Amazon VPC

VPC Components

A VPC (Virtual Private Cloud) is a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you define.

Main Components

- **Region**: A geographical area containing multiple Availability Zones.
- **Availability Zone**: One or more discrete data centers with redundant power, networking, and connectivity in an AWS Region.
- **VPC**: The main container for networking resources in AWS.

Networking

- **Public Subnet**: Subnet with a route to the Internet Gateway.
- **Private Subnet**: Subnet without a direct route to the Internet.

Connectivity

- **Internet Gateway**: Enables internet access for resources in a public subnet.
- **NAT Gateway**: Allows outbound internet traffic from private subnets while preventing inbound connections.
- **Router**: Handles traffic between subnets and outside of the VPC.
- **Route Table**: Contains rules (routes) to determine where network traffic is directed.

Security

- **Security Group**: Stateful firewall for EC2 instances, controls inbound and outbound traffic.
- **NACL (Network ACL)**: Stateless firewall at the subnet level, controls inbound and outbound traffic.

Compute

- **Public EC2 Instance**: An EC2 instance placed in a public subnet.
- **Private EC2 Instance**: An EC2 instance placed in a private subnet.

Monitoring and Logs

- **VPC Flow Logs**: Capture information about IP traffic going to and from network interfaces.
- **CloudWatch**: Monitoring and observability service.

Additional Services

- **S3**: Object storage service accessible via VPC endpoint or Internet Gateway.
- **Amazon DynamoDB**: NoSQL database service accessible via VPC endpoint or Internet Gateway.

Hybrid Connectivity

- **Customer Gateway**: Represents your on-premise device or software application.
- **VPN Gateway**: AWS side of the VPN connection.
- **S2S VPN Connection**: Site-to-Site VPN connection between your on-premises network and your VPC.
- **Direct Connect (DX)**: Dedicated network connection from your premises to AWS.
- **DX Location**: The physical AWS location where the Direct Connect connection terminates.

Peering and Transit

- **VPC Peering Connection**: Enables networking between two VPCs.
- **Transit Gateway**: Connects VPCs and on-premises networks through a central hub.

Understanding CIDR – IPv4

CIDR stands for Classless Inter-Domain Routing. It is a method for allocating IP addresses and routing IP packets, and is used extensively in AWS networking and Security Group rules.

Key Concepts

- **CIDR Notation**: Defines a range of IP addresses.
- **Examples**:
  - `WW.XX.YY.ZZ/32` → a single IP address
  - `0.0.0.0/0` → all IP addresses (used to allow traffic from anywhere)
  - `192.168.0.0/26` → the range from `192.168.0.0` to `192.168.0.63` (total of 64 IP addresses)

Understanding CIDR – IPv4

A CIDR block (Classless Inter-Domain Routing) consists of two components:

1. Base IP

- Represents an IP address within the range.
- Examples: `10.0.0.0`, `192.168.0.0`, etc.

2. Subnet Mask

- Defines how many bits in the IP address are fixed (network part) and how many can vary (host part).
- Expressed in two forms:
  - Slash notation (CIDR): `/0`, `/24`, `/32`, etc.
  - Dotted decimal:
    - `/8` → `255.0.0.0`
    - `/16` → `255.255.0.0`
    - `/24` → `255.255.255.0`
    - `/32` → `255.255.255.255`

Understanding CIDR – Subnet Mask

The subnet mask determines how many IP addresses can be derived from a base IP address. It defines how many bits are available for host addresses.

Examples

- `192.168.0.0/32` → 1 IP → `192.168.0.0`
- `192.168.0.0/31` → 2 IPs → `192.168.0.0 – 192.168.0.1`
- `192.168.0.0/30` → 4 IPs → `192.168.0.0 – 192.168.0.3`
- `192.168.0.0/29` → 8 IPs → `192.168.0.0 – 192.168.0.7`
- `192.168.0.0/28` → 16 IPs → `192.168.0.0 – 192.168.0.15`
- `192.168.0.0/27` → 32 IPs → `192.168.0.0 – 192.168.0.31`
- `192.168.0.0/26` → 64 IPs → `192.168.0.0 – 192.168.0.63`
- `192.168.0.0/25` → 128 IPs → `192.168.0.0 – 192.168.0.127`
- `192.168.0.0/24` → 256 IPs → `192.168.0.0 – 192.168.0.255`
- ...
- `192.168.0.0/16` → 65,536 IPs → `192.168.0.0 – 192.168.255.255`
- ...

- `192.168.0.0/0` → All IPs → `0.0.0.0 – 255.255.255.255`

Quick Memo

- `/32` : no octets can change
- `/24` : the last octet can change
- `/16` : the last two octets can change
- `/8` : the last three octets can change
- `/0` : all octets can change

Understanding CIDR – Little Exercise

Examples

- `192.168.0.0/24`
  → IP Range: `192.168.0.0 – 192.168.0.255`
  → Total: 256 IPs

- `192.168.0.0/16`
  → IP Range: `192.168.0.0 – 192.168.255.255`
  → Total: 65,536 IPs

- `134.56.78.123/32`
  → Just one IP: `134.56.78.123`

- `0.0.0.0/0`
  → All IP addresses

Tip

For quick CIDR calculations, use:
https://www.ipaddressguide.com/cidr

Public vs. Private IP (IPv4)

The Internet Assigned Numbers Authority (IANA) has designated specific IPv4 address ranges for private (LAN) and public (Internet) use.

Private IP Ranges

These ranges are reserved for use within private networks and are not routable on the public Internet:

- `10.0.0.0 – 10.255.255.255`
  → CIDR: `10.0.0.0/8`
  → Typically used in large enterprise networks

- `172.16.0.0 – 172.31.255.255`
  → CIDR: `172.16.0.0/12`
  → AWS default VPCs often fall in this range

- `192.168.0.0 – 192.168.255.255`
  → CIDR: `192.168.0.0/16`
  → Commonly used in home and small office networks

Public IP

- All IP addresses that do not fall within the above private ranges are considered public and are routable on the Internet.

Default VPC Walkthrough

- Every new AWS account comes with a **default VPC**.
- When launching a new EC2 instance without specifying a subnet, it is automatically launched into the default VPC.
- The default VPC includes **Internet connectivity**.
- All EC2 instances in the default VPC receive a **public IPv4 address**.
- Each instance also gets both a **public and private IPv4 DNS name**.

VPC in AWS – IPv4

What is a VPC?

- **VPC** stands for **Virtual Private Cloud**.
- It allows you to define a logically isolated network within the AWS cloud.

Key Facts

- You can create **multiple VPCs per AWS Region** (default soft limit: 5).
- Each VPC can have up to **5 CIDR blocks**.

CIDR Block Constraints

- **Minimum CIDR size**: `/28` → 16 IP addresses
- **Maximum CIDR size**: `/16` → 65,536 IP addresses

Allowed IP Ranges (Private IPv4 only)

- `10.0.0.0 – 10.255.255.255` → CIDR: `10.0.0.0/8`
- `172.16.0.0 – 172.31.255.255` → CIDR: `172.16.0.0/12`
- `192.168.0.0 – 192.168.255.255` → CIDR: `192.168.0.0/16`

Best Practice

- Make sure your **VPC CIDR blocks do not overlap** with existing networks (e.g., corporate networks) to avoid routing conflicts.

State of Hands-on

This section introduces the practical setup involving:

- **Region**: The geographical area where AWS resources are deployed.
- **VPC**: The Virtual Private Cloud configured within that region.

This likely serves as the starting point for a hands-on lab or exercise involving VPC configuration in a selected AWS region.

Adding Subnets

When designing a VPC within a region, you can create multiple subnets to organize your network.

Structure

- **Region**: The AWS geographical area where resources are deployed.
- **VPC**: A logically isolated network in the cloud within that region.
- **Availability Zone**: A physical data center within the region.
- **Subnets**:
  - **Public Subnet**: Typically has access to the internet via an Internet Gateway.
  - **Private Subnet**: No direct internet access; used for internal services.

Subnets are scoped to Availability Zones and help in separating and organizing resources within a VPC.

VPC – Subnet (IPv4)

In each AWS subnet, 5 IP addresses are reserved by AWS and cannot be used for EC2 instances.

Reserved IP Addresses in a Subnet

For example, if the subnet has CIDR block `10.0.0.0/24`, the reserved IPs are:

- `10.0.0.0` – Network Address
- `10.0.0.1` – Reserved for the VPC router
- `10.0.0.2` – Reserved for Amazon-provided DNS
- `10.0.0.3` – Reserved for future use
- `10.0.0.255` – Broadcast address (AWS does not support broadcast)

Exam Tip

If you need **29 usable IP addresses** for EC2 instances:

- **/27 subnet** provides 32 IPs → 32 – 5 = 27 usable → **not enough**
- **/26 subnet** provides 64 IPs → 64 – 5 = 59 usable → **sufficient**

Internet Gateway (IGW)

An Internet Gateway allows communication between resources in a VPC (e.g., EC2 instances) and the public Internet.

Key Characteristics

- Enables **Internet access** for resources inside a VPC.
- **Scales horizontally**, ensuring high availability and redundancy.
- Must be **created separately** from the VPC.
- A **one-to-one relationship**: one IGW can be attached to only one VPC, and vice versa.

Important Note

- An IGW alone is **not sufficient** to enable Internet access.
- You must also **update the Route Table** to include a route to the Internet via the IGW.

Adding Internet Gateway

To provide internet access to resources in a public subnet, you must attach an Internet Gateway (IGW) to your VPC.

Structure Overview

- **Region**: Geographical AWS location
- **VPC**: Virtual Private Cloud within the region
- **Availability Zone**: Physical data center
- **Subnets**:

    - **Private Subnet**: No direct internet access
    - **Public Subnet**: Connected to the Internet Gateway
- **Internet Gateway**: Component that allows communication between public subnet resources and the internet

After attaching the IGW, you must configure the route table to direct outbound traffic (e.g., `0.0.0.0/0`) to the IGW.

Editing Route Tables

To enable internet access for EC2 instances in a public subnet, you must configure the Route Table associated with the subnet.

Components Involved

- **Region**: The AWS geographical area
- **VPC**: The virtual network
- **Availability Zone**: The physical data center
- **Subnets**:
    - **Private Subnet**: No internet access
    - **Public Subnet**: Can access the internet through the IGW
- **Internet Gateway**: Connects the VPC to the internet
- **Router**: Handles traffic within and outside the VPC
- **Route Table**:
    - Must include a route: `0.0.0.0/0 → Internet Gateway`
- **Security Group**: Controls inbound and outbound traffic
- **Public EC2 Instance**: Deployed in the public subnet and reachable from the internet

Summary

- Attaching an IGW is not enough: you must edit the **Route Table** to direct traffic to the IGW.
- Ensure the **Security Group** also allows inbound and outbound traffic as needed.

Bastion Hosts

A **Bastion Host** is used to securely access private EC2 instances via SSH.

How It Works

- The Bastion Host is deployed in a **public subnet** with Internet access.
- It serves as a secure entry point to **private subnets** in the VPC.
- Users connect to the Bastion Host via **SSH**, then access private EC2 instances from there.

Security Group Configuration

- **Bastion Host Security Group**:
    - Must allow **inbound traffic on port 22 (SSH)**.
    - Should restrict access to trusted IP ranges (e.g., your corporate public CIDR).
- **Private EC2 Instance Security Group**:
    - Must allow **SSH access from the Bastion Host**, either by:
        - Allowing the **Security Group of the Bastion Host**, or
        - Allowing the **private IP address** of the Bastion Host.

This setup ensures secure access to private resources without exposing them directly to the Internet.

NAT Instance (Outdated, but Still in the Exam)

A **NAT Instance** (Network Address Translation) allows EC2 instances in private subnets to initiate outbound traffic to the Internet, but prevents unsolicited inbound traffic from the Internet.

Key Characteristics

- Must be launched in a **public subnet**.
- Requires an **Elastic IP (EIP)**.
- You must **disable the EC2 setting**: *Source/Destination Check*.
- The **Route Table** for the private subnet must direct internet-bound traffic (e.g., `0.0.0.0/0`) to the NAT instance.

How It Works

- A private EC2 instance sends a request to an external server.
- The NAT instance translates the **source IP** from the private IP to its **Elastic IP**.
- The response from the external server is received by the NAT instance, which forwards it back to the private EC2 instance.

Summary

- NAT Instance provides **outbound Internet access** for **private subnet** resources.
- This method is now considered **outdated**, but still appears on the AWS exam.

NAT Instance – Architecture Overview

This diagram illustrates the architecture of a VPC using a **NAT Instance** to provide Internet access to private EC2 instances.

Components

- **Region**: AWS geographical location
- **VPC**: Virtual Private Cloud
- **Availability Zone**: Data center within the region
- **Subnets**:
    - **Public Subnet**: Contains the NAT Instance and Public EC2 Instance
    - **Private Subnet**: Contains Private EC2 Instance

Network Components

- **Internet Gateway**: Provides internet access to the public subnet
- **Router**: Handles traffic within and outside the VPC
- **Route Tables**:
    - Public subnet route table includes route to Internet Gateway
    - Private subnet route table includes route to the NAT Instance
- **Security Groups**: Control traffic to/from EC2 instances and NAT Instance
- **Elastic IP (EIP)**: Attached to the NAT Instance for outbound internet communication

Flow Summary

- The **private EC2 instance** routes internet-bound traffic to the **NAT Instance**.
- The **NAT Instance**, via its EIP, communicates with the Internet.

- Responses return to the **NAT**, which forwards them back to the **private EC2 instance**.

NAT Instance – Comments

Pre-Configured AMI

- Amazon provided a **pre-configured Amazon Linux AMI** for NAT instances.
- This AMI **reached end of standard support on December 31, 2020**.

Availability and Resilience

- A NAT Instance is **not highly available by default**.
- To improve resilience, you must:
  - Deploy it in an **Auto Scaling Group (ASG)**.
  - Use a **multi-AZ** deployment.
  - Implement a **resilient user-data script** for reconfiguration.

Performance

- **Internet bandwidth** depends on the **EC2 instance type** used.

Security Group Configuration

- **Inbound Rules**:

  - Allow **HTTP/HTTPS traffic** from private subnets.
  - Allow **SSH access** from your trusted network (e.g., your home or corporate public IP range).

- **Outbound Rules**:

  - Allow **HTTP/HTTPS traffic** to the Internet.

Proper configuration of security groups and careful instance type selection are essential for using NAT Instances effectively.

NAT Gateway

A **NAT Gateway** (NATGW) is a fully managed AWS service that allows instances in a private subnet to access the Internet without allowing unsolicited inbound traffic.

Key Features

- **Managed by AWS**: No need for manual maintenance or patching.
- **High Bandwidth**: Starts at 5 Gbps and automatically scales up to 100 Gbps.
- **Highly Available**: Designed for automatic failover within the Availability Zone.
- **Elastic IP**: Assigned automatically upon creation.

Usage and Billing

- You are charged **per hour** and **per GB of data processed**.

Deployment Constraints

- NAT Gateway must be **created in a public subnet**.
- **Only instances in other subnets** (typically private subnets) can use the NAT Gateway.
- **Requires an Internet Gateway (IGW)** to reach the Internet:
  - Flow: **Private Subnet → NAT Gateway → IGW**

Simplified Security

- **No Security Groups** are required for the NAT Gateway.

The NAT Gateway is the recommended solution over NAT Instances for production use due to its performance, availability, and ease of management.

NAT Gateway – Architecture Overview

This architecture illustrates how a **NAT Gateway** provides Internet access to private EC2 instances.

Components

- **Region**: AWS geographic location
- **VPC**: Virtual Private Cloud
- **Availability Zone**: Physical data center in the region
- **Subnets**:
  - **Public Subnet**: Hosts the NAT Gateway and public EC2 instance
  - **Private Subnet**: Hosts private EC2 instances that need outbound Internet access

Network Components

- **Internet Gateway (IGW)**: Enables access to the Internet
- **Router**: Handles internal and external routing within the VPC
- **Route Tables**:
  - Public subnet routes outbound traffic to the IGW
  - Private subnet routes Internet-bound traffic to the NAT Gateway
- **Security Groups**:
  - Protect the public and private EC2 instances
  - NAT Gateway does **not** require a security group

Flow Summary

- Private EC2 instances send traffic to the NAT Gateway.
- NAT Gateway, via the Internet Gateway, routes traffic to the Internet.
- Responses return through the NAT Gateway and are delivered to the private instances.

This setup ensures private instances can access the Internet securely without being directly exposed.

NAT Gateway with High Availability

Resilience Characteristics

- A **NAT Gateway is resilient** within a **single Availability Zone (AZ)**.
- To ensure **fault tolerance**, you must deploy **multiple NAT Gateways**, one in **each AZ** where private subnets require Internet access.

No Cross-AZ Failover

- AWS does **not support cross-AZ failover** for NAT Gateways.
- If an AZ becomes unavailable, traffic from that AZ will not route through a NAT Gateway in another AZ.
- However, if an AZ is down, resources in that AZ are also down, so **failover is unnecessary** for the NAT Gateway.

Architecture Overview

- **Region** with multiple **Availability Zones (e.g., AZ-A, AZ-B)**
- Each AZ contains:
  - A **Public Subnet** with a **NAT Gateway**
  - A **Private Subnet** with **EC2 Instances**
- All NAT Gateways connect to a common **Internet Gateway** (IGW) via appropriate route tables
- Each private subnet routes Internet-bound traffic to its local NAT Gateway

This setup ensures **high availability and AZ-level isolation** for outbound Internet connectivity.

NAT Gateway vs. NAT Instance

A comparison between **NAT Gateway** and **NAT Instance**:

Feature	NAT Gateway	NAT Instance
Availability	Highly available within an AZ (create in multiple AZs for fault-tolerance)	Requires custom script to manage failover
Bandwidth	Automatically scales up to **100 Gbps**	Depends on **EC2 instance type**
Maintenance	Fully **managed by AWS**	Must be **managed by user** (OS, software, etc.)
Cost	Billed per **hour** and **data transferred**	Billed per **hour**, instance type + **network usage**
Public IPv4	Required	Required
Private IPv4	Supported	Supported
Security Groups	**Not required**	**Required**
Use as Bastion Host?	❌ No	✅ Yes

For more information, refer to the [AWS NAT Comparison Guide].

Subnet – Security Groups & NACLs

This diagram illustrates how **Security Groups (SG)** and **Network Access Control Lists (NACLs)** control traffic at different levels within a subnet.

Key Concepts

- **Security Groups (SG)** are **stateful**:
  - If an inbound rule allows traffic, the response is automatically allowed.
  - Same for outbound traffic.
- **NACLs** are **stateless**:
  - Rules must explicitly allow both inbound and outbound traffic.
  - Each direction (inbound/outbound) is evaluated separately.

Inbound Traffic Flow

1. Incoming request hits **NACL Inbound Rules** (stateless).
2. Then it is evaluated by the **SG Inbound Rules** (stateful).
3. If allowed, traffic reaches the **EC2 instance**.

Outbound Traffic Flow

1. Outgoing request hits **SG Outbound Rules** (stateful).
2. Then it is evaluated by the **NACL Outbound Rules** (stateless).
3. If allowed, traffic leaves the subnet.

Summary

- **NACLs** operate at the **subnet level**, are stateless, and evaluate traffic independently for each direction.
- **Security Groups** operate at the **instance level**, are stateful, and automatically allow return traffic.

Network Access Control List (NACL)

A **Network ACL (NACL)** acts as a firewall that controls inbound and outbound traffic at the **subnet level**.

Key Characteristics

- **One NACL per subnet**
  - New subnets are assigned the **default NACL** by default.
- NACLs are **stateless**, meaning return traffic must be explicitly allowed.

NACL Rules

- Each rule has a **number** from **1 to 32766**.
  - Lower numbers have **higher precedence**.
- The **first matching rule** determines the outcome (ALLOW or DENY).
- Example:
  - Rule #100: `ALLOW 10.0.0.10/32`
  - Rule #200: `DENY 10.0.0.10/32`
  - The IP will be **allowed** due to the lower rule number.
- The last implicit rule is `*`, which **denies all unmatched traffic**.

Recommendations

- AWS recommends incrementing rule numbers by **100** (e.g., 100, 200, 300) to allow room for future changes.
- **Newly created NACLs deny all traffic** by default.
- NACLs are ideal for **blocking specific IPs** at the subnet level.

NACLs – Architecture Overview

This architecture shows how **Network Access Control Lists (NACLs)** are used to control traffic at the **subnet level** within a VPC.

Components

- **Region**: AWS geographical location
- **VPC**: Virtual network
- **Availability Zone**: Isolated data center within the region
- **Subnets**:
  - **Public Subnet**: Contains Public EC2 Instance and NAT Gateway

- **Private Subnet**: Contains Private EC2 Instance

- **Internet Gateway (IGW)**: Enables internet access
- **Router**: Handles routing between subnets and external traffic
- **Route Tables**: Control the routing of traffic for each subnet
- **Security Groups**: Applied at the EC2 instance level
- **NACLs**: Applied at the subnet level for both inbound and outbound traffic

Summary

- NACLs are associated with both public and private subnets.
- Each subnet can have **one NACL**, which evaluates **inbound and outbound** rules.
- NACLs act as a **stateless** filter, requiring explicit rules for both directions.

This layered security approach allows fine-grained control of network traffic at both the subnet and instance levels.

Default NACL

The **Default Network Access Control List (NACL)** in a VPC is designed to allow all traffic by default.

Behavior

- **Inbound and outbound traffic** are both **fully allowed**.
- Applies to all **subnets** associated with the default NACL.

Inbound Rules (Default)

Rule #	Type	Protocol	Port Range	Source	Allow/Deny
100	All IPv4 Traffic	All	All	0.0.0.0/0	ALLOW
*	All IPv4 Traffic	All	All	0.0.0.0/0	DENY

Outbound Rules (Default)

Rule #	Type	Protocol	Port Range	Destination	Allow/Deny
100	All IPv4 Traffic	All	All	0.0.0.0/0	ALLOW
*	All IPv4 Traffic	All	All	0.0.0.0/0	DENY

Best Practice

- **Do NOT modify** the Default NACL.
- Instead, **create custom NACLs** tailored to your network security needs.

The default configuration is permissive, suitable for testing or basic scenarios, but not recommended for production environments.

Ephemeral Ports

For any two endpoints to establish a connection, they must use ports.

- Clients connect to a defined port and expect a response on an ephemeral port.

- Different operating systems use different ephemeral port ranges:

- IANA & Microsoft Windows 10: 49152 – 65535
- Many Linux Kernels: 32768 – 60999

Example: HTTPS Request

- **Web Server**
  - IP: 55.66.77.88
  - Fixed Port: 443
- **Client**
  - IP: 11.22.33.44
  - Ephemeral Port: 50105

Request (Client to Server)

- Source IP: 11.22.33.44
- Source Port: 50105
- Destination IP: 55.66.77.88
- Destination Port: 443
- Payload: HTTP request (e.g., `http://www.datacumulus.com/`)

Response (Server to Client)

- Source IP: 55.66.77.88
- Source Port: 443
- Destination IP: 11.22.33.44
- Destination Port: 50105
- Payload: HTTP response

NACL with Ephemeral Ports

Architecture Overview

- **VPC** is divided into:
  - Web Subnet (Public)
  - DB Subnet (Private)
- **Tiers**:
  - Web Tier
  - Database Tier
- **DB Instance** listens on port **3306** (MySQL default port)

NACL Rules Configuration

Web-NACL

- **Allow Outbound TCP**
  - Port: 3306
  - Destination: DB Subnet CIDR
- **Allow Inbound TCP**

    - Port range: 1024–65535
    - Source: DB Subnet CIDR

DB-NACL

- **Allow Inbound TCP**
    - Port: 3306
    - Source: Web Subnet CIDR
- **Allow Outbound TCP**
    - Port range: 1024–65535
    - Destination: Web Subnet CIDR

Purpose

- When the web tier connects to the DB tier on port 3306, the response from the DB uses an ephemeral port (typically in the range 1024–65535).
- The NACLs must explicitly allow these return paths using ephemeral port ranges.

References

- [AWS Documentation on NACL and Ephemeral Ports]

Create NACL Rules for Each Target Subnet's CIDR

Architecture Overview

- **VPC** with multiple subnets in different Availability Zones:
    - Web Subnet - A (Public)
    - Web Subnet - B (Public)
    - DB Subnet - A (Private)
    - DB Subnet - B (Private)
- **Tiers**:
    - Web Tier: spans Subnet A and B (public)
    - Database Tier: spans Subnet A and B (private)
    - Each tier contains its own DB Instance

NACL Configuration Guideline

- For proper communication between tiers across Availability Zones, you must:
    - **Create separate NACL rules** for each target subnet's **CIDR block**.
    - This ensures that traffic between subnets (e.g., Web Subnet A to DB Subnet B) is allowed explicitly.
    - NACLs are stateless: return traffic must be explicitly allowed in the opposite direction.
- Apply **Web-NACL** to public web subnets.
- Apply **DB-NACL** to private database subnets.

Key Takeaway

When setting up NACLs in a multi-AZ VPC design:

- Define rules for each specific subnet CIDR.
- Ensure inbound and outbound rules are symmetric to maintain bi-directional communication.

Security Group vs. NACLs

Feature	Security Group	NACL (Network ACL)
Level of Operation	Instance level	Subnet level
Rule Types Supported	Allow rules only	Allow and deny rules
Statefulness	Stateful – return traffic is automatically allowed	Stateless – return traffic must be explicitly allowed
Rule Evaluation	All rules are evaluated	Rules evaluated in order, lowest to highest (first match wins)
Application Scope	Applies to an EC2 instance when assigned	Automatically applies to all instances in the subnet

Key Differences

- **Security Groups** are easier to manage for individual EC2 instances and allow all return traffic by default.
- **NACLs** offer more granular control at the subnet level but require explicit rules for return traffic, including ephemeral ports.

References

- NACL Examples in AWS Documentation

VPC Peering

- VPC Peering allows two VPCs to connect privately using the AWS network.
- It enables the VPCs to behave as if they were part of the same network.
- **CIDR blocks must not overlap** between the peered VPCs.
- **VPC Peering is not transitive**:
    - If VPC A is peered with VPC B, and VPC B is peered with VPC C, VPC A is **not** automatically connected to VPC C.
    - You must explicitly create a peering connection between VPC A and VPC C if communication is required.

Routing Configuration

- You must update the **route tables** in each VPC's subnets to allow EC2 instances to communicate across VPCs.

Example Connections

- **VPC Peering (A – B)**
- **VPC Peering (B – C)**
- **VPC Peering (A – C)** — must be created separately for A to talk to C

VPC Peering – Good to Know

- You can create a **VPC Peering connection between VPCs in different AWS accounts** or even **across regions**.
- It is possible to **reference a security group in a peered VPC**, provided the VPCs are in the **same region**, even if they belong to **different accounts**.

VPC Peering – Architecture Overview

Components

- **Region**: The geographic location where the VPC resides.
- **VPC**: Contains both public and private subnets.
- **Availability Zone**: Each subnet is located in an AZ within the region.

Subnet Types

- **Public Subnet**:
  - Connected to the **Internet Gateway**.
  - Hosts **Public EC2 Instances** that can access the internet.
  - Uses a **Route Table** that directs traffic to the Internet Gateway.
  - Protected by **Security Groups** and **NACLs**.
- **Private Subnet**:
  - No direct internet access.
  - Routes outbound internet traffic through a **NAT Gateway** (located in the public subnet).
  - Hosts **Private EC2 Instances**.
  - Uses its own **Route Table** and **Security Groups**/**NACLs** for control.

VPC Peering Connections

- Enables private connectivity between different VPCs.
- Requires configuration of **Route Tables** and **Security Groups** to allow communication.
- VPC Peering does not allow transitive routing—connections must be explicit.

VPC Endpoints – Architecture Overview

Purpose

- VPC Endpoints allow **private connectivity** between your VPC and AWS services **without requiring an internet gateway, NAT device, VPN, or AWS Direct Connect**.
- Useful for accessing AWS services like **S3**, **DynamoDB**, and **CloudWatch** from **private subnets**.

Architecture Components

- **VPC**: Contains both public and private subnets.
- **Public Subnet**:
  - Connected to the **Internet Gateway**.
  - Hosts **Public EC2 Instances**.
  - Uses a Route Table and Security Group.
- **Private Subnet**:
  - No direct internet access.
  - Uses a **NAT Gateway** for outbound internet.

- Hosts **Private EC2 Instances**.
- Connected to **VPC Endpoints** for private access to AWS services.

- **VPC Endpoints**:

    - Appear in private subnets.
    - Provide private links to services like **Amazon S3**, **DynamoDB**, **CloudWatch**, etc.
    - Traffic stays within the AWS network.

Benefits

- Enhances security by avoiding traffic over the public internet.
- Reduces data transfer costs and increases performance.
- Simplifies compliance for private workloads.

VPC Endpoints (AWS PrivateLink)

Overview

- By default, **AWS services are accessible via public URLs** over the internet.
- **VPC Endpoints**, powered by **AWS PrivateLink**, allow you to connect to AWS services over the **AWS private network**, avoiding public internet exposure.

Key Features

- Provide **private connectivity** to AWS services from within your VPC.
- Eliminate the need for:
    - Internet Gateway (IGW)
    - NAT Gateway (NATGW)
- **Highly available** and **horizontally scalable**.

Troubleshooting Tips

- If connectivity issues occur:
    - Verify **DNS settings** and resolution within your VPC.
    - Ensure **Route Tables** are correctly configured to direct traffic to the VPC Endpoint.

Architecture

- **Public Subnet**:

    - Connected to Internet Gateway.
    - Can contain EC2 instances with public access.

- **Private Subnet**:

    - Uses **VPC Endpoints** to access services like **Amazon SNS** privately.
    - Can also route through a NAT Gateway for public access if needed.

Options for SNS Access

- **Option 1**: EC2 instance in Private Subnet uses **VPC Endpoint** to access **Amazon SNS**.
- **Option 2**: EC2 instance in Public Subnet accesses **Amazon SNS** via public internet (not recommended for private access).

Types of VPC Endpoints

1. Interface Endpoints (powered by PrivateLink)

- Creates an **Elastic Network Interface (ENI)** with a private IP address in your subnet.
- Used as the entry point to the AWS service.
- Must be associated with a **Security Group**.
- Supports **most AWS services**.
- **Pricing**:
  - Charged **per hour**
  - Plus **per GB of data processed**

2. Gateway Endpoints

- Creates a **gateway** in your route table to connect to specific AWS services.
- Does **not use security groups**.
- Must be specified as a **target in a route table**.
- Supports only:
  - **Amazon S3**
  - **Amazon DynamoDB**
- **Free of charge**

Architecture Summary

- **Interface Endpoint Example**:
  - Region → VPC → Private Subnet
  - EC2 Instance connects to **Amazon SNS** via a **VPC Endpoint (Interface)** using PrivateLink ENI.
- **Gateway Endpoint Example**:
  - Region → VPC → Private Subnet
  - EC2 Instance connects to **Amazon S3** or **DynamoDB** via a **VPC Endpoint (Gateway)**.

Gateway or Interface Endpoint for S3?

Exam Tip

- **Gateway Endpoint** is **preferred** for **Amazon S3** in most exam scenarios.

Cost Comparison

- **Gateway Endpoint**: **Free**
- **Interface Endpoint**: **Paid**
  - Charges apply per hour and per GB of data processed.

When to Use Interface Endpoint

- Use **Interface Endpoint** (PrivateLink) if access to S3 is required from:
  - **On-premises** environments (via Site-to-Site VPN or AWS Direct Connect)
  - A **different VPC**
  - A **different region**

Architecture Summary

- Applications inside a **VPC** can access **Amazon S3**:
  - Using a **Gateway Endpoint** when traffic stays within the same VPC and region.

    - Using an **Interface Endpoint** for **cross-VPC, cross-region, or hybrid cloud** access scenarios.

Lambda in VPC Accessing DynamoDB

Context

- **DynamoDB** is a public AWS service.
- When a **Lambda function is placed inside a VPC**, it cannot access the public internet by default.

Option 1: Access via Public Internet

- Requires:
    - **NAT Gateway** in a **Public Subnet**
    - **Internet Gateway (IGW)**
- This setup allows Lambda in a **Private Subnet** to access DynamoDB via the internet.
- **Not ideal** due to additional cost and complexity.

Option 2 (Preferred & Free): Use VPC Gateway Endpoint

- Deploy a **VPC Gateway Endpoint for DynamoDB**.
- Modify the **Route Tables** in the private subnets to use this endpoint.
- Enables **private, secure, and free** access from Lambda to DynamoDB **without using NAT or IGW**.

Summary

- Always prefer **Gateway Endpoint** for DynamoDB access from Lambda in a VPC when possible.

VPC Flow Logs

Overview

- **VPC Flow Logs** capture information about **IP traffic** going into and out of network interfaces in your VPC.
- Types of flow logs:
    - **VPC-level Flow Logs**
    - **Subnet-level Flow Logs**
    - **Elastic Network Interface (ENI) Flow Logs**

Use Cases

- **Monitor and troubleshoot** network connectivity issues.
- Analyze traffic patterns for security and compliance.

Destinations for Flow Log Data

- **Amazon S3**
- **Amazon CloudWatch Logs**
- **Amazon Kinesis Data Firehose**

Supported AWS Services

VPC Flow Logs can capture data from AWS-managed interfaces such as:

- **Elastic Load Balancers (ELB)**
- **Amazon RDS**
- **Amazon ElastiCache**

- **Amazon Redshift**
- **Amazon WorkSpaces**
- **NAT Gateway (NATGW)**
- **Transit Gateway**

VPC Flow Logs – Architecture Overview

Where Flow Logs Operate

- **Region → VPC → Subnets (Public & Private) → Network Interfaces**
- Flow logs can be enabled at:
  - **VPC level**
  - **Subnet level**
  - **Elastic Network Interface (ENI) level**

Components in the Architecture

- **Public Subnet**
  - Contains public EC2 instances
  - Connected to an **Internet Gateway**
  - Associated with a **Route Table**, **Security Group**, and **NACL**
- **Private Subnet**
  - Contains private EC2 instances
  - Routes external traffic through a **NAT Gateway** in the public subnet
  - Also has its own **Route Table**, **Security Group**, and **NACL**
- **Other Connections**
  - **VPC Peering**
  - **VPC Endpoints** (e.g., to access S3, DynamoDB)
  - **CloudWatch** and **S3** as destinations for Flow Log data

Flow Log Output

- Flow Logs capture traffic information and send it to:
  - **Amazon CloudWatch Logs**
  - **Amazon S3**

This allows monitoring and debugging network activity across the architecture.

VPC Flow Logs Syntax

Key Fields

- **srcaddr**: Source IP address — useful to identify the source of traffic.
- **dstaddr**: Destination IP address — helps identify the target of the traffic.
- **srcport**: Source port — helps identify the originating application or service.
- **dstport**: Destination port — identifies which service was targeted.
- **protocol**: Indicates the protocol used (e.g., TCP, UDP).
- **action**: Indicates if the request was accepted or rejected (due to Security Group or NACL).
- **log-status**: Indicates if the log record was successfully captured.

Additional Fields

- **version**: Flow log version.
- **account-id**: AWS account ID.
- **interface-id**: Elastic Network Interface ID.
- **packets**: Number of packets transferred.
- **bytes**: Number of bytes transferred.
- **start** / **end**: Timestamps for the beginning and end of the captured flow.

Use Cases

- **Analytics**: Usage pattern analysis, performance monitoring, and capacity planning.
- **Security**: Detect unusual or malicious traffic behavior.
- **Troubleshooting**: Identify connectivity issues related to ports, IPs, or access rules.

Querying Tools

- **Amazon Athena** (for logs stored in S3)
- **CloudWatch Logs Insights** (for logs sent to CloudWatch)

Reference

- [AWS Flow Logs Examples]

VPC Flow Logs – Troubleshooting Security Group & NACL Issues

Incoming Requests

- **Inbound REJECT**:
  - Could be due to **NACL inbound rule** or **Security Group inbound rule**.
- **Inbound ACCEPT**, but **Outbound REJECT**:
  - Most likely caused by **NACL outbound rule**.

Outgoing Requests

- **Outbound REJECT**:
  - Could be due to **NACL outbound rule** or **Security Group outbound rule**.
- **Outbound ACCEPT**, but **Inbound REJECT**:
  - Most likely caused by **NACL inbound rule**.

Key Concepts

- **Security Groups (SG)** are **stateful**:
  - If a request is allowed in one direction, the response is automatically allowed.
- **Network ACLs (NACLs)** are **stateless**:
  - You must explicitly allow traffic in both directions (inbound and outbound).

How to Use VPC Flow Logs for Troubleshooting

- Look at the **"action"** field:
  - `ACCEPT` means the traffic was allowed.

- `REJECT` means the traffic was blocked.
- Combine this with direction and port information (`srcport` , `dstport`) to identify whether the issue is with **SG** or **NACL**.

VPC Flow Logs – Architectures

Architecture 1: CloudWatch Logs Integration

- **VPC Flow Logs → CloudWatch Logs**
- Use **CloudWatch Contributor Insights** to analyze:
    - Top-10 IP addresses
    - Traffic trends and anomalies
- Define **Metric Filters** and **CloudWatch Alarms** to detect events.
- Trigger **Amazon SNS Alerts** for notifications (e.g., on SSH/RDP access attempts).

Architecture 2: S3 and Athena Integration

- **VPC Flow Logs → S3 Bucket**
- Query logs using **Amazon Athena**
- Visualize insights with **Amazon QuickSight**

Use Cases

- Detect unusual access patterns or port scanning.
- Monitor usage of protocols like **SSH**, **RDP**.
- Generate visual dashboards and analytics for security auditing and compliance.

AWS Site-to-Site VPN – Architecture Overview

Overview

AWS Site-to-Site VPN enables secure communication between an on-premises **corporate data center** and an **AWS VPC** over the internet using an encrypted connection.

Key Components

- **Customer Gateway (CGW)**: Represents the on-premises VPN device or software.
- **VPN Gateway (VGW)**: AWS side of the VPN connection attached to the VPC.
- **Site-to-Site VPN Connection**: The secure IPSec tunnel between CGW and VGW.

VPC Layout

- **Public Subnet**:
    - Connected to an **Internet Gateway**
    - Hosts **Public EC2 Instances**
    - Associated with **Security Groups**, **Route Tables**, and **NACLs**
- **Private Subnet**:
    - Hosts **Private EC2 Instances**
    - Routes outbound internet traffic through a **NAT Gateway** in the public subnet
    - Can also access AWS services (e.g., **DynamoDB**) via **VPC Endpoints**
    - Uses **VPC Flow Logs** for monitoring and analysis

Additional Components

- **VPC Peering Connections**: Allow communication between VPCs
- **CloudWatch & S3**: Used for storing and analyzing Flow Logs
- **Route Tables**: Must be updated to route traffic to/from the VPN Gateway
- **Security Groups & NACLs**: Control access at the instance and subnet level

Use Cases

- Extend your on-prem network into the cloud securely
- Hybrid cloud workloads with consistent networking
- Access AWS services privately without exposing traffic to the public internet

AWS Site-to-Site VPN – Core Components

Virtual Private Gateway (VGW)

- Acts as the **VPN concentrator** on the AWS side.
- Must be **created and attached** to the VPC that will use the Site-to-Site VPN connection.
- You can **customize the ASN (Autonomous System Number)** for routing purposes.

Customer Gateway (CGW)

- Represents the customer-side **VPN endpoint**.
- Can be a **software application** or a **physical device** (e.g., router, firewall).
- For a list of tested customer devices, refer to:
  - [AWS Tested Customer Gateway Devices]

Summary

- The VGW and CGW together establish a secure **IPSec tunnel** between your AWS VPC and your on-premises infrastructure.

Site-to-Site VPN Connections – Configuration Details

Customer Gateway Device (On-Premises)

- Must use a **public, internet-routable IP address** for your **Customer Gateway (CGW)** device.
- If the device is **behind a NAT**, and **NAT Traversal (NAT-T)** is enabled:
  - Use the **public IP address of the NAT device** instead.

Routing Configuration

- A critical step is to **enable Route Propagation** for the **Virtual Private Gateway (VGW)** in the **Route Table** associated with your VPC subnets.
  - This allows dynamically learned routes from the VGW to be added to the routing table.

Security Group Configuration

- To allow traffic from on-premises (e.g., ICMP for ping):
  - You must **explicitly allow the ICMP protocol** in the **inbound rules** of your EC2 instance **Security Groups**.

Summary Diagram

- **VPC** with **Private Subnet**
- **Virtual Private Gateway** connected to AWS
- **Customer Gateway** device on-premises with either:
  - **Public IP** directly

- OR a **Private IP** behind a **NAT device** with **Public IP**

- Ensure **Route Table** has Route Propagation enabled

AWS VPN CloudHub

Overview

- **AWS VPN CloudHub** allows secure communication **between multiple remote sites** using **VPN connections**.
- It is designed for organizations with multiple on-premises networks needing interconnectivity via AWS.

Key Features

- Uses a **hub-and-spoke topology**:
  - The **Virtual Private Gateway (VGW)** acts as the central hub.
  - Each **Customer Gateway (CGW)** connects to the VGW.

- **Cost-effective** solution for **primary or secondary network connectivity**.
- All traffic is encrypted and travels over the **public Internet** (standard VPN behavior).

Setup Steps

1. Establish **multiple Site-to-Site VPN connections** to the same VGW from different customer sites.
2. Enable **dynamic routing** (typically via BGP).
3. Configure **route tables** accordingly to allow cross-site communication.

Architecture Summary

- **VPC** with private subnets in multiple Availability Zones.
- **EC2 Instances** hosted within the VPC.
- **Customer Gateways** from multiple customer networks connect to the same **Virtual Private Gateway**.

AWS Direct Connect (DX)

Overview

- AWS Direct Connect provides a **dedicated private network connection** from your on-premises **data center** to your **AWS VPC**.
- Enables access to both:
  - **Private resources** (e.g., EC2 instances)
  - **Public AWS services** (e.g., Amazon S3)
  - Through the **same physical connection**

Requirements

- A **dedicated line** must be provisioned between your on-premises location and an **AWS Direct Connect location**.
- You must set up a **Virtual Private Gateway (VGW)** on your VPC to establish the connection.

Use Cases

- **High bandwidth needs**: Ideal for transferring large datasets; offers **lower cost per GB** than standard internet-based connections.
- **Low latency and high reliability**: Useful for **real-time data feeds** and critical applications.
- **Hybrid cloud environments**: Seamlessly integrate on-prem and AWS workloads.

Additional Notes

- Supports both **IPv4** and **IPv6** protocols.

AWS Direct Connect – Architecture Diagram

Key Components

- **Region (e.g., us-east-1)**: AWS Region where your VPC is located.
- **VPC**: Contains private subnets and **EC2 Instances**.
- **Virtual Private Gateway (VGW)**: Attached to the VPC to terminate the Direct Connect connection.

Connectivity Path

1. **Corporate Data Center**:
    - Connects to **Customer Router/Firewall**.
    - Traffic is directed over dedicated connections using **VLANs**.
2. **Direct Connect Infrastructure**:
    - **Customer or Partner Router** located in an AWS Direct Connect colocation facility (cage).
    - Connects to the **AWS Direct Connect Endpoint** in the **AWS Cage**.
3. **Virtual Interfaces**:
    - **Private Virtual Interface (VLAN 1)**: Used to access private resources in the VPC (e.g., EC2).
    - **Public Virtual Interface (VLAN 2)**: Used to access public AWS services (e.g., S3, Glacier).

Summary

- Enables **low-latency**, **high-throughput**, and **private** connectivity between on-premises and AWS.
- Supports **segregated traffic** via virtual interfaces for public and private access over the same physical link.

Direct Connect Gateway

Purpose

- A **Direct Connect Gateway** is used when you want to establish **a single Direct Connect connection** to **multiple VPCs** located in **different AWS Regions** within the **same AWS account**.

Benefits

- Eliminates the need to create separate Direct Connect connections for each VPC in different regions.
- Simplifies network architecture by **centralizing connectivity**.
- Reduces operational and infrastructure costs.

Architecture Overview

- A **Direct Connect Gateway** is connected to the on-premises **Customer Network** via an **AWS Direct Connect connection**.
- From the Direct Connect Gateway, **private virtual interfaces** are created to connect to multiple VPCs:
    - Example:
        - **VPC in us-east-1** with CIDR `10.0.0.0/16`
        - **VPC in us-west-1** with CIDR `172.16.0.0/16`

Each VPC is associated with the Direct Connect Gateway through its **Virtual Private Gateway (VGW)**.

AWS Direct Connect – Connection Types

1. Dedicated Connections

- Speeds available: **1 Gbps**, **10 Gbps**, **100 Gbps**
- Provides a **physical Ethernet port** dedicated exclusively to a single customer.
- The request is initiated directly with **AWS**, and the setup is completed in coordination with an **AWS Direct Connect Partner**.

2. Hosted Connections

- Speeds available: **50 Mbps**, **500 Mbps**, up to **10 Gbps**
- Requests are made through **AWS Direct Connect Partners**.
- **On-demand scalability**: capacity can be increased or decreased as needed.
- Fixed sizes: **1, 2, 5, 10 Gbps** options available at select partners.

Important Note

- Establishing a new Direct Connect connection typically involves **long lead times**, often **exceeding one month**.

AWS Direct Connect – Encryption

Default Behavior

- **Data in transit over Direct Connect is not encrypted**, but the connection is **private** and does not traverse the public internet.

Enhancing Security with VPN

- You can combine **AWS Direct Connect with a VPN** connection to add **IPsec encryption** on top of the private link.
- This setup ensures:
  - **End-to-end encryption**
  - **Private network routing**
  - Higher **security assurance**, especially for sensitive workloads

Considerations

- Combining Direct Connect with VPN adds **extra complexity** to the setup and maintenance.
- Useful for **high-security environments** requiring both **performance** and **encryption**.

Architecture Summary

- **Region (us-east-1)** with a VPC spanning multiple Availability Zones
- EC2 instances in **private subnets**
- **Customer Network** connected via:
  - **AWS Direct Connect** for private, low-latency connectivity
  - **VPN Connection** layered on top for encrypted data in transit

AWS Direct Connect – Resiliency

High Resiliency for Critical Workloads

- Achieved by having **one Direct Connect connection at multiple locations**.
- This setup provides **redundancy** in case a single location or device fails.

Maximum Resiliency

- Achieved by provisioning **separate Direct Connect connections**:
  - Terminate on **separate devices**
  - Located in **more than one physical AWS Direct Connect location**
- Ensures **full fault isolation** and **maximum uptime**, ideal for **mission-critical applications**.

Architecture Summary

High Resiliency

- Single AWS region
- Two Direct Connect locations
- One corporate data center connecting to both

Maximum Resiliency

- Same AWS region
- Two separate Direct Connect connections:
  - Terminate on different routers/devices
  - Located at different AWS Direct Connect facilities
- Each connection links to its own corporate data center infrastructure

Use Case

- Ensures **business continuity** and **network fault tolerance** for enterprise-grade workloads.

Site-to-Site VPN Connection as a Backup for Direct Connect

Purpose

- Provides a **redundant failover path** in case the **Direct Connect (DX)** primary connection fails.
- Helps maintain **connectivity between your on-premises data center and AWS** during outages.

Options for Redundancy

1. **Backup Direct Connect Connection**
   - Offers high availability and performance
   - **More expensive** to implement
2. **Site-to-Site VPN Connection**
   - **Cost-effective** alternative for backup
   - Uses the public internet but provides **encrypted IPsec tunnels**
   - Automatically takes over if Direct Connect becomes unavailable

Architecture Summary

- **Primary Path**: Direct Connect between corporate data center and AWS VPC
- **Backup Path**: Site-to-Site VPN connection that activates upon DX failure

Best Practice

- Configure **dynamic routing** (e.g., using BGP) to facilitate seamless failover between Direct Connect and VPN.

Complex Network Topologies in AWS

Overview

As AWS architectures grow, **network topologies can become complex**, especially when multiple connectivity options and VPCs are involved.

Common Connectivity Elements

- **Customer Gateway (CGW)**:
    - On-premises VPN endpoint for establishing Site-to-Site VPN connections.
- **VPN Connection**:
    - Encrypted tunnel from CGW to AWS, used for hybrid connectivity.
- **VPC Peering Connection**:
    - Private connectivity between two VPCs.
    - Not transitive—each peering must be explicitly defined.
- **Direct Connect Gateway (DXGW)**:
    - Enables private connectivity from on-premises networks to multiple VPCs across regions.

Architectural Challenges

- Multiple **VPC Peering connections** create a **mesh** topology that can be difficult to scale and manage.
- Combining **VPN**, **Direct Connect**, and **peering** can introduce complexity in:
    - **Routing configuration**
    - **Security management**
    - **Troubleshooting and monitoring**

Recommendation

- Evaluate when to use **Transit Gateway** or **Direct Connect Gateway** to simplify architecture and centralize control.

AWS Transit Gateway

Purpose

- **Transit Gateway** enables **transitive peering** between **thousands of VPCs and on-premises networks** using a **hub-and-spoke (star) architecture**.

Key Features

- **Regional resource**:
    - Can operate across Availability Zones within a region
    - Can also **peer across regions**
- **Cross-account sharing**:
    - Use **AWS Resource Access Manager (RAM)** to share Transit Gateway across multiple AWS accounts
- **Routing Control**:
    - Uses **Route Tables** to control and **restrict which VPCs can communicate** with each other
- **Integration Support**:

- Works with:
    - **AWS Direct Connect Gateway**
    - **Site-to-Site VPN Connections**
    - **Customer Gateways**

- **Unique Capabilities**:

  - **Supports IP Multicast**, which is **not supported by any other AWS service**

Architecture Summary

- Multiple **Amazon VPCs** connect to a **Transit Gateway**
- On-premises connectivity via **VPN Connection** or **Direct Connect Gateway**
- Centralized hub simplifies complex topologies and enhances scalability

Transit Gateway: Site-to-Site VPN with ECMP

What is ECMP?

- **ECMP** stands for **Equal-Cost Multi-Path routing**.
- It's a routing strategy that allows packets to be forwarded over **multiple best paths** simultaneously.

Use Case

- By creating **multiple Site-to-Site VPN connections** between your **corporate data center** and **AWS Transit Gateway**, you can:
  - **Distribute traffic** across those connections
  - **Increase total available bandwidth**
  - **Improve fault tolerance** and **throughput**

Architecture Overview

- A **Transit Gateway** connects to multiple **VPCs** using **VPC attachments**.
- The **corporate data center** connects via **VPN attachments** with multiple tunnels enabled.
- All VPN connections share the same cost in the routing table, enabling **ECMP-based load balancing**.

Benefits

- Efficient use of **parallel VPN tunnels**
- No need to upgrade a single VPN connection to higher bandwidth
- Supports **scalable and redundant hybrid architectures**

Transit Gateway: Throughput with ECMP

VPN to Virtual Private Gateway (VGW)

- **1 VPN connection** (with 2 tunnels)
- Bandwidth: **1.25 Gbps**

VPN to Transit Gateway (TGW) with ECMP

- **1x VPN connection** with ECMP:

  - Bandwidth: **2.5 Gbps** (2 tunnels used in parallel)

- **2x VPN connections** with ECMP:

  - Bandwidth: **5.0 Gbps**

- **3x VPN connections** with ECMP:
  - Bandwidth: **7.5 Gbps**

Key Takeaways

- **Transit Gateway + ECMP** allows **scalable bandwidth** by aggregating multiple VPN connections.
- Enables **high-throughput hybrid connectivity** without requiring a single large bandwidth pipe.
- Costs are based **per GB of Transit Gateway processed data**.

Use Case

Ideal for customers needing:

- **High throughput VPN connectivity**
- **Cost-effective scaling** using multiple tunnels instead of upgrading to Direct Connect

Transit Gateway – Share Direct Connect Between Multiple Accounts

Objective

- Enable **multiple AWS accounts** to **share a single Direct Connect connection** using **Transit Gateway** and **Resource Access Manager (RAM)**.

Architecture Components

- **Corporate Data Center**:
  - Connected to AWS via **Direct Connect** and a **customer router/firewall**
- **AWS Direct Connect Location**:
  - Houses the **Direct Connect Endpoint**
  - Uses a **Transit Virtual Interface (Transit VIF)** over a **VLAN**
- **Direct Connect Gateway (DXGW)**:
  - Acts as an intermediate gateway between Direct Connect and Transit Gateway
- **Transit Gateway (TGW)**:
  - Connects to multiple **VPCs** from different **AWS accounts**
  - Allows shared access using **AWS RAM**

Benefits

- **Centralized and scalable connectivity**
- **Cost-efficient**: single DX line shared across accounts
- **Simplified network management** for multi-account environments

VPC – Traffic Mirroring

Purpose

- **Traffic Mirroring** allows you to **capture and inspect network traffic** within your VPC.
- Useful for **security analysis**, **performance troubleshooting**, and **network monitoring**.

How It Works

- **Source**: Elastic Network Interfaces (ENIs) attached to EC2 instances or Auto Scaling Groups.

- **Target**:
  - Another ENI (e.g., attached to a monitoring instance)
  - Or a **Network Load Balancer** forwarding traffic to security appliances

Features

- You can **mirror all packets** or use **filters** to capture only specific types of traffic.
- Supports **packet truncation** to reduce volume.
- **Source and target** can reside in:
  - The **same VPC**
  - **Different VPCs** using **VPC Peering**

Use Cases

- **Content inspection**
- **Threat detection and monitoring**
- **Troubleshooting network issues**

Example Setup

- An Auto Scaling group of EC2 instances sends mirrored traffic to:
  - A Network Load Balancer
  - Which routes traffic to EC2 instances running **security appliances** for analysis

What is IPv6?

IPv4 Limitations

- **IPv4** was designed to support approximately **4.3 billion unique addresses**.
- Due to the growth of the internet, these addresses are being **exhausted**.

IPv6 – The Successor

- **IPv6** was developed to overcome IPv4 limitations.
- Capable of providing **3.4×10^{38} unique IP addresses**.

Characteristics in AWS

- Every **IPv6 address** is:
  - **Public**
  - **Internet-routable**
  - There is **no private IPv6 range** in AWS

IPv6 Format

- Consists of **8 segments** separated by colons `:`
- Each segment is a **hexadecimal** value (`0000` to `ffff`)
- Standard format: `x:x:x:x:x:x:x:x`

Examples

- Full address:
  `2001:db8:3333:4444:5555:6666:7777:8888`

- Another full address:
  `2001:db8:3333:4444:cccc:dddd:eeee:ffff`

- Shortened notation:
  - `::` → All 8 segments are zero
  - `2001:db8::` → Last 6 segments are zero
  - `::1234:5678` → First 6 segments are zero
  - `2001:db8::1234:5678` → Middle 4 segments are zero

IPv6 in VPC

Dual-Stack Networking

- In AWS, **IPv4 cannot be disabled** for your **VPC** or **subnets**.
- You can enable **IPv6** to operate in **dual-stack mode**, allowing the use of both IPv4 and IPv6.

EC2 Instance Addressing

- Each EC2 instance will receive:
  - A **private internal IPv4 address**
  - A **public IPv6 address**
- Instances can communicate with the **internet** using:
  - IPv4 via NAT or Internet Gateway
  - IPv6 directly via **Internet Gateway** (no NAT required)

Example

- **EC2 Instance IPs**:
  - IPv4: `10.0.0.5`
  - IPv6: `2001:db8::ff00:42:8329`
- Connected through an **Internet Gateway**
- Supports **outbound and inbound** IPv4/IPv6 traffic

Summary

- Enabling IPv6 allows modern, scalable, and internet-routable address space.
- VPCs with dual-stack enable flexibility in communication and compatibility.

IPv4 Troubleshooting in VPC

Key Point

- **IPv4 cannot be disabled** in AWS for VPCs or subnets.
- When you **cannot launch an EC2 instance**, it's **not due to IPv6 availability**, but rather **IPv4 exhaustion** in your subnet.

Common Problem

- Subnet runs out of **available IPv4 addresses**.
- Even with **IPv6 enabled**, AWS still requires assigning an **IPv4 address** to launch an EC2 instance.

Solution

- **Expand the subnet** by adding a **new IPv4 CIDR block** to the VPC.

Example

- VPC CIDRs:

  - IPv4: `192.168.0.0/24`
  - IPv4: `10.0.0.0/24`
  - IPv6: `2001:db8:1234:5678::/56`

- Used IPs:

  - `192.168.0.10`
  - `192.168.0.15`
  - `10.0.0.35`

- If no IPv4 addresses are left in a subnet, instance creation will fail.

Recommendation

- Monitor **IPv4 utilization** in your subnets.
- **Plan subnet sizing** to avoid early exhaustion.
- **Add secondary IPv4 CIDR blocks** if necessary.

Egress-only Internet Gateway

Purpose

- Designed **exclusively for IPv6 traffic**
- Functions similarly to a **NAT Gateway**, but for **IPv6**

Behavior

- Allows instances in a **private subnet** to **initiate outbound IPv6 connections** to the internet.
- **Blocks unsolicited inbound IPv6 traffic** from the internet to those instances.

Key Points

- Must be explicitly **added to the Route Table** to route IPv6 traffic.
- Useful for securing **IPv6-enabled workloads** in **private subnets** while still allowing them to access the internet.

Architecture Summary

- **Public Subnet**: Uses a standard **Internet Gateway** for full bidirectional IPv6 connectivity.
- **Private Subnet**: Routes IPv6 traffic through an **Egress-only Internet Gateway**
  - Instances can reach out to the internet
  - The internet **cannot initiate** connections back

Example IPv6 Addresses

- Public Subnet instance: `2001:db8::b1c2`
- Private Subnet instance: `2001:db8::e1c3`

IPv6 Routing – Example Architecture

VPC Configuration

- **VPC CIDRs**:
  - IPv4: `10.0.0.0/16`

- IPv6: `2001:db8:1234:1a00::/56`

Subnets

- **Public Subnet**
  - IPv4: `10.0.0.0/24`
  - IPv6: `2001:db8:1234:1a00::/64`
  - Contains web server with:
    - Private IPv4: `10.0.0.5`
    - Elastic IP: `198.51.100.1`
    - IPv6: `2001:db8:1234:1a00::123`
- **Private Subnet**
  - IPv4: `10.0.1.0/24`
  - IPv6: `2001:db8:1234:1a02::/64`
  - Contains backend server with:
    - Private IPv4: `10.0.1.5`
    - IPv6: `2001:db8:1234:1a02::456`

Internet Access

- **NAT Gateway** for IPv4 (in the public subnet)
  - Enables IPv4 access to the internet from private subnet
  - Uses Elastic IP: `198.51.100.1`
- **Internet Gateway**:
  - Supports both **IPv4 and IPv6**
  - Enables public subnet access to the internet
- **Egress-only Internet Gateway**:
  - Used for **outbound IPv6 traffic** from the private subnet
  - Prevents unsolicited inbound IPv6 connections

Route Tables

Public Subnet Route Table

Destination	Target
`10.0.0.0/16`	local
`2001:db8:1234:1a00::/56`	local
`0.0.0.0/0`	igw-id
`::/0`	igw-id

Private Subnet Route Table

Destination	Target

`10.0.0.0/16`	local
`2001:db8:1234:1a00::/56`	local
`0.0.0.0/0`	nat-gateway-id
`::/0`	eigw-id

VPC Section Summary (1/3)

Key Concepts

- **CIDR (Classless Inter-Domain Routing)**:
    - Defines an **IP range** for the VPC or subnet.
- **VPC (Virtual Private Cloud)**:
    - A **virtual network** defined by one or more **IPv4 and/or IPv6 CIDR blocks**.
- **Subnets**:
    - Divided sections of a VPC, each tied to a specific **Availability Zone (AZ)**.
    - Each subnet must have a defined CIDR.
- **Internet Gateway (IGW)**:
    - Attached at the **VPC level**.
    - Provides **IPv4 and IPv6 internet access** for instances in **public subnets**.
- **Route Tables**:
    - Control traffic routing.
    - Must be updated to direct subnet traffic to resources like:
        - **Internet Gateway**
        - **VPC Peering Connections**
        - **VPC Endpoints**

Network Access Components

- **Bastion Host**:
    - A **public EC2 instance** used for **SSH access** to EC2s in **private subnets**.
- **NAT Instance** (legacy):
    - Grants **internet access to private EC2 instances**.
    - Must be placed in a **public subnet**.
    - Requires **Source/Destination check** to be disabled.
- **NAT Gateway** (preferred):
    - AWS-managed service.
    - **Scalable and reliable** way to allow **IPv4 internet access** from **private subnets**.

VPC Section Summary (2/3)

Security and Access Control

- **NACL (Network ACL)**:
    - **Stateless**
    - Rules defined at the **subnet level** for both **inbound and outbound traffic**
    - Must allow **ephemeral ports** for return traffic
- **Security Groups**:
    - **Stateful**
    - Operate at the **EC2 instance level**
    - Automatically allow return traffic

Connectivity

- **VPC Peering**:
    - Direct connection between **two VPCs**
    - Requires **non-overlapping CIDRs**
    - **Not transitive**: each peering must be explicitly defined
- **VPC Endpoints**:
    - Enable **private access** to AWS services such as:
        - S3
        - DynamoDB
        - CloudFormation
        - Systems Manager (SSM)
    - Traffic stays within the AWS network

Monitoring and Analysis

- **VPC Flow Logs**:
    - Can be enabled at the **VPC**, **subnet**, or **ENI** level
    - Capture both **ACCEPT** and **REJECT** traffic
    - Useful for:
        - Security auditing
        - Attack detection
        - Analytics via **Athena** or **CloudWatch Logs Insights**

Hybrid Connectivity

- **Site-to-Site VPN**:
    - Connects on-premises **Customer Gateway** to AWS **Virtual Private Gateway**
    - Uses the **public internet** with **IPsec encryption**
- **AWS VPN CloudHub**:
    - **Hub-and-spoke VPN architecture**
    - Connects multiple on-premises sites to a single AWS VPC

VPC Section Summary (3/3)

Connectivity

- **Direct Connect**:
  - Establish a **dedicated private connection** to AWS via a **Virtual Private Gateway** attached to your VPC.
- **Direct Connect Gateway**:
  - Enables connecting **multiple VPCs across different AWS regions** to a single Direct Connect.
- **Transit Gateway**:
  - Supports **transitive routing** between VPCs, **VPNs**, and **Direct Connect** connections.
  - Scalable **hub-and-spoke architecture**.

PrivateLink / VPC Endpoint Services

- Used to **privately connect services** between VPCs:
  - Customer VPCs access services in the provider's **Service VPC**.
- No need for:
  - **VPC Peering**
  - **Public Internet**
  - **NAT Gateway**
  - **Route Table updates**
- Requires:
  - **Network Load Balancer (NLB)**
  - **Elastic Network Interfaces (ENIs)**

Other Features

- **ClassicLink**:
  - Allows legacy **EC2-Classic** instances to connect privately to a VPC.
- **Traffic Mirroring**:
  - Copies **network traffic from ENIs** for inspection, analysis, or threat detection.
- **Egress-only Internet Gateway**:
  - Allows **outbound IPv6** internet access from private subnets.
  - Blocks unsolicited **inbound IPv6 traffic**, similar to a NAT Gateway for IPv6.

Networking Costs in AWS per GB – Simplified

Intra-VPC and Intra-Region Traffic

- **Within the same Availability Zone** using **private IPs**:
  Free
- **Between Availability Zones in the same Region**:
  - Using **private IPs**:
    $0.01 per GB

    - Using **public IPs / Elastic IPs**:
      $0.02 per GB

Inter-Region Traffic

- **Between different AWS Regions**:
  $0.02 per GB

Best Practices for Cost Savings

- Prefer **private IPs** over public IPs:
    - Reduces data transfer costs
    - Improves network performance (lower latency)
- Deploy workloads in the **same Availability Zone** when high availability is not required:
    - **Eliminates cross-AZ data transfer charges**

Summary Table

Scenario	Cost per GB
Same AZ, private IP	Free
Cross-AZ, private IP	$0.01
Cross-AZ, public/Elastic IP	$0.02
Inter-region	$0.02

Minimizing Egress Traffic Network Cost

Definitions

- **Egress traffic**: Data sent **from AWS to external destinations** (charged).
- **Ingress traffic**: Data sent **to AWS from external sources** (typically **free**).

Cost Optimization Tips

- **Keep internet traffic within AWS** as much as possible to avoid high egress charges.
- Use **AWS services and resources located in the same region** to reduce data transfer costs.
- Consider **Direct Connect locations co-located within the same AWS region** to lower the egress cost.

Example Scenarios

Efficient Setup (Minimized Egress Cost)

- Database and application both hosted in **AWS Cloud**
- Only **small query results** (e.g., 50 KB) are sent to on-premises
- Large queries (e.g., 100 MB) remain within AWS

Inefficient Setup (High Egress Cost)

- Application is on-premises, querying a **database in AWS**
- Every large query result (e.g., 100 MB) needs to be transferred out
- Generates **high egress charges**

Recommendation

- For bandwidth-intensive workloads, **co-locate compute and storage in AWS**
- Use **Direct Connect** when consistent, low-cost outbound data transfer is required

S3 Data Transfer Pricing – USA Analysis

Basic Transfer Costs

- **S3 Ingress (uploads to S3)**:
  Free

- **S3 to Internet**:
  $0.09 per GB

- **S3 to CloudFront**:
  $0.00 per GB (free)

- **CloudFront to Internet**:
  $0.085 per GB

  - Slightly **cheaper than direct S3 to Internet**
  - Benefits from **edge caching** (lower latency and cost)

S3 Transfer Acceleration

- **Provides faster upload/download speeds** (50–500% improvement)
- Adds **$0.04–$0.08 per GB** on top of standard S3 transfer rates

Cross-Region Replication

- **S3 Cross-Region Replication**:
  $0.02 per GB

Optimization Tips

- Use **CloudFront**:

  - Reduces cost compared to direct S3 access
  - Minimizes **S3 request pricing**
  - Provides **better performance via caching**

- Use **Transfer Acceleration** only when necessary:

  - Ideal for **global uploads/downloads** where speed is a priority

Summary Table

Transfer Type	Cost per GB
S3 Ingress	Free
S3 to Internet	$0.09
S3 to CloudFront	$0.00
CloudFront to Internet	$0.085
Transfer Acceleration (add-on)	+$0.04–$0.08

Cross-Region Replication	$0.02

Pricing: NAT Gateway vs Gateway VPC Endpoint

VPC Configuration

- **Region**: `us-east-1`
- **VPC CIDR**: `10.0.0.0/16`
- **Subnets**:
  - **Public Subnet**: contains **NAT Gateway**
  - **Private Subnet 1**: `10.0.0.0/24`, accesses S3 via **VPC Endpoint**
  - **Private Subnet 2**: `10.0.1.0/24`, accesses internet via **NAT Gateway**

Route Table Configuration

Public Subnet Route Table

Destination	Target
`10.0.0.0/16`	local
`0.0.0.0/0`	igw-id

Private Subnet 1 Route Table (VPC Endpoint)

Destination	Target
`10.0.0.0/16`	local
`pl-id` (S3)	`vpce-id`

Private Subnet 2 Route Table (NAT)

Destination	Target
`10.0.0.0/16`	local
`0.0.0.0/0`	nat-gateway-id

Cost Comparison

NAT Gateway

- **$0.045 per hour**
- **$0.045 per GB** of data processed
- **$0.09 per GB** for data transfer out to **S3 (cross-region)**
- **$0.00 per GB** for data transfer out to **S3 (same-region)**

Gateway VPC Endpoint (for S3)

- **No hourly cost**
- **$0.01 per GB** of data transfer in/out (same-region)

Key Takeaways

- For **S3 access from private subnets**, using a **Gateway VPC Endpoint** is significantly **cheaper** than routing through a **NAT Gateway**.
- Avoid using NAT for S3 access when possible to **optimize cost** and **improve performance**.

Network Protection on AWS

Available Protection Mechanisms

- **Network Access Control Lists (NACLs)**:
    - Stateless firewall rules applied at the **subnet level**.
- **Amazon VPC Security Groups**:
    - Stateful firewalls applied at the **instance level**.
- **AWS WAF (Web Application Firewall)**:
    - Protects against **malicious HTTP(S) requests** (e.g., SQL injection, XSS).
    - Typically used with **CloudFront**, **ALB**, or **API Gateway**.
- **AWS Shield & AWS Shield Advanced**:
    - **DDoS protection** for AWS resources.
    - Shield Advanced provides **enhanced detection and mitigation**, plus support.
- **AWS Firewall Manager**:
    - Centralized management of **WAF**, **Shield**, and **security groups** across **multiple accounts** and **resources** in an organization.

Key Question

- What if you want to protect your **entire VPC in a sophisticated and centralized way**?

(This leads to the next topic: AWS Network Firewall)

AWS Network Firewall

Purpose

- Provides **advanced network protection** for your **entire Amazon VPC**.
- Covers traffic inspection from **Layer 3 (Network Layer)** to **Layer 7 (Application Layer)**.

Supported Traffic Directions

- **VPC to VPC** (e.g., via peering)
- **Outbound** traffic to the **Internet**
- **Inbound** traffic from the **Internet**
- Traffic **to/from**:
    - **AWS Direct Connect**
    - **Site-to-Site VPN**
    - **Corporate data centers**

Architecture

- Deployed **within a VPC** in a **dedicated subnet**
- Internally uses **AWS Gateway Load Balancer** for scalability and availability

Centralized Management

- Integrates with **AWS Firewall Manager**
  - Allows **cross-account** rule management
  - Enables enforcement of **consistent security policies** across **multiple VPCs**

Use Case

Ideal for organizations needing:

- **Sophisticated inspection and control** of all VPC network traffic
- **Centralized rule management** for large or multi-account AWS environments

AWS Network Firewall – Fine-Grained Controls

Rule Capabilities

- Supports **thousands of rules** for highly customizable traffic filtering.

Rule Types

- **IP & Port Filtering**:

  - Example: block or allow traffic from **10,000+ IPs**.

- **Protocol Filtering**:

  - Example: block **SMB protocol** for outbound communication.

- **Stateful Domain List Rule Groups**:

  - Example: allow only outbound traffic to `*.mycorp.com` or to **specific third-party software repositories**.

- **Pattern Matching**:

  - Use **regular expressions (regex)** for identifying specific data patterns in traffic.

Actions

- Traffic can be:
  - **Allowed**
  - **Dropped**
  - **Alerted** on (for logging and monitoring)

Intrusion Prevention

- Performs **active flow inspection** to detect and block network threats.
- Provides **intrusion-prevention capabilities** (similar to Gateway Load Balancer, but fully **managed by AWS**).

Logging

- Logs of matched rules can be sent to:
  - **Amazon S3**
  - **CloudWatch Logs**
  - **Amazon Kinesis Data Firehose**

Summary

AWS Network Firewall offers **deep, flexible, and scalable control** over VPC traffic, suitable for advanced **security monitoring and threat mitigation**.