AMAZON VPC (VIRTUAL PRIAVTE CLOUD)

What is Amazon VPC?

Amazon Virtual Private Cloud (VPC) is a logically isolated virtual network inside AWS where you launch and run your cloud resources. It is the networking foundation of AWS. Every EC2 instance, ECS service, EKS pod, RDS database, or load balancer runs inside a VPC.

Conceptually, a VPC is very similar to a traditional on-premises data center network. You define the IP address space, divide it into subnets, control routing, manage internet access, and apply security rules. The difference is that AWS fully manages the underlying infrastructure while giving you software-defined control over the network behavior.

A VPC gives you:

- Isolation from other AWS customers
- Full control over IP addressing
- Control over inbound and outbound traffic
- Integration with AWS security and identity services
- The ability to connect to on-premises networks and other clouds

If EC2 is the “computer,” **VPC is the “network that makes everything possible.”**

Real-life analogy

Imagine AWS as a **huge city**:

- **AWS City** → Amazon's global infrastructure
- **VPC** → Your private gated community
- **Subnets** → Streets or blocks (public vs private)
- **EC2 instances** → Houses (your servers)
- **Load Balancer** → Main entrance with traffic control

- **Security Groups** → Security guards at each house
- **Internet Gateway** → The gate that connects your neighborhood to the outside world

You control all the rules. AWS just provides the city.

Why VPC Knowledge Is Critical

DevOps engineers don't just deploy applications, they design **systems that are secure, scalable, and resilient**. Almost all production issues in AWS eventually touch networking in some way:

- "Why can't my service reach the database?"
- "Why does traffic not reach the load balancer?"
- "Why can't this private instance access the internet?"
- "Why is this port blocked?"

Understanding VPC allows you to:

- Design highly available architectures
- Prevent accidental public exposure
- Reduce blast radius during failures
- Troubleshoot connectivity issues quickly
- Build secure multi-account cloud platforms

Poor VPC design often leads to expensive refactoring later, so learning it **properly and early** is essential.

Key VPC building blocks

- **CIDR Block** → Your network's address range (like choosing a ZIP code for your neighborhood)
- **Subnets** → Smaller sections of your VPC (public streets vs private streets)

- **Internet Gateway (IGW)** → The door to the internet
- **Route Tables** → GPS rules for traffic ("Where should this packet go?")
- **Security Groups** → Firewall at the resource level (stateful)
- **NACLs (Network ACLs)** → Firewall at the subnet level (stateless)
- **NAT Gateway** → Allows private resources to access the internet without being exposed

VPC IP Addressing and CIDR Blocks

When you create a VPC, you assign it a **CIDR block**. This defines the **entire IP address space** available to the VPC.

Example:
10.0.0.0/16

This means:

- The VPC has ~65,536 IP addresses
- You can divide this space into smaller networks (subnets)
- The CIDR block **cannot be changed later**

CIDR notation consists of:

- An IP address
- A prefix length (/16, /24, etc.)
  The prefix length defines how many bits represent the network portion.

Choosing the CIDR block is one of the **most important decisions** in VPC design because:

- You must support future growth
- Overlapping CIDRs prevent peering and hybrid connections
- Renumbering later is extremely difficult

RFC 1918 and Private IP Addressing

AWS VPCs typically use **private IP ranges defined by RFC 1918**, a networking standard created to allow organizations to use internal IP addresses without conflicting with the public internet.

The RFC 1918 private ranges are:

- 10.0.0.0/8 — very large networks
- 172.16.0.0/12 — medium-sized networks
- 192.168.0.0/16 — smaller networks

These IP addresses:

- Are **not routable on the public internet**
- Can be reused across different organizations
- Are ideal for internal cloud networking

In AWS, almost all production VPCs use these ranges.

Subnets: Dividing the Network

A subnet is a **smaller network inside a VPC**. Subnets allow you to organize resources, control routing, and isolate workloads.

Important rules:

- Each subnet belongs to **one Availability Zone**
- Subnets cannot span AZs
- AWS reserves **5 IP addresses per subnet**

Subnets are commonly categorized as:

- **Public subnets** — have a route to the Internet Gateway
- **Private subnets** — no direct internet route

In real architectures:

- Public subnets host internet-facing components (ALB, NAT Gateway)
- Private subnets host application servers and databases

Using multiple subnets across AZs is how AWS achieves **high availability**.

Internet Gateway (IGW)

An Internet Gateway connects a VPC to the public internet. It enables:

- Inbound internet traffic (if allowed)
- Outbound internet traffic

An Internet Gateway:

- Is highly available by default
- Supports IPv4 and IPv6
- Does not impose bandwidth limits
- Is attached at the VPC level (one per VPC)

A subnet is **not public by default**. It becomes public only when:

1. The subnet's route table includes a route to an IGW
2. Instances have public or Elastic IP addresses

Route Tables: Controlling Traffic Flow

Route tables define **how traffic moves** inside and outside the VPC. Each route consists of:

- A destination CIDR
- A target (IGW, NAT Gateway, Transit Gateway, etc.)

Every subnet must be associated with a route table. AWS provides a **main route table**, but best practice is to create **custom route tables**.

Typical routes include:

- Local VPC traffic (10.0.0.0/16 → local)
- Internet traffic (0.0.0.0/0 → igw)
- Private outbound traffic (0.0.0.0/0 → nat-gateway)

Route tables are the **decision engine** of VPC networking.

NAT Gateway: Private Internet Access

A NAT Gateway allows instances in **private subnets** to access the internet **without being reachable from the internet**.

Key characteristics:

- Outbound-only connectivity
- No inbound connections allowed
- Deployed in a specific AZ
- Requires an Elastic IP address

Common use cases:

- Software updates
- Downloading dependencies
- Calling external APIs

Best practice is to deploy **one NAT Gateway per AZ** to avoid cross-AZ traffic and single points of failure.

Security Groups and Network ACLs

AWS provides two layers of network security.

Security Groups

Security Groups act as **stateful firewalls** at the instance or service level.

- Allow rules only
- Automatically allow return traffic
- Primary security mechanism in AWS

Network ACLs (NACLs)

NACLs are **stateless firewalls** applied at the subnet level.

- Allow and deny rules
- Rules are evaluated in order
- Require explicit return rules

In practice:

- Security Groups handle most access control
- NACLs provide coarse-grained protection and additional safeguards

VPC Peering

VPC Peering allows two VPCs to communicate privately using their private IP addresses.

Characteristics:

- Low latency

- No bandwidth bottleneck
- Cross-account and cross-region
- No transitive routing

Peering is simple but becomes difficult to manage at scale because every VPC must connect directly to every other VPC it needs to reach.

Transit Gateway: Scalable Networking Hub

Transit Gateway solves the scaling problem of VPC peering by acting as a **central router**.

It allows:

- Thousands of VPC connections
- Transitive routing
- Centralized traffic inspection
- Clean hub-and-spoke architectures

Transit Gateway is commonly used in:

- Large organizations
- Multi-account environments
- Hybrid cloud architectures
- Shared services models

On-Premises and Hybrid Connectivity

AWS supports hybrid networking using:

Site-to-Site VPN

- Encrypted IPSec tunnels

- Internet-based
- Fast to set up
- Lower bandwidth

Direct Connect

- Dedicated physical connection
- Predictable performance
- Lower latency
- Often combined with VPN for encryption

Hybrid connectivity allows organizations to gradually migrate workloads to AWS.

VPC Endpoints and Private AWS Access

VPC Endpoints allow private access to AWS services without using the internet.

Gateway Endpoints

Used for:

- Amazon S3
- DynamoDB

Interface Endpoints (PrivateLink)

- ENI-based
- Secure, private service access
- Eliminates need for NAT or IGW

Endpoints significantly improve security posture and reduce data transfer costs.

Monitoring and Troubleshooting

VPC Flow Logs

VPC Flow Logs capture metadata about network traffic and are used for:

- Debugging connectivity
- Security investigations
- Compliance auditing

Common issues DevOps engineers troubleshoot:

- Missing route table entries
- Incorrect subnet association
- Security Group vs NACL conflicts
- NAT Gateway in wrong AZ
- DNS resolution disabled

Common VPC Mistakes

- Putting everything in public subnets
- Confusing public IPs with public subnets
- Forgetting to configure route tables
- Opening security groups too wide (0.0.0.0/0)
- Exposing databases to the internet
- Mixing dev, test, and prod in the same VPC
- Designing CIDR ranges too small
- Overlapping CIDR blocks
- Ignoring multi-AZ design
- Relying only on security groups and ignoring NACLs
- Treating VPC as “just networking”
- Copy-pasting architectures without understanding them

What happens if you don't design VPC properly?

You may face:

- Security breaches (exposed databases, open ports)
- Compliance failures (SOC2, HIPAA, PCI-DSS)
- Hard-to-debug network issues
- Poor scalability and availability
- Unexpected costs (data transfer between regions/AZs)

VPC Best Practices

Production-grade VPC design includes:

- Multi-AZ subnet layouts
- Clear public/private separation
- Proper CIDR planning
- Least-privilege security rules
- Centralized networking (Transit Gateway)
- Enabled Flow Logs
- No public databases

Official AWS VPC Resources

- **VPC Main Page**: https://aws.amazon.com/vpc/
- **VPC Documentation**: https://docs.aws.amazon.com/vpc/
- **VPC User Guide**: https://docs.aws.amazon.com/vpc/latest/userguide/
- **Hands-On Tutorial**: https://docs.aws.amazon.com/vpc/latest/userguide/vpc-tutorials-intro.html