which allows for a large number of IP addresses for extensive growth.

Assign a name like `MyVPC` to easily identify your VPC.

Tenancy: Select 'Default' unless you require dedicated hardware for your instances.

2. Create Subnets

Create two public and two private subnets in separate Availability Zones (AZs) to ensure high availability:

Public Subnet 1:

CIDR block: `10.0.0.0/24`
Availability Zone: Choose `ap-southeast-1a` or any AZ in your region.
Enable Auto-assign Public IPv4: Yes

Public Subnet 2:

CIDR block: `10.0.1.0/24`
Availability Zone: Select `ap-southeast-1b` or another available AZ.
Enable Auto-assign Public IPv4: Yes

Private Subnet 1:

CIDR block: `10.0.2.0/24`
Availability Zone: `ap-southeast-1a`

Private Subnet 2:

CIDR block: `10.0.3.0/24`
Availability Zone: `ap-southeast-1b`

3. Create an Internet Gateway

Step: In the VPC dashboard, go to 'Internet Gateways' and click 'Create Internet Gateway'.

Attach the Internet Gateway to `MyVPC` created in Step 1.

4. Create Route Tables

Public Route Table:
- Go to 'Route Tables' and create a route table for the public subnets.
- Associate this route table with both public subnets (`10.0.0.0/24` and `10.0.1.0/24`).
- Add a route that sends all traffic (`0.0.0.0/0`) to the Internet Gateway.

Private Route Table:
- Create a route table for the private subnets.
- Associate this route table with both private subnets (`10.0.2.0/24` and `10.0.3.0/24`).

5. Create a NAT Gateway

Step: Go to 'NAT Gateways' and create a NAT Gateway in one of the public subnets (e.g., `10.0.1.0/24`).

Allocate an Elastic IP and attach it to the NAT Gateway.

Update the private route table to send all outbound traffic (`0.0.0.0/0`) to the NAT Gateway.

6. Launch EC2 Instances

Frontend EC2 (in Public Subnet):

AMI: Choose Ubuntu 24.04 or your preferred OS.
Subnet: Use `10.0.0.0/24` (Public Subnet 1).
Security Group: Allow HTTP (port 80) and SSH (port 22).

Backend EC2 (in Private Subnets):

AMI: Choose Ubuntu 24.04 for backend servers.
Subnet: Select `10.0.2.0/24` and `10.0.3.0/24`.
Security Group: Restrict access to only the frontend instance via HTTP or a custom port.

Database EC2 (in Private Subnets):

AMI: Choose Ubuntu 24.04 or a PostgreSQL image.
Subnet: Use private subnets `10.0.2.0/24` and `10.0.3.0/24`.
Security Group: Allow access only from backend servers on database ports (e.g., PostgreSQL on port 5432).

7. Configure Security Groups

Frontend Security Group: Allow HTTP (80), HTTPS (443), and SSH (22) from your IP. Allow internal traffic from the backend security group.

Backend Security Group: Restrict to allow only traffic from the frontend on the necessary port.

Database Security Group: Restrict access to PostgreSQL (port 5432) from backend servers only.

8. Connect and Test

Frontend: Connect to the public EC2 instance via SSH using its public IP.
- Step 1: Create a file, paste your `pem.key` in it.
- Step 1.1: Set read-only permissions for the file (chmod 400).

Backend: From the frontend, connect to backend instances using private IPs.
- Step 2: Repeat the key setup and ensure permissions are secure.

Database: Verify the backend EC2 can connect to the database EC2 over its private IP. Ensure connectivity and permissions are correctly configured.