VPC notes – Noor

**How to Open the VPC Dashboard:**

1. **Login to AWS Console**:  
   Go to https://console.aws.amazon.com/ and sign in.
2. **Search for "VPC"**:  
   In the **top search bar**, type VPC and select **"VPC"** from the dropdown (it will say “Virtual Private Cloud”).
3. You will now be in the **VPC Dashboard**, where you can manage:
   * VPCs
   * Subnets
   * Route Tables
   * Internet Gateways
   * NAT Gateways, etc.

<https://console.aws.amazon.com/vpc/home>

## VPC Architecture Overview:

* **1 VPC**
* **2 Availability Zones**
* **2 Public Subnets (1 in each AZ)**
* **2 Private Subnets (1 in each AZ)**
* **1 Internet Gateway (IGW)**
* **1 NAT Gateway (for private subnet internet access)**

## 🌐 Step-by-Step (AWS Console)

### ****Step 1: Create VPC****

1. Go to **VPC Dashboard > Your VPCs > Create VPC**
2. Choose **"VPC and more"**
3. Fill in the following:
   * **Name**: MyVPC
   * **IPv4 CIDR block**: 10.0.0.0/16
   * **Tenancy**: Default
4. Leave IPv6 unchecked (optional), click **Create VPC**.

### ****Step 2: Create Subnets****

Go to **Subnets > Create subnet**

#### ****Public Subnet 1**** (AZ-a)

* **Name**: PublicSubnet1
* **VPC**: MyVPC
* **AZ**: us-east-1a (example)
* **CIDR block**: 10.0.1.0/24

#### ****Public Subnet 2**** (AZ-b)

* **Name**: PublicSubnet2
* **AZ**: us-east-1b
* **CIDR block**: 10.0.2.0/24

#### ****Private Subnet 1**** (AZ-a)

* **Name**: PrivateSubnet1
* **AZ**: us-east-1a
* **CIDR block**: 10.0.3.0/24

#### ****Private Subnet 2**** (AZ-b)

* **Name**: PrivateSubnet2
* **AZ**: us-east-1b
* **CIDR block**: 10.0.4.0/24

### ****Step 3: Create Internet Gateway****

1. Go to **Internet Gateways > Create**
   * Name: MyIGW
2. After creation, click **Attach to VPC** → Select MyVPC

### ****Step 4: Create Route Tables****

#### ****Public Route Table****

* Go to **Route Tables > Create Route Table**
  + Name: PublicRT
  + VPC: MyVPC
* Select it > Routes tab > Edit Routes
  + Add route: 0.0.0.0/0 → Target: Internet Gateway (MyIGW)
* Associate PublicSubnet1 and PublicSubnet2 with this route table.

#### ****Private Route Table**** (Optional step; default table works unless customizing further)

* No need to manually create one unless doing more advanced routing.

### ****Step 5: Create NAT Gateway (for Private Subnet access to internet)****

1. Go to **Elastic IPs > Allocate**
2. Go to **NAT Gateways > Create**
   * Name: MyNATGateway
   * Subnet: PublicSubnet1
   * Elastic IP: Choose the one allocated
3. Click **Create NAT Gateway**

Then update the **main/private route table**:

* Add route: 0.0.0.0/0 → Target: NAT Gateway

## 🧰 AWS CLI Version

bash

CopyEdit

# Create VPC

aws ec2 create-vpc --cidr-block 10.0.0.0/16 --tag-specifications \

"ResourceType=vpc,Tags=[{Key=Name,Value=MyVPC}]"

# Create subnets (AZs must match your region)

aws ec2 create-subnet --vpc-id vpc-xxxx --cidr-block 10.0.1.0/24 --availability-zone us-east-1a --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=PublicSubnet1}]'

aws ec2 create-subnet --vpc-id vpc-xxxx --cidr-block 10.0.2.0/24 --availability-zone us-east-1b --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=PublicSubnet2}]'

aws ec2 create-subnet --vpc-id vpc-xxxx --cidr-block 10.0.3.0/24 --availability-zone us-east-1a --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=PrivateSubnet1}]'

aws ec2 create-subnet --vpc-id vpc-xxxx --cidr-block 10.0.4.0/24 --availability-zone us-east-1b --t

## What Are Availability Zones?

* **AZs** are **isolated physical data centers** within an AWS Region.
* Each AWS Region (like us-east-1) has **multiple AZs** (like us-east-1a, us-east-1b, etc.).
* AZs are designed to be **independent**, with **separate power, cooling, and networking**.

## Why Use 2 Availability Zones?

You're using **two AZs** to improve **availability and reliability** of your infrastructure:

### ****Redundancy****

Each **subnet is placed in a different AZ**, so that:

* If **AZ-A** fails, **AZ-B** is still up.
* Your EC2 instances, load balancers, and NAT gateways can still operate.

### ****Auto Scaling Across Zones****

If you deploy EC2 instances with **Auto Scaling**, AWS can:

* Launch instances in **both AZs**, balancing load and reducing risk.

### ****Best Practice****

Using **at least 2 AZs** is considered an **AWS Well-Architected best practice** for production workloads.

Region (us-east-1)

┌────────────────────┐

│ VPC: MyVPC │

└────────────────────┘

│ │

▼ ▼

AZ-a (1a) AZ-b (1b)

┌──────────┐ ┌──────────┐

│ Public 1 │ │ Public 2 │ <- for ELB, NAT GW

└──────────┘ └──────────┘

┌──────────┐ ┌──────────┐

│ Private 1 │ Private 2 │ <- for app, DB (Data base)

└──────────┘ └──────────┘

**What is ELB?**

**ELB** stands for **Elastic Load Balancer** — a service from AWS that **automatically distributes incoming traffic** across multiple **EC2 instances**, containers, IPs, or Lambda functions.

**🔄 Purpose of ELB:**

Think of ELB as a **traffic manager**. When users connect to your app:

* ELB decides which backend instance should handle the request.
* It checks instance health and **only routes traffic to healthy instances**.
* It helps you **scale, balance, and protect** your app automatically.

**🧩 Types of Load Balancers in AWS ELB:**

| **Load Balancer Type** | **Use Case** |
| --- | --- |
| **Application Load Balancer (ALB)** | Best for HTTP/HTTPS traffic, layer 7 (URL path-based routing) |
| **Network Load Balancer (NLB)** | Best for high-performance TCP/UDP traffic, layer 4 (very fast) |
| **Gateway Load Balancer** | Used for deploying security appliances (like firewalls) |
| **Classic Load Balancer** (legacy) | Layer 4 & 7, but now outdated. Use ALB/NLB instead |

**📦 Example Use Case (Web App):**

Imagine you have 3 EC2 instances hosting your website:

* A user hits your **domain** (www.myapp.com)
* ELB receives the request and routes it to one of your EC2 instances
* If one instance is down, ELB skips it and sends traffic to the healthy ones

**🛡️ Bonus Features:**

* **SSL Termination** (HTTPS)
* **Auto Scaling Integration**
* **Health Checks**
* **Cross-Zone Load Balancing**
* **Sticky Sessions**

**📌 Real-World Scenario:**

In your setup:

* You launch **EC2 instances in 2 public subnets** (across 2 AZs).
* Place an **Application Load Balancer (ALB)** in front of them.
* ELB will **distribute traffic evenly** across both subnets/AZs.

Why IGW?

The **Internet Gateway (IGW)** plays a critical role in allowing your AWS resources (like EC2 instances) to **communicate with the internet**.

An **Internet Gateway (IGW)** is a **horizontal scaling, redundant AWS-managed gateway** that connects your **VPC to the internet**.

It allows:

* **Inbound traffic** from the internet to your **public-facing resources**
* **Outbound traffic** from your resources (like EC2) **to the internet**

## Why Do You Need an Internet Gateway?

| **Purpose** | **Explanation** |
| --- | --- |
| **Public Access to EC2** | Without an IGW, EC2 instances can’t be accessed using SSH, HTTP, etc., from outside AWS. |
| **Internet Communication** | EC2 instances won’t be able to download packages or updates from the internet. |
| **Route Table Integration** | IGW is the **target for 0.0.0.0/0** in route tables used by **public subnets**. |
| **Required for NAT Gateway** | Even a NAT Gateway (used for private subnets) needs IGW to reach the internet. |

Internet

│

┌────────────┐

│ IGW │ ← Internet Gateway

└────┬───────┘

│

┌───────▼────────┐

│ VPC │

└────────────────┘

│ │

PublicSubnet PrivateSubnet

(has IGW) (no direct IGW)

│

EC2 (public IP)

## What If You Don’t Use IGW?

Without IGW:

* EC2 in **public subnets** won’t have **internet access**, even if it has a public IP.
* No **inbound** or **outbound** traffic to/from the internet.
* You'll need **other networking solutions** like **VPNs**, **VPC Peering**, or **Direct Connect** for connectivity.

### ****IGW Per VPC Limit****

* AWS **allows only one IGW per VPC** — you physically **can't attach two IGWs to the same VPC**.
* That IGW handles all internet traffic for **every subnet marked as "public"** (i.e., those with a route to IGW).

### 3. ****Subnets Use Route Tables to Connect to IGW****

You define which subnets get internet access using **route tables**, not by creating more IGWs.

For example:

* Route table for public subnet:  
  0.0.0.0/0 → igw-abc123

This tells AWS: "Send all internet-bound traffic from this subnet to the IGW."

### 4. ****Cost-Efficient and Simplified Design****

* No need to manage or pay for multiple IGWs.
* Reduces complexity and is aligned with AWS **best practices**.

## Analogy

Think of a **VPC as a gated colony**, and the **IGW as the main gate**:

* You **only need one well-managed main gate** to enter/exit.
* Internal roads (subnets) connect to that gate via route maps (route tables).
* You don’t need a separate gate for every street.

Great question again — you’re really digging into the **core networking architecture of AWS**, which is key for mastering cloud infrastructure!

## 🔁 What is ****NAT****?

**NAT** stands for **Network Address Translation**.

In AWS, you typically use a **NAT Gateway** or **NAT Instance** to allow **private subnet instances** (like backend servers or databases) to:

✅ **Access the internet (outbound)**  
❌ **Without being exposed to the internet (inbound)**

## 🔍 Why Do You Need a NAT Gateway?

### 🔒 1. ****Private Subnet EC2s Can’t Reach Internet Directly****

* Private subnets **do not** have a route to the **Internet Gateway (IGW)**.
* So, EC2s in private subnets **cannot reach the internet directly**, even for:
  + Software updates (e.g., apt, yum, pip)
  + Docker pulls
  + External APIs

### ✅ 2. ****NAT Gateway Allows Outbound Internet****

* A NAT Gateway is placed in a **public subnet** and connected to the **IGW**.
* The private subnet sends traffic **to the NAT Gateway**, which forwards it to the internet.

The return traffic is then routed **back through the NAT Gateway to the private EC2**.

## 📦 Real-World Use Case

You have:

* Web servers in **public subnets**
* App servers or RDS in **private subnets**

Your **app server needs to download OS updates**, or your DB needs to sync something — but you **don’t want them publicly reachable**.

→ Use a **NAT Gateway**.

## 🔁 NAT Gateway Flow Diagram

scss

CopyEdit

Internet

│

┌───────────────┐

│ IGW │

└────┬──────────┘

│

┌───────▼─────────┐

│ Public Subnet │

│ (NAT Gateway) │

└──────┬──────────┘

│

┌──────▼───────────┐

│ Private Subnet │

│ (EC2 Instance) │

└──────────────────┘

## ⚠️ Without NAT Gateway?

* EC2 in private subnet can't:
  + Reach the internet
  + Download updates
  + Connect to external services

And you wouldn't want to give it a **public IP + IGW**, because that would expose it to the internet (defeating the point of a **private** subnet).

## 🧠 Summary

| **Question** | **Answer** |
| --- | --- |
| **Why use NAT Gateway?** | To allow **private subnet EC2s to access the internet** securely. |
| **What does it prevent?** | Inbound traffic from internet (so EC2s stay private). |
| **Where is it placed?** | In a **public subnet**, with a route from private subnets. |
| **Alternatives?** | NAT Instance (manual, less scalable). |

Would you like a step-by-step or CLI command to set up a NAT Gateway and configure your route tables?

##### You said:

explain me in a layman understaing about NAT

###### ChatGPT said:

Absolutely! Let’s break down **NAT (Network Address Translation)** in the **simplest, most relatable way** — like you’re explaining it to someone with zero tech background.

## 🧠 NAT Explained Like You’re 5

Imagine your house is inside a gated community:

* 🏠 **Your House (EC2 instance)** = Private Subnet
* 🛡️ **The Security Gate (NAT)** = NAT Gateway
* 🌍 **The Outside World (Internet)** = The Internet
* 🚧 **The Main Highway Gate (IGW)** = Internet Gateway

### 💡 The Rule in This Neighborhood:

You’re not allowed to talk to strangers directly (you’re in a private subnet). But sometimes you need to:

* Order pizza online 🍕
* Get updates from the weather app 🌦️
* Watch YouTube videos 🎥

### 🧳 So What Do You Do?

You ask the **security guard (NAT Gateway)** to do it **for you**.

You say:

“Hey, can you please order me a pizza from Domino’s?”

The **NAT gateway** walks out through the **main gate (IGW)**, places your order, and when the pizza guy comes back, the NAT delivers it to your door.

✅ You got what you needed  
❌ But no one from outside ever saw **you directly**

### 📦 Key Points in This Analogy

| **Concept** | **Real-Life Meaning** |
| --- | --- |
| Your house | EC2 in private subnet |
| You | Your app, DB, or backend server |
| Security Guard | NAT Gateway |
| Gated Community | VPC |
| Main Gate | Internet Gateway (IGW) |
| Pizza guy / Website | Internet services |

### 🔐 What You Achieve:

* You get **outbound access** (to update software, download files).
* The world **can’t knock on your door** (no inbound access).
* You stay **private and secure**, but still **get your work done**.

### 🚫 Without NAT?

It’s like living in a house with **no way to talk to the outside world**, no delivery guy, no calls, no messages. You're cut off.

## ✅ Summary (Layman Style)

**"NAT is like a bodyguard or middleman. It lets your private servers safely talk to the internet — but keeps them hidden from the world."**

## NAT Gateway Options:

| **Option Selected** | **Meaning** |
| --- | --- |
| **None** (currently selected) | Private subnets will **not have internet access**. This is fine if you don’t need to download packages or update software from those instances. |

### ✅ Recommended Option (for real-world use):

* Choose **"In 1 AZ"** to create **1 NAT Gateway** in one public subnet:
  + This allows **both private subnets** to access the internet.
  + **Saves cost** compared to “1 per AZ” (which creates 2 NAT Gateways).

## 💡 Quick Notes:

| **Setting** | **Meaning** |
| --- | --- |
| **"None" NAT Gateway** | Private subnets have **no outbound internet** access |
| **"In 1 AZ"** NAT | Recommended for most use cases: secure + cost-effective |
| **"1 per AZ"** NAT | High-availability setup, more expensive |

## 📦 VPC Endpoints – S3 Gateway:

You've selected **S3 Gateway endpoint**, which is a smart move:

* Allows EC2 in **private subnets** to access **Amazon S3** **without NAT** or internet.
* Reduces **NAT Gateway data transfer costs**.
* Ideal for private apps that interact with S3 (e.g., backups, file uploads, etc.)

## ✅ Final Thoughts

You're setting up a **textbook best-practice VPC**:

| **Component** | **Setup** |
| --- | --- |
| Availability Zones | 2 (high availability) |
| Public Subnets | 2 (web, NAT, load balancer) |
| Private Subnets | 2 (app servers, DBs) |
| NAT Gateway | Suggest selecting "In 1 AZ" |
| S3 Gateway Endpoint | ✅ Good choice for cost + security |

Let me know if you'd like help reviewing the **CIDR blocks** next, or configuring **route tables/NAT access** after VPC creation!

Ask ChatGPT

Top of Form



Bottom of Form