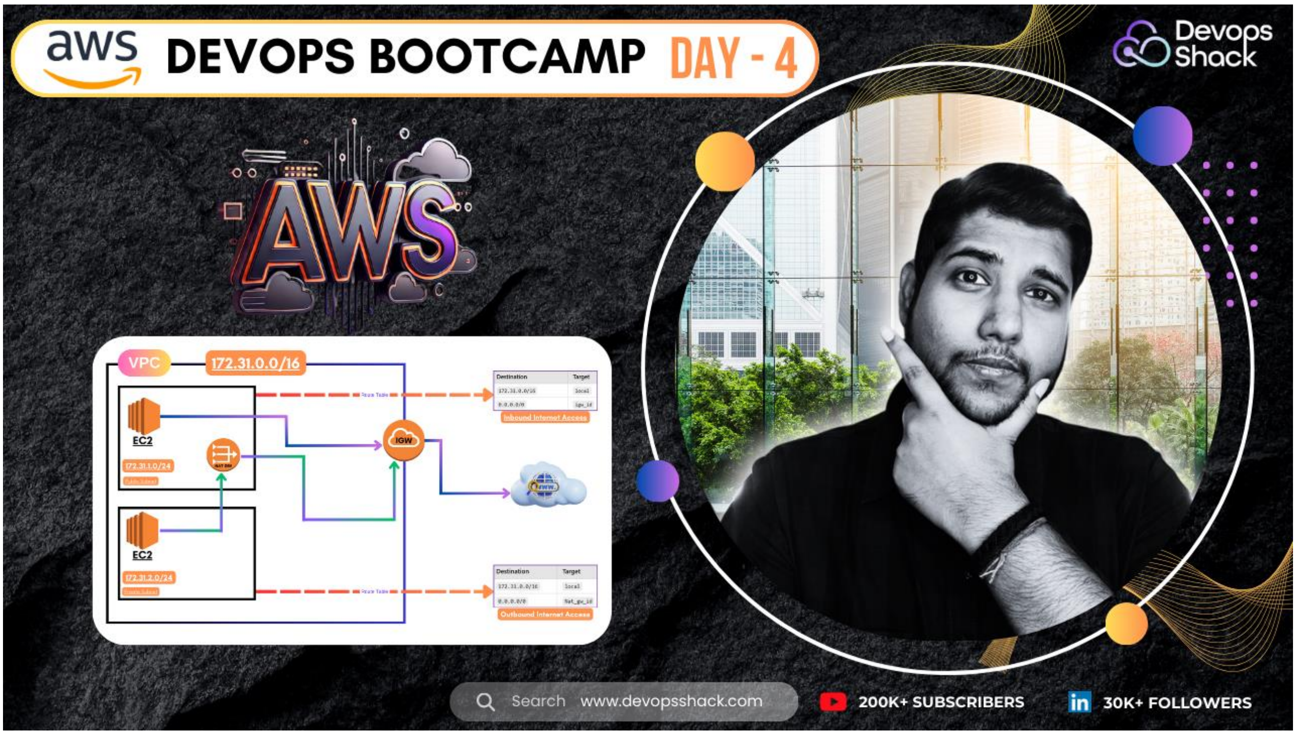
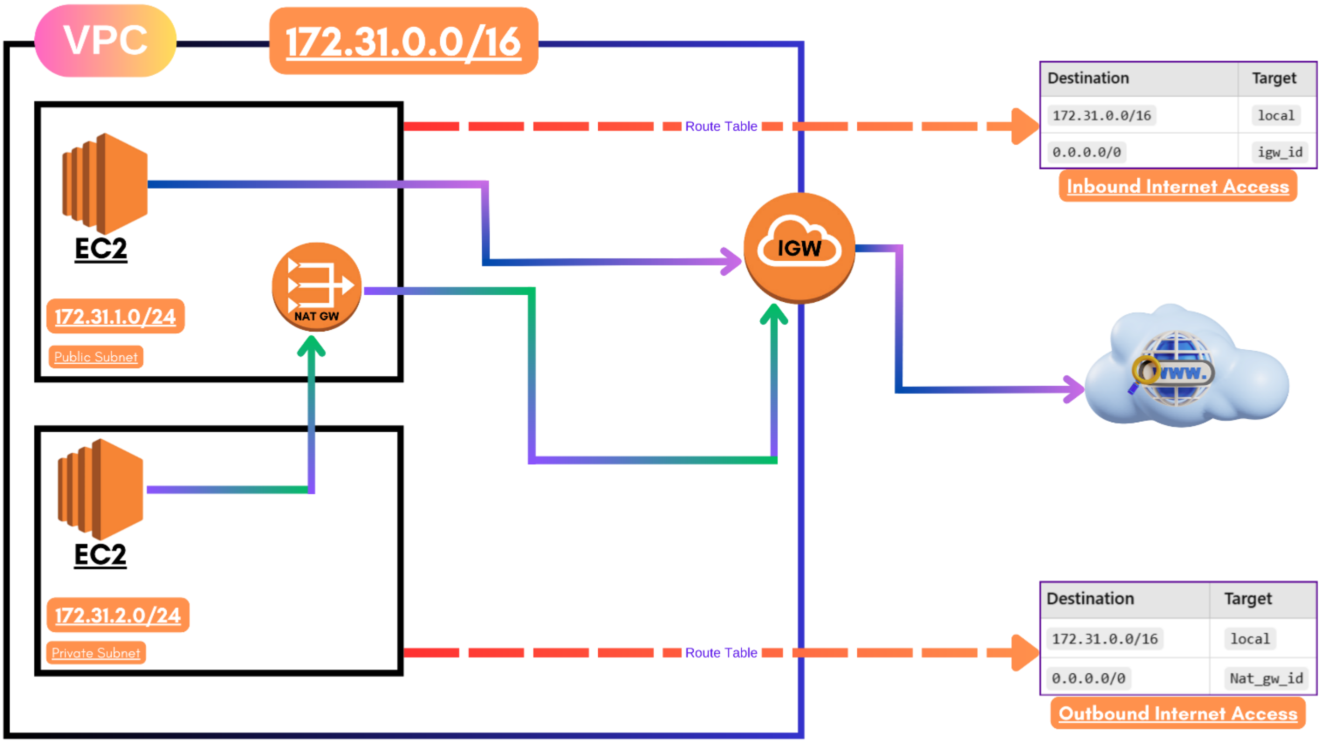
DevOps Shack

Complete Guide AWS Networking





Destination	Target
172.31.0.0/16	local
0.0.0.0/0	igw_id

Inbound Internet Access

Destination	Target
172.31.0.0/16	local
0.0.0.0/0	Nat_gw_id

Outbound Internet Access

AWS Networking serves as the foundation for enabling communication between resources in the AWS cloud and external systems. This section focuses on creating secure, scalable, and highly available network architectures within AWS.

What is AWS Networking?

AWS Networking involves the creation and management of network components in the AWS Cloud. It includes services like **Virtual Private Cloud (VPC)**, **Subnets**, **Internet Gateways**, **NAT Gateways**, **Security Groups**, **Route Tables**, and other networking tools to build private, secure, and scalable environments.

Benefits of AWS Networking

1. **Isolation**: AWS provides logically isolated virtual networks with VPC.
2. **Scalability**: Easily expand or modify network components as your needs grow.
3. **Security**: Use Security Groups and Network ACLs to control traffic in and out of resources.
4. **Flexibility**: Create custom network topologies to meet application needs.
5. **Cost Efficiency**: Only pay for resources used, such as NAT Gateways and Elastic IPs.

Core AWS Networking Components

1. **VPC (Virtual Private Cloud)**:
   - A private, isolated virtual network within AWS.
   - Allows you to define IP ranges, subnets, and routing rules.
   - Acts as a container for other networking components.
2. **Subnets**:
   - Logical subdivisions within a VPC.
   - Enable segregation of resources into **public** or **private** zones.
3. **Route Tables**:
   - Define how network traffic is directed between subnets, gateways, and external resources.
4. **Internet Gateway (IGW)**:
   - Enables internet access for resources in public subnets.
5. **NAT Gateway**:
   - Allows private subnet resources to access the internet for outbound traffic while blocking inbound internet access.
6. **Security Groups (SGs)**:
   - Act as virtual firewalls to control inbound and outbound traffic for individual resources.

7. **Network ACLs (NACLs)**:
    - Provide an additional layer of security by controlling traffic at the subnet level.
8. **Elastic IPs (EIPs)**:
    - Static, public IP addresses that can be reassigned across AWS resources.

AWS Networking Use Cases

1. **Multi-Tier Architectures**:
    - Separate web servers in public subnets and databases in private subnets.
2. **Hybrid Networking**:
    - Connect on-premises data centers with AWS using VPNs or Direct Connect.
3. **Load Balancing and High Availability**:
    - Use Elastic Load Balancers and multi-AZ deployment for redundancy.
4. **Private Communication**:
    - Use VPC Peering or AWS Transit Gateway for communication between multiple VPCs.

High-Level Workflow for AWS Networking

1. **Create a VPC**: Define your isolated virtual network and its CIDR range.
2. **Add Subnets**: Divide the VPC into public and private subnets.
3. **Attach Gateways**: Add Internet Gateway for public subnets and NAT Gateway for private subnets.
4. **Configure Route Tables**: Direct traffic between subnets and external networks.
5. **Set Up Security Groups**: Apply access rules for resources like EC2, RDS, etc.
6. **Test Connectivity**: Verify the setup by testing internal and external traffic flows.

Example AWS Network Architecture

```
VPC: 10.0.0.0/16
├── Public Subnet: 10.0.1.0/24
|   ├── Internet Gateway (IGW)
|   ├── EC2 Instance (Web Server)
|   └── Route Table:
|       - 10.0.0.0/16 → Local
|       - 0.0.0.0/0 → IGW
|
└── Private Subnet: 10.0.2.0/24
```

```
├── NAT Gateway
├── EC2 Instance (Database)
└── Route Table:
  - 10.0.0.0/16 → Local
  - 0.0.0.0/0 → NAT Gateway
```

Best Practices for AWS Networking

1. **Use VPCs for Isolation**:
   - Create separate VPCs for Dev, Test, and Production environments.
2. **Least Privilege Access**:
   - Restrict inbound and outbound traffic to only what's necessary using SGs and NACLs.
3. **Enable Logging**:
   - Use VPC Flow Logs to monitor network traffic.
4. **Monitor Cost**:
   - Optimize NAT Gateway placement to reduce data transfer costs.
5. **Secure Network Boundaries**:
   - Use AWS WAF, Shield, and Firewall Manager for additional protection.

AWS Networking Tools

1. **AWS Direct Connect**: High-speed, dedicated connection between AWS and on-premises.
2. **VPC Peering**: Connect two VPCs privately.
3. **AWS Transit Gateway**: Hub-and-spoke model to connect multiple VPCs and on-premises networks.
4. **Elastic Load Balancer (ELB)**: Distribute traffic across instances for high availability.

Example Scenario

You need to set up a 3-tier web application:

1. Web servers in the **public subnet** with internet access via IGW.
2. Application servers in the **private subnet** communicating with the database.
3. The private subnet accesses updates from the internet via a NAT Gateway.

This setup isolates sensitive data (e.g., databases) while ensuring internet connectivity for essential resources.

1. IP Addresses, CIDR, and Subnetting

Understanding IP addresses, CIDR notation, and subnetting is critical to designing and managing AWS networking effectively.

2.1 IP Address

An IP (Internet Protocol) address is a unique numerical identifier assigned to each device on a network. It allows devices to communicate with each other over a network.

Types of IP Addresses in AWS

1. **Public IP Address**:
   - Assigned to resources in public subnets.
   - Enables direct communication with the internet.
   - Automatically assigned or can be replaced with an Elastic IP (static).
   - Example: 54.233.129.45.
2. **Private IP Address**:
   - Used for internal communication within a VPC or between resources in private subnets.
   - Cannot access the internet directly.
   - Example: 10.0.2.15.
3. **Elastic IP Address (EIP)**:
   - A static, public IPv4 address you can allocate and attach to a resource.
   - Ideal for instances that need persistent public access.
4. **IPv6 Address**:
   - Optional, globally unique addresses with a larger address space than IPv4.
   - Example: 2600:1f18:1234:abcd::1.

2.2 CIDR (Classless Inter-Domain Routing)

CIDR is a method to define IP address ranges in a network using a combination of:

- **Base IP Address**: The starting address of the range (e.g., 10.0.0.0).
- **Subnet Mask**: A suffix (/16, /24, etc.) indicating how many IPs are in the range.

CIDR Range and Subnet Mask

CIDR Block	Subnet Mask	Total IPs	Example Range
10.0.0.0/16	255.255.0.0	65,536	10.0.0.0 - 10.0.255.255
10.0.1.0/24	255.255.255.0	256	10.0.1.0 - 10.0.1.255
10.0.1.128/25	255.255.255.128	128	10.0.1.128 - 10.0.1.255

AWS VPC CIDR Block Rules

- The CIDR block for a VPC must be between /16 and /28.
- Example: 10.0.0.0/16.

2.3 Subnetting

Subnetting divides a larger IP address range (CIDR block) into smaller, manageable networks called subnets. Each subnet can isolate or organize resources in a specific part of your network.

Why Subnetting is Important

1. **Organizational Clarity**: Helps manage resources more efficiently.
2. **Traffic Segmentation**: Public vs. private subnets for better security and performance.
3. **IP Allocation**: Optimize the usage of IP addresses.

Subnetting Example

Parent CIDR Block: 10.0.0.0/16

- Total IPs: 65,536 (reserved IPs: AWS takes 5 per subnet).

Divide into Subnets

Subnet Name	CIDR Block	IP Addresses	Usage
Public Subnet 1	10.0.1.0/24	256	Web servers with internet
Private Subnet 1	10.0.2.0/24	256	Databases or apps

Subnet Reservation

For each subnet, AWS reserves 5 IP addresses for internal use:

- **First IP**: Network address.
- **Second IP**: Reserved by AWS.
- **Third IP**: Reserved by AWS.
- **Last IP**: Broadcast address.

For a /24 block:

- 10.0.1.0: Network address.
- 10.0.1.1: Reserved.
- 10.0.1.2: Reserved.
- 10.0.1.3: Reserved.
- 10.0.1.255: Broadcast address.

How AWS Uses IPs

- Resources (EC2, RDS, etc.) are assigned private IPs from the subnet's CIDR.
- Public IPs are assigned to internet-facing resources in public subnets.

Subnet Planning in AWS

1. Plan the CIDR block for your VPC based on your expected workload.
2. Divide the VPC into public and private subnets:
   - Public Subnets: Allocate IPs for internet-facing resources.
   - Private Subnets: Allocate IPs for internal resources like databases.

Textual Diagram for Subnetting

```
VPC: 10.0.0.0/16
├── Public Subnet (10.0.1.0/24)
|   ├── Web Server (10.0.1.5)
|   ├── NAT Gateway (10.0.1.10)
|
└── Private Subnet (10.0.2.0/24)
    ├── Database Server (10.0.2.5)
    ├── Application Server (10.0.2.10)
```

Example with AWS CLI

Create a VPC:

```
aws ec2 create-vpc --cidr-block 10.0.0.0/16 --region us-east-1
```

Create Subnets:

```
aws ec2 create-subnet --vpc-id vpc-abc123 --cidr-block 10.0.1.0/24 --region us-east-1
aws ec2 create-subnet --vpc-id vpc-abc123 --cidr-block 10.0.2.0/24 --region us-east-1
```

Key Takeaways

- Use /16 for large VPCs with multiple subnets.
- Divide the VPC CIDR block into smaller subnets to segregate workloads.
- Keep IP reservations in mind when planning subnet sizes.

2. AWS Virtual Private Cloud (VPC)

What is a VPC?

A **Virtual Private Cloud (VPC)** is a logically isolated network in AWS where you can define your custom IP address range, create subnets, and configure routing to meet your networking requirements.

VPCs are the backbone of AWS networking, enabling you to securely host and manage resources like EC2 instances, RDS databases, and application services.

Key Features of VPC

1. **Logical Isolation**:
    - Each VPC is isolated from other VPCs and AWS accounts.
    - You control the IP range, routing, and access.
2. **Customizable Networking**:
    - Define your CIDR block.
    - Subdivide the network into **public** and **private subnets**.
3. **Secure Communication**:
    - Use **Security Groups** and **Network ACLs** to control traffic.
    - Establish secure connections to on-premises networks via VPN or AWS Direct Connect.
4. **Hybrid Networking**:
    - Connect on-premises networks to AWS using **VPN** or **AWS Direct Connect**.
5. **Scalability and Flexibility**:
    - Add or remove subnets, gateways, and route tables as your requirements evolve.

VPC Components

1. **CIDR Block**:
    - The IP range assigned to the VPC.
    - Example: 10.0.0.0/16 (65,536 IP addresses).
2. **Subnets**:
    - Logical divisions within the VPC, either public or private.
3. **Internet Gateway (IGW)**:
    - Connects the VPC to the internet.
4. **NAT Gateway**:
    - Allows private subnets to initiate outbound internet connections.
5. **Route Tables**:
    - Direct traffic between subnets, internet gateways, and NAT gateways.
6. **Security Groups (SG)**:
    - Control inbound and outbound traffic at the instance level.
7. **Network ACLs (NACL)**:

  - Control inbound and outbound traffic at the subnet level.

Steps to Create a VPC

1. **Go to the AWS Management Console**:
   - Navigate to **VPC** → **Create VPC**.
2. **Define VPC Details**:
   - **Name:** My-VPC.
   - **IPv4 CIDR Block:** 10.0.0.0/16 (or your desired range).
   - **IPv6 CIDR Block (Optional):** Enable if you require IPv6 addresses.
   - **Tenancy:** Default (shared hardware) or Dedicated.
3. **Configure Additional Options** (if required):
   - Enable **DNS hostnames** and **DNS resolution** for better accessibility.
4. **Click Create**:
   - The new VPC is now available for use.

Example VPC Setup

CIDR Block

- **VPC CIDR Block:** 10.0.0.0/16 (Total IPs: 65,536).
- **Subnet CIDR Blocks**:
  - Public Subnet: 10.0.1.0/24.
  - Private Subnet: 10.0.2.0/24.

Logical Architecture

```
VPC: 10.0.0.0/16
├── Public Subnet (10.0.1.0/24)
|   ├── Internet Gateway (IGW)
|   ├── EC2 Instance (Web Server)
|
└── Private Subnet (10.0.2.0/24)
    ├── NAT Gateway
    ├── EC2 Instance (Database)
```

Routing in a VPC

A **Route Table** determines how traffic flows in a VPC.

- **Default Local Route**:
  - Enables communication between all subnets in the VPC.
  - Example: 10.0.0.0/16 → Local.
- **Internet Gateway Route**:
  - Allows internet access for public subnets.
  - Example: 0.0.0.0/0 → Internet Gateway (IGW).
- **NAT Gateway Route**:
  - Allows private subnets to access the internet for outbound traffic.
  - Example: 0.0.0.0/0 → NAT Gateway.

VPC Use Cases

1. **Web Applications**:
   - Host web servers in public subnets and databases in private subnets.
2. **Hybrid Cloud**:
   - Connect on-premises data centers with AWS using VPN or AWS Direct Connect.
3. **Secure Networking**:
   - Isolate environments like Dev, Test, and Prod using separate VPCs.

Best Practices for VPC

1. **Use Multiple Subnets**:
   - Divide your network into public and private subnets for better resource management.
2. **Implement Least Privilege Access**:
   - Use Security Groups and NACLs to allow only required traffic.
3. **Enable VPC Flow Logs**:
   - Monitor and analyze network traffic for troubleshooting and compliance.
4. **Plan CIDR Blocks**:
   - Use /16 CIDR for the VPC to allow scalability, and divide it into /24 or smaller subnets.
5. **Use Multiple Availability Zones**:
   - Deploy resources in multiple AZs for high availability.

AWS CLI Commands for VPC

Create a VPC

```
aws ec2 create-vpc --cidr-block 10.0.0.0/16 --region us-east-1
```

Enable DNS Hostnames

```
aws ec2 modify-vpc-attribute --vpc-id vpc-12345678 --enable-dns-hostnames
```

Diagram: VPC and Subnets

```
VPC: 10.0.0.0/16
├── Public Subnet (10.0.1.0/24)
|   ├── Web Server (10.0.1.5)
|   ├── NAT Gateway (10.0.1.10)
|
└── Private Subnet (10.0.2.0/24)
    ├── Database Server (10.0.2.5)
```

4. Route Tables

What is a Route Table?

A **Route Table** in AWS contains a set of rules, called **routes**, that determine where network traffic is directed within a VPC. Each subnet in a VPC is associated with a route table that guides traffic to its destination.

Types of Routes

1. **Local Routes**:
   - Automatically created for all VPCs.
   - Enable communication between resources within the VPC.
   - Example: Destination: 10.0.0.0/16 → Target: Local.
2. **Custom Routes**:
   - Defined by you to direct traffic to specific targets.
   - Examples:
     - Route to the internet via an **Internet Gateway (IGW)**.
     - Route to external networks via a **NAT Gateway**.
     - Route to other VPCs via **VPC Peering**.

Key Components of a Route Table

1. **Destination**: The IP address range of the traffic (e.g., 0.0.0.0/0 for all traffic).
2. **Target**: The resource that will handle the traffic (e.g., IGW, NAT Gateway, Local).
3. **Subnet Association**: Subnets associated with the route table.

Default and Custom Route Tables

Default Route Table

- Every VPC comes with a main route table by default.
- Includes a local route for communication within the VPC:
- Destination: 10.0.0.0/16 → Target: Local

Custom Route Table

- Manually created route tables for specific use cases (e.g., public and private subnets).

Steps to Create and Configure Route Tables

1. Create a Route Table

1. Go to the **VPC Console** → **Route Tables** → **Create Route Table**.
2. Enter details:
   - **Name:** Public-Route-Table.
   - **VPC:** Select the VPC.
3. Click **Create**.

2. Add Routes

1. **For Public Subnets**:
   - Add a route to the Internet Gateway (IGW) for outbound internet traffic:
   - Destination: 0.0.0.0/0
   - Target: Internet Gateway (igw-123456)
2. **For Private Subnets**:
   - Add a route to the NAT Gateway for outbound traffic:
   - Destination: 0.0.0.0/0
   - Target: NAT Gateway (nat-123456)
3. **Local Routes** (Default):
   - Automatically included for communication within the VPC:
   - Destination: 10.0.0.0/16
   - Target: Local

3. Associate Subnets with Route Tables

- Go to **Route Tables** → Select the route table.
- Click **Subnet Associations** → **Edit Subnet Associations**.
- Select the subnet(s) to associate with the route table.

Example:

- **Public Subnet**: Associated with a route table that directs traffic to the IGW.
- **Private Subnet**: Associated with a route table that directs traffic to the NAT Gateway.

Example Route Table Configuration

Public Route Table

- Associated with the **public subnet**.
- Routes:
- Destination: 10.0.0.0/16 → Target: Local
- Destination: 0.0.0.0/0 → Target: Internet Gateway (igw-abc123)

Private Route Table

- Associated with the **private subnet**.
- Routes:
- Destination: 10.0.0.0/16 → Target: Local
- Destination: 0.0.0.0/0 → Target: NAT Gateway (nat-xyz456)

Route Table and Traffic Flow

1. **Public Subnet Traffic Flow**:
   - An EC2 instance in a public subnet communicates with the internet via the IGW.
   - Traffic path:
     - **Public EC2 → IGW → Internet**.
2. **Private Subnet Traffic Flow**:
   - An EC2 instance in a private subnet accesses the internet via the NAT Gateway.
   - Traffic path:
     - **Private EC2 → NAT Gateway → IGW → Internet**.
3. **Local VPC Traffic Flow**:
   - Communication between instances in the same VPC (e.g., between subnets).
   - Traffic path:
     - **Private EC2 → Local Route → Public EC2**.

Testing Route Tables

1. **Public Subnet**:
   - Launch an EC2 instance in the public subnet.
   - Test internet connectivity:

ping google.com

2. **Private Subnet**:
   - Launch an EC2 instance in the private subnet.
   - Use a Bastion Host or NAT Gateway to access the internet:

curl http://example.com

Best Practices for Route Tables

1. **Isolate Traffic**:
   - Use separate route tables for public and private subnets.
2. **Secure Private Resources**:
   - Ensure private subnets only allow outbound internet access via NAT Gateways.
3. **Use Meaningful Names**:
   - Name route tables clearly (e.g., Public-RT, Private-RT).
4. **Enable VPC Flow Logs**:
   - Monitor and analyze traffic for troubleshooting.

Example Textual Diagram

```
Route Table: Public-RT
├── Destination: 10.0.0.0/16 → Target: Local
├── Destination: 0.0.0.0/0   → Target: Internet Gateway


Route Table: Private-RT
├── Destination: 10.0.0.0/16 → Target: Local
├── Destination: 0.0.0.0/0   → Target: NAT Gateway
```

5. Internet Gateway (IGW)

What is an Internet Gateway?

An **Internet Gateway (IGW)** is a highly available, horizontally scaled component that allows resources in a **VPC** (specifically, public subnets) to communicate with the internet.

It serves two main purposes:

1. Enables outbound internet connectivity for public subnet resources.
2. Allows inbound traffic from the internet to resources with public IPs.

Key Features of an Internet Gateway

1. **Scalable and Highly Available**:

- Fully managed by AWS and scales automatically to handle traffic demands.

2. **One-to-One NAT for Public IPs**:
   - Maps a public IP address to a private IP address in your VPC.
3. **No Bandwidth Limits**:
   - Handles traffic without limits or additional costs (charges are based on data transfer).
4. **VPC Association**:
   - Each VPC can have only one Internet Gateway attached.
5. **No Cost for IGW**:
   - The Internet Gateway itself is free; charges only apply for data transfer.

When Do You Need an Internet Gateway?

- If your EC2 instances in a public subnet require:
  i. Outbound internet access (e.g., downloading updates or accessing APIs).
  ii. Inbound traffic from the internet (e.g., web servers or APIs).

Steps to Create and Attach an Internet Gateway

1. Create an Internet Gateway

1. Go to the **AWS Management Console** → **VPC Dashboard** → **Internet Gateways**.
2. Click **Create Internet Gateway**.
3. Enter a **Name** (e.g., Demo-IGW).
4. Click **Create Internet Gateway**.

2. Attach the Internet Gateway to a VPC

1. Select the newly created IGW (e.g., Demo-IGW).
2. Click **Actions** → **Attach to VPC**.
3. Choose the **VPC** you want to attach it to and click **Attach**.

3. Update the Public Route Table

For resources in a public subnet to access the internet, you must update the associated route table.

1. Go to **VPC Dashboard** → **Route Tables**.
2. Select the **Public Route Table**.
3. Add a new route:
   - **Destination:** 0.0.0.0/0
   - **Target:** Internet Gateway (e.g., igw-abc123).

Traffic Flow with an Internet Gateway

1. Outbound traffic from public instances:
   - EC2 instance → Route Table → IGW → Internet.
2. Inbound traffic from the internet:
   - Internet → IGW → Route Table → Public Subnet → EC2 instance.

Example Internet Gateway Configuration

VPC CIDR Block: 10.0.0.0/16

- Public Subnet CIDR: 10.0.1.0/24.
- Private Subnet CIDR: 10.0.2.0/24.

Public Route Table

Destination	Target	Description
10.0.0.0/16	Local	Intra-VPC communication
0.0.0.0/0	Internet Gateway	Internet access for public subnets

Testing Internet Gateway

1. Launch an EC2 Instance in a Public Subnet

- Create an instance in the public subnet with a public IP.
- Ensure the instance's **Security Group** allows outbound traffic to port 80 (HTTP) or 443 (HTTPS).

2. Test Internet Access

- SSH into the instance and run:
- ping google.com

curl http://example.com

Best Practices for Using an Internet Gateway

1. **Restrict Inbound Access**:
   - Use Security Groups to allow only necessary ports (e.g., 22 for SSH, 80 for HTTP).
2. **Combine with NAT Gateway**:
   - Use an IGW with NAT Gateway to provide secure outbound internet access for private subnets.
3. **Enable VPC Flow Logs**:
   - Monitor and log all traffic through the IGW for troubleshooting and auditing.

4. **Assign Elastic IPs**:
   - Use Elastic IPs for persistent public IPs that are attached to instances.

Example Textual Diagram

```
VPC: 10.0.0.0/16
├── Public Subnet (10.0.1.0/24)
|   ├── Internet Gateway (IGW)
|   ├── EC2 Instance with Public IP (54.233.129.45)
|
└── Private Subnet (10.0.2.0/24)
    ├── No direct internet access
```

Common Issues with Internet Gateways

1. **No Internet Access**:
   - Ensure the route table for the public subnet includes a route to the IGW.
   - Verify that the EC2 instance has a public IP.
2. **Improper Security Group Rules**:
   - Ensure inbound and outbound rules allow necessary traffic (e.g., HTTP, HTTPS, SSH).
3. **Missing IGW Association**:
   - Verify that the IGW is attached to the correct VPC.

6. NAT Gateway

A **NAT Gateway** (Network Address Translation Gateway) enables resources in a **private subnet** to access the internet for outbound traffic, such as downloading updates or communicating with external APIs, without exposing these resources to incoming traffic from the internet.

Key Features of NAT Gateway

1. **Outbound Internet Access**:
   - Provides internet access for private resources.
   - Blocks inbound internet traffic for security.
2. **Scalability**:
   - Managed by AWS, automatically scales to handle high traffic.
3. **High Availability**:
   - Redundant within an Availability Zone (AZ).
   - Use multiple NAT Gateways in different AZs for fault tolerance.
4. **Elastic IP**:

  - Associated with a public Elastic IP address, translating private IPs to a public IP for internet access.

When Do You Need a NAT Gateway?

- Use a NAT Gateway when resources in a **private subnet** need:
  i. Outbound internet access (e.g., application updates, external APIs).
  ii. Enhanced security by preventing direct exposure to the internet.

How a NAT Gateway Works

1. Outbound traffic from private instances is routed to the NAT Gateway.
2. The NAT Gateway translates the private IPs to the public Elastic IP.
3. The NAT Gateway forwards the traffic to the Internet Gateway (IGW).
4. Replies from the internet are routed back to the NAT Gateway, which then forwards them to the private instance.

Steps to Create and Use a NAT Gateway

1. Create a NAT Gateway

1. Go to **VPC Dashboard** → **NAT Gateways** → **Create NAT Gateway**.
2. Configure:
   - **Name**: Demo-NAT.
   - **Subnet**: Select a **public subnet** (e.g., 10.0.1.0/24).
   - **Elastic IP**: Allocate a new Elastic IP or use an existing one.
3. Click **Create NAT Gateway**.

2. Update the Private Route Table

1. Navigate to **Route Tables** in the **VPC Console**.
2. Select the **Route Table** associated with your private subnet.
3. Add a new route:
   - **Destination**: 0.0.0.0/0.
   - **Target**: NAT Gateway (e.g., nat-123456).
4. Save the changes.

Example Configuration

VPC CIDR Block: 10.0.0.0/16

Subnet Type	CIDR Block	Route Table Destination	Target
Public Subnet	10.0.1.0/24	0.0.0.0/0	Internet Gateway (IGW)
Private Subnet	10.0.2.0/24	0.0.0.0/0	NAT Gateway (nat-123456)

Testing NAT Gateway

1. Launch an Instance in the Private Subnet

1. Create an EC2 instance in the private subnet.
2. Ensure the instance has:
   - No public IP address.
   - A security group allowing outbound traffic on required ports (e.g., HTTP/HTTPS).

2. Test Internet Connectivity

1. Use a Bastion Host (in the public subnet) to SSH into the private instance.
2. From the private instance, test internet connectivity:
3. curl http://example.com

ping 8.8.8.8

High Availability with NAT Gateway

- A NAT Gateway is AZ-specific. To ensure high availability:
  i. Deploy NAT Gateways in multiple Availability Zones.
  ii. Associate the subnets in each AZ with the corresponding NAT Gateway via route tables.

Advantages of NAT Gateway

1. **Security**:
   - Private instances remain hidden from the internet.
2. **Ease of Use**:
   - Fully managed by AWS, with no configuration overhead.
3. **Performance**:
   - Automatically scales to handle traffic.
4. **Cost-Effective**:
   - Charged based on usage (hours and data transfer).

NAT Gateway vs NAT Instance

Feature	NAT Gateway	NAT Instance
Management	Fully managed by AWS	Requires manual setup
Scalability	Auto-scales	Fixed instance capacity
High Availability	Built-in	Requires manual setup
Performance	High	Limited by instance type
Cost	Higher	Lower

Common Issues with NAT Gateway

1. **No Internet Access**:
   - Ensure the private route table has a route to the NAT Gateway.
   - Verify that the NAT Gateway is deployed in a public subnet with an associated IGW.
2. **Incorrect Security Group Rules**:
   - Ensure the security group allows outbound traffic for the required ports (e.g., 80, 443).
3. **Single Point of Failure**:
   - Use multiple NAT Gateways across Availability Zones to avoid downtime.

Example Textual Diagram

```
VPC: 10.0.0.0/16
├── Public Subnet (10.0.1.0/24)
|   ├── Internet Gateway (IGW)
|   ├── NAT Gateway (Elastic IP)
|
└── Private Subnet (10.0.2.0/24)
    ├── EC2 Instance (Database)
    ├── Outbound Traffic → NAT Gateway → IGW → Internet
```

7. Network Security Group (NSG)

What is a Network Security Group (NSG)?

In AWS, a **Security Group (SG)** is analogous to a **Network Security Group (NSG)** in other cloud environments. It acts as a virtual firewall, controlling inbound and outbound traffic at the instance level. Security Groups are stateful, meaning that if you allow inbound traffic, the corresponding outbound response is automatically allowed.

Key Features of Security Groups

1. **Instance-Level Control**:
   - Security Groups are attached to individual resources (e.g., EC2 instances, RDS databases).
2. **Stateful Nature**:
   - Inbound rules automatically allow corresponding outbound responses and vice versa.
3. **Fine-Grained Traffic Control**:
   - Define rules based on protocol (TCP/UDP), port numbers, and source/destination IP ranges.
4. **Flexible Configuration**:
   - Attach multiple Security Groups to a single resource.
   - Modify rules dynamically without restarting resources.

Components of a Security Group

1. **Inbound Rules**:
   - Define the type of traffic allowed **into** the resource.
   - Example: Allow SSH (port 22) from a specific IP.
2. **Outbound Rules**:
   - Define the type of traffic allowed **out** of the resource.
   - Example: Allow all outbound traffic to the internet.
3. **Source/Destination**:
   - Specify the source (for inbound) or destination (for outbound).
   - Can be an IP range (e.g., 203.0.113.0/32) or another Security Group.

Default Security Group Behavior

1. **Inbound Traffic**:
   - All inbound traffic is denied by default.
2. **Outbound Traffic**:
   - All outbound traffic is allowed by default.

Steps to Create and Configure a Security Group

1. Create a Security Group

1. Go to the **AWS Management Console** → **EC2 Dashboard** → **Security Groups**.
2. Click **Create Security Group**.

3. Configure:
    - **Name:** Web-Server-SG.
    - **VPC:** Select the appropriate VPC.
4. Add rules for inbound and outbound traffic (see examples below).

2. Add Inbound and Outbound Rules

Example for a Web Server:

- **Inbound Rules**:

Protocol	Port Range	Source	Description
HTTP	80	0.0.0.0/0	Allow web traffic
HTTPS	443	0.0.0.0/0	Allow secure traffic
SSH	22	203.0.113.0/32	Allow admin access

- **Outbound Rules** (Default: Allow All):

Protocol	Port Range	Destination	Description
All	All	0.0.0.0/0	Allow outbound traffic

3. Attach Security Group to Resources

- During the launch of an EC2 instance:
    - Select the Security Group from the "Configure Security Group" step.
- For existing instances:
    - Navigate to the EC2 instance → Actions → Networking → Change Security Groups.

Example Security Group Configurations

Public EC2 Instance (Web Server)

- **Purpose**: Allow HTTP/HTTPS access from the internet.
- **Inbound Rules**:
- HTTP: Port 80, Source: 0.0.0.0/0
- HTTPS: Port 443, Source: 0.0.0.0/0
- SSH: Port 22, Source: 203.0.113.0/32
- **Outbound Rules** (Default):
- Allow all outbound traffic.

Private EC2 Instance (Database Server)

- **Purpose**: Allow traffic only from the application layer.
- **Inbound Rules**:
- MySQL: Port 3306, Source: Web-Server-SG
- **Outbound Rules**:
- Allow all outbound traffic.

Textual Diagram for Security Group

Public EC2 Instance (Web Server)

├── Inbound Rules:

| ├── HTTP: Port 80, Source: 0.0.0.0/0

| ├── HTTPS: Port 443, Source: 0.0.0.0/0

| ├── SSH: Port 22, Source: 203.0.113.0/32

├── Outbound Rules:

| ├── All Traffic: Destination: 0.0.0.0/0

|

Private EC2 Instance (Database Server)

├── Inbound Rules:

| ├── MySQL: Port 3306, Source: Web-Server-SG

├── Outbound Rules:

| ├── All Traffic: Destination: 0.0.0.0/0

Testing Security Groups

1. Test Public EC2 Connectivity

1. Launch a public EC2 instance with the Web-Server-SG.
2. Test HTTP/HTTPS access from a browser using the public IP.

2. Test Private EC2 Connectivity

1. Launch a private EC2 instance with the Database-SG.
2. From the public EC2 instance, test connectivity to the private instance:

```
telnet <private-ec2-ip> 3306
```

Best Practices for Security Groups

1. **Least Privilege**:
   - Allow only the necessary traffic (specific ports and sources).

2. **Use Descriptive Names**:
   - Name Security Groups based on their function (e.g., Web-Server-SG, Database-SG).
3. **Restrict SSH Access**:
   - Limit SSH access to specific IPs or CIDR blocks.
4. **Regular Audits**:
   - Periodically review rules to ensure they align with your requirements.
5. **Avoid Wide-Open Rules**:
   - Avoid using 0.0.0.0/0 unless explicitly required for internet-facing resources.

Common Issues with Security Groups

1. **Connectivity Problems**:
   - Ensure rules allow traffic for the correct protocol, port, and source/destination.
2. **Misconfigured Source or Destination**:
   - Use specific IP ranges or Security Group references instead of 0.0.0.0/0 for sensitive resources.
3. **Rule Overlap**:
   - Avoid conflicting rules in multiple Security Groups attached to the same instance.

Comparison: Security Groups vs. NACLs

Feature	**Security Groups (SG)**	**Network ACLs (NACLs)**
Scope	Instance-Level	Subnet-Level
Statefulness	Stateful	Stateless
Inbound Rules	Automatically match outbound	Must be explicitly defined
Use Case	Resource-level traffic control	Broad subnet-level control