MODULE 7 · STUDY NOTES

Creating a Networking Environment

Building a VPC from the ground up — subnets and route tables, internet and NAT gateways, security groups and network ACLs, private links to managed services, flow logs, and the Well-Architected principles that tie them together

This module builds an AWS network from nothing but an empty account. The Amazon Virtual Private Cloud (VPC) service gives you a programmatically defined, logically isolated slice of an AWS Region, and every decision that follows is about controlling how traffic moves into, out of, and within it. You size the VPC with a CIDR block and divide it into subnets whose route tables make them public or private; you connect the public tier to the internet with an internet gateway and give the private tier outbound-only access through a NAT device. You then secure everything with layered controls — stateful security groups at the resource level, stateless network ACLs at the subnet level, plus AWS Network Firewall and bastion hosts — link privately to managed services such as Amazon S3 with VPC endpoints, monitor and troubleshoot with VPC Flow Logs and analyzer tools, and finally judge the design against the AWS Well-Architected Framework.

LEARNING OBJECTIVES

- Explain the role of a virtual private cloud (VPC) in AWS Cloud networking
- Identify the components in a VPC that can connect an AWS networking environment to the internet
- Isolate and secure resources within your AWS networking environment
- Create and monitor a VPC with subnets, an internet gateway, route tables, and a security group
- Use the AWS Well-Architected Framework principles when creating and planning a network environment

7.1 Introducing Amazon VPC

Everything you build on AWS runs on physical infrastructure organized in a strict hierarchy. Thousands of servers sit in racks inside **data centers**; data centers are grouped into **Availability Zones (AZs)** connected by a wired, low-latency network with **single-digit millisecond** latency that supports synchronous data replication; and AZs are grouped into an **AWS Region** — a separate geographic area made up of **at least three** isolated, physically separate AZs, where latency between Regions runs into **tens of milliseconds**. An AZ's code is its Region code plus a letter, such as `us-east-1a`. On top of this physical network AWS builds a **virtual (overlay) network** of software-defined switches, routers, firewalls, and load balancers that you create and manage programmatically — and that virtual network is what the Amazon VPC service exposes.

ACCOUNT AND VPC ISOLATION

- A new AWS account gets **dedicated** network resources and **spans all Regions**.
- Each public-accessible Region contains a **default VPC** — a software-defined network already wired for connectivity.
- A VPC **belongs to one Region** and can **span multiple AZs**; a new custom VPC is **logically isolated** from other networks and the internet until you configure access.
- Amazon EC2 and Amazon RDS run **inside** a VPC; serverless services such as AWS Lambda and Amazon CloudWatch run **outside** the customer VPC and connect in.
- Don't use the default VPC for production — its built-in connectivity isn't hardened; define a new VPC with appropriate security and connectivity settings.

You **size a VPC by allocating a range of private IP addresses**, expressed as a Classless Inter-Domain Routing (**CIDR**) block: the maximum IPv4 VPC is a `/16` netmask (**65,536** addresses) and the smallest is a `/28` (**16** addresses). Choose generously — adding addresses later is difficult. A VPC can be **IPv4-only** or **dual-stack** (an IPv4 CIDR plus an IPv6 CIDR), and **Amazon VPC IP Address Manager (IPAM)** helps you plan and track addresses. IPv6 addresses are **128-bit** (versus **32-bit** IPv4) and don't use NAT — and since early 2024 **every public IPv4 address carries an hourly charge**, whether attached or not. Spread a multi-AZ VPC's addresses **evenly across the AZs**.

A VPC automatically comes with a **router** that uses a **main route table** of rules called **routes**. The main table's default route sends the VPC's own CIDR (for example destination `10.0.0.0/16`, target `local`) everywhere inside the VPC, so **every resource can reach every other resource** — and that `local` route **can't be deleted**. To carve up the VPC you create **subnets**: each subnet is a **container for routing policy** that **belongs to one AZ**, takes a **non-overlapping** segment of the VPC range (for example a `/20` with 4,096 addresses), and should be significantly smaller than the VPC.

TABLE 7.1 AWS reserves five addresses in every subnet (example 10.0.0.0/20)

ADDRESS	RESERVED FOR
10.0.0.0	Network address
10.0.0.1	VPC local router
10.0.0.2	DNS resolution
10.0.0.3	Future use
10.0.15.254	Network broadcast address

To expose resources to the internet, pair a subnet with an **internet gateway (IGW)** — a horizontally scaled, redundant, highly available VPC component that supports both IPv4 and IPv6. An IGW does two jobs: it provides a **target in your route tables** for internet-routable traffic, and it performs **NAT** for instances that have public IPv4 addresses. Attach the IGW to the VPC (it is a **Regional** resource and attaches to only **one VPC**), give the instance a **public IP address**, and add a route to the **public subnet route table**: destination `0.0.0.0/0` (all traffic), target the internet gateway ID. That subnet-level table **overrides** the main route table for its subnet. Public IPv4 addresses aren't tied to your account and are released back to AWS when not in use.

ELASTIC IP ADDRESSES

- A static, public **IPv4** address you associate with an EC2 instance — unlike an instance's **dynamic private IP**, which is released when the instance terminates.
- If an instance becomes unhealthy, **transfer** the Elastic IP to a new, healthy instance to keep the same address.
- The **first** Elastic IP on a **running** instance is free; you're charged for **additional** ones, for one on a **stopped** instance, or for one that's **detached and unused**.
- Can also be associated with a **load balancer** or a **VPC network interface**.

For resources that must not be reachable from the internet, use a **private subnet** — one whose route table has **no route to an internet gateway**. It's a best practice to give **every subnet its own custom route table**; a private subnet's table can be identical to the main table (just the `local` route). If you don't explicitly associate a route table, the subnet uses the **main** route table by default. A subnet can be associated with **only one route table at a time**, but one route table can serve **many subnets**.

HOW A NAT DEVICE REWRITES ADDRESSES

A NAT device hides a private IP behind its own public IP. Say a source server at private `10.0.0.17` needs to reach a destination server at public `190.50.20.87`: (1) the source sends the request with source IP `10.0.0.17` to the NAT device; (2) the NAT device swaps in its **own** public IP `89.89.0.100` and forwards the request; (3) the destination replies to `89.89.0.100`; (4) the NAT device maps that back to `10.0.0.17` and forwards the reply to the source. The destination never learns the private address.

Sometimes private resources legitimately need **outbound** internet access — downloading OS patches, for example — while staying undiscoverable. A **NAT device** provides exactly that: private-subnet instances can reach the internet, other VPCs, or on-premises networks, but **can't receive unsolicited connection requests**. AWS offers two forms: a **NAT gateway** (AWS-managed, **hourly cost**) or a **NAT instance** (your own EC2 instance, **EC2 costs**). Route the private subnet's `0.0.0.0/0` to the **NAT gateway ID**, and place the NAT device in a **public** subnet so it can reach the internet gateway. **AWS recommends NAT gateways over NAT instances** for better availability, bandwidth, and less administration; deploy **one per AZ** for resilience. For IPv6, use an **egress-only internet gateway** — outbound IPv6 only, no inbound.

TABLE 7.2 Choose the right subnet for each use case

RESOURCE / USE CASE	SUBNET	WHY
Database instances	Private	Must be isolated from the internet
Batch-processing instances	Private	No need for inbound internet access
Web application instances	Public or private	AWS prefers private behind a load balancer; public only if an Elastic IP is attached directly
NAT gateway or instance	Public	Needs access to the internet gateway

COMMON PITFALL

A public subnet by itself is **not** security. Placing resources in a public subnet only makes them reachable — it does nothing to protect them. You still need security groups, network ACLs, and least-privilege routing, which the next section covers.

7.2 Securing network resources

Reachability is not security. Once your network is connected, isolate and protect your workloads with **multiple layers of defense**. Traffic from a client can be encrypted in transit with a **secure network protocol** such as **TLS/HTTPS** so third parties can't read or impersonate it, and then it passes through a series of VPC controls — the **route table**, the **network ACL**, the **subnet**, and finally the **security group** — before reaching an EC2 instance. Using **both** security groups and network ACLs is defense in depth: a misconfiguration in one control won't expose the instance on its own.

SECURITY GROUP = YOUR APARTMENT DOOR	NETWORK ACL = THE BUILDING DOORMAN
protects the resource ▪ Only the tenant with the key gets in ▪ Acts as a firewall for a **resource** — an EC2 instance or network interface	*protects the subnet* ▪ Only people who live in the building get past the lobby ▪ Acts as a firewall for a **subnet**

Security groups are **stateful** virtual firewalls that act at the level of an **instance or network interface** and can span multiple AZs. You define **separate inbound and outbound rule sets**, choosing the ports and protocols to allow — **allow rules only, never deny rules**. Group resources that share security requirements together, and you can even write rules that **reference another security group** (for example, a database tier that accepts traffic only from the application tier's security group, or a web tier that allows HTTPS/TCP on port `443` from a load balancer's security group). Being **stateful** means **return traffic is automatically allowed**: if an inbound rule permits an HTTPS request, the response flows back out even if your outbound rules would otherwise restrict it. A **new** security group has **no inbound rules** (so no inbound traffic until you add some) and **one outbound rule allowing all outbound traffic**; remove it and, with no outbound rules, no outbound traffic is allowed.

A **network ACL** is an **optional, stateless** layer of security for **one or more subnets**. Every subnet **must** be associated with one; if you don't choose it, the VPC's **default** network ACL applies. A network ACL can serve many subnets, but a subnet has **exactly one** at a time (a new association removes the old). Rules are **numbered**, evaluated **in order**, and can **allow or deny**. The **default** network ACL **allows all** inbound and outbound traffic — a numbered allow-all rule (`100`) plus a catch-all **deny-all** rule (shown as an asterisk) that traps anything a misconfigured rule lets slip. A **custom** network ACL starts with only that catch-all deny rule, so you add numbered rules for specific IPs or CIDRs (for example, inbound rule `100` allowing HTTPS/TCP from `188.7.55.9/32` on port `443`). Being **stateless**, it keeps no connection state — **return traffic must be explicitly allowed**.

TABLE 7.3 Comparing security groups and network ACLs

BEHAVIOR	SECURITY GROUP	NETWORK ACL
Scope	Resource level (instance / ENI)	Subnet level
Rule types	Allow rules only	Allow and deny rules
State	Stateful — response auto-allowed	Stateless — response evaluated against rules
Rule evaluation	All rules evaluated	Numbered order; stops at first match
New / default inbound	No inbound allowed (new SG)	All inbound allowed (default NACL)
New / default outbound	All outbound allowed	All outbound allowed

AWS recommends using security groups over network ACLs where possible for ease of maintenance and flexibility. Reach for network ACLs when you need subnet-wide **deny** rules or an extra, independent layer of control. Two more tools extend this defense-in-depth model for demanding workloads.

AWS NETWORK FIREWALL

- A **stateful, managed** network firewall and intrusion detection/prevention service for a VPC — an extra layer beyond security groups and network ACLs.
- Deployed in a dedicated **firewall subnet** that inspects all incoming VPC traffic.
- You **modify route tables** to send traffic through the firewall endpoints (for example: IGW → route table 1 → firewall endpoint → route table 2 → private-subnet instance).
- Aimed at workloads handling **sensitive data or strict compliance** requirements.

BASTION HOST FOR PRIVATE-SUBNET ADMINISTRATION

A **bastion host** is a hardened EC2 instance in a **public** subnet that provides **maintenance access** to private-subnet resources instead of exposing them directly. To allow SSH to a Linux instance: **Security Group A** on the bastion allows inbound **SSH (TCP port 22)** from a specific administrator **IP address range**; **Security Group B** on the private application instance allows inbound **SSH (TCP port 22)** only from **Security Group A**. The bastion should be the **only** source of SSH to the private instance, which minimizes the attack surface while preserving admin access. Bastion instances also carry IAM policies granting access to VPC resources.

COMMON PITFALL

Don't confuse the two firewalls' defaults. A **new security group** blocks inbound but allows **all outbound**; a **default network ACL** allows **everything** in both directions. Assuming a fresh network ACL is closed like a security group is a common — and dangerous — mistake.

7.3 Connecting to managed AWS services

Suppose a workload on an EC2 instance in a **private subnet** needs an **Amazon S3** bucket in the same account and Region. S3 is a **managed service that operates outside your VPC**, so there is **no direct connectivity** from the instance to the bucket. You could reach S3 through its **public Region access point**, but that routes the request **over the internet** and incurs **data transfer costs** — giving up the privacy of the VPC. A better answer is a VPC endpoint.

A **VPC endpoint** provides a direct, private path from your VPC to a managed service, and there are **two types**. **Interface endpoints** are provided by **AWS PrivateLink**, the service designed to connect VPCs to managed AWS services. For every interface endpoint, AWS creates an **elastic network interface (ENI)** — a logical virtual network card — in every specified subnet and gives it a **private IP** from that subnet's range; your instance then reaches the service **as if it were inside your VPC**. You govern access with an **IAM resource policy**, and interface endpoints are **priced** for hourly usage plus data processed per month.

SETTING UP AN INTERFACE ENDPOINT (VPC CONSOLE)

- **Specify the service** — an AWS service, endpoint service, or AWS Marketplace service to connect to.
- **Choose the VPC**, and specify **subnets in multiple AZs** so the endpoint is resilient to AZ failure — an ENI is created in each.
- **Choose the subnet**; a network interface with a private IP becomes the entry point for traffic to the service.
- **Attach security groups** to control traffic to the interface; if none is specified, the VPC's default security group is used.
- Services **can't initiate** requests into your VPC through the endpoint — it only returns responses to traffic your VPC starts.

Gateway endpoints take a different approach: they connect **directly** to a service **using route tables** rather than PrivateLink, and support **only Amazon S3 and Amazon DynamoDB**. You add a route whose **destination is a prefix list ID** (a named group of CIDR blocks) and whose **target is the gateway endpoint**. They carry **no additional charge** and have **no throughput or packet limits**. Because S3 supports both types, the choice comes down to the factors below.

TABLE 7.4 Amazon S3 endpoint considerations

FACTOR	INTERFACE ENDPOINT	GATEWAY ENDPOINT
S3 access point	Private IP from a VPC subnet	S3 public IP addresses
On-premises access	Allowed	Not allowed
Other AWS Region	Allowed	Not allowed
Cost	Billed	Not billed
Bandwidth	Up to 10 Gbps per AZ, scales to 100 Gbps	No limit
Packet size	Maximum 8,500 bytes	No limit

GATEWAY LOAD BALANCER ENDPOINT

- A specialized VPC endpoint that provides **private connectivity** between **security appliances** (in another VPC) and your application instances.
- Inbound traffic through the internet gateway is routed **first to the Gateway Load Balancer endpoint**, then to the **Gateway Load Balancer**, which distributes it to an **EC2 security appliance** for inspection.
- The inspected traffic returns to the endpoint and then to the **application instance**; outbound traffic follows the same path in reverse.

COMMON PITFALL

Gateway endpoints are **S3- and DynamoDB-only** and **free**; every other managed service uses an **interface endpoint**, which is **billed** and has bandwidth and packet limits. On "connect a private instance to a managed service cheaply" questions, a gateway endpoint wins for S3 or DynamoDB — but when you need on-premises or cross-Region access, it must be an interface endpoint.

7.4 Monitoring your network

A running VPC still has problems to diagnose, so monitor it and build automated recovery. Common scenarios point at specific culprits: **slow EC2 response times** can mean unwanted traffic — a **distributed denial of service (DDoS)** attack floods a server with bot messages until legitimate requests can't get through; a **failed SSH connection** usually means a **security group** isn't allowing port 22; and a **private database not receiving patches** points at its **security group, route tables, and NAT gateway** configuration. To see what traffic is actually flowing, use **VPC Flow Logs**.

VPC FLOW LOGS

- Capture **packet-level metadata** about traffic; you can scope logs to the whole **VPC**, a single **subnet**, or an **elastic network interface**.
- Choose to capture **all**, **accepted**, or **rejected** traffic.
- Deliver to **Amazon CloudWatch Logs** (filter and view in the console), **Amazon S3** (query with **Amazon Athena**; stored as plain text or Parquet), or **Amazon Kinesis Data Firehose** (on to **OpenSearch Service** or third-party tools such as Splunk).
- Flow logs run **outside** your VPC network path, so they **don't affect latency or performance**.

GRANTING FLOW-LOG PERMISSIONS WITH IAM

Users can't work with flow logs by default. Create an IAM role whose policy **allows** the actions `ec2:CreateFlowLogs`, `ec2:DescribeFlowLogs`, and `ec2:DeleteFlowLogs` with `Resource` set to a wildcard (any resource), then attach the role to an IAM user or group. This grants permission to create, describe, and delete flow logs.

TABLE 7.5 Default (version 2) flow log record fields

FIELD	DESCRIPTION	EXAMPLE
version	Flow Logs version — the default format is version 2	2
account-id	AWS account that owns the network interface	123456789010
interface-id	Network interface the traffic is for	eni-1235b8ca123456789
srcaddr / dstaddr	Source and destination IP addresses	172.31.16.139 / 172.31.16.21
srcport / dstport	Source and destination ports	20641 / 22
protocol	IANA protocol number	6 (TCP)
packets / bytes	Packets and bytes transferred	20 / 4249
start / end	Unix time of first and last packet	1418530010 / 1418530070
action	ACCEPT or REJECT (blocked by SG/ACL rules or a closed connection)	ACCEPT
log-status	OK (normal), NODATA (no traffic), or SKIPDATA (records skipped)	OK

MORE TROUBLESHOOTING TOOLS

- **Reachability Analyzer** — tests connectivity between a source and destination in a VPC; when reachable it returns the **hop-by-hop path**, and when not it names the **blocking component** (security group, network ACL, route table, or load balancer).
- **Network Access Analyzer** — identifies **unintended network access** to your resources, helping you improve security posture and demonstrate compliance (for example, proving a cardholder-data network is isolated).
- **Traffic Mirroring** — copies actual network traffic, including message payloads, to security and monitoring appliances for deep packet inspection.

COMMON PITFALL

A `REJECT` in a flow log isn't always an attack. Traffic is rejected when a **security group or network ACL rule** blocks it, or simply when **packets arrive after a connection is closed**. And `log-status` matters: **NODATA** means there was no traffic during the interval, while **SKIPDATA** means records were dropped due to a capacity constraint or error — neither means traffic itself was blocked.

7.5 Applying Well-Architected Framework principles to a network

The **AWS Well-Architected Framework** supplies best practices — a set of foundational questions — for designing, operating, and maintaining workloads on AWS. Of its six pillars, four bear directly on networking: **Reliability, Security, Performance Efficiency, and Cost Optimization**. Your VPC should meet the same standards as the workloads it carries: resilient, secure, performant, and cost-effective. The practices below are highlights, not an exhaustive list.

TABLE 7.6 The four network-related Well-Architected pillars

PILLAR	NETWORK BEST PRACTICE	WHAT IT MEANS
Reliability	Plan your network topology; ensure IP subnet allocation accounts for expansion and availability	Anticipate failure and future growth; resiliency is shared — AWS for the cloud, you in the cloud
Security	Create network layers, control traffic at all layers, implement inspection and protection	Apply Zero Trust and defense in depth with security groups, network ACLs, subnets, route tables, and gateways
Performance Efficiency	Understand networking's performance impact, evaluate features, and choose protocols	Match latency, throughput, jitter, and bandwidth to each workload
Cost Optimization	Choose Regions based on cost	Regional prices differ; also weigh latency and data sovereignty

RELIABILITY – IP ALLOCATION BEST PRACTICES

- Leave CIDR space for **multiple subnets across multiple AZs**, and reserve **unused CIDR space** for future expansion.
- Remember each subnet CIDR has **five reserved** addresses, and that some services (such as **container services**) consume extra addresses.
- **Deploy large VPC CIDR blocks** — a CIDR **can't be changed or deleted** after creation, though you can **add non-overlapping** blocks.
- Plan **subnet** CIDR ranges carefully — subnet IPv4 CIDRs **can't be changed** once set.

SECURITY, PERFORMANCE, AND COST HIGHLIGHTS

- **Zero Trust**: treat every component as untrusted; group workloads by **sensitivity** into layers, and let traffic flow only from the adjacent, least-sensitive resource.
- **Performance**: synchronous replication is feasible **between AZs** (single-digit ms) but not **between Regions** (tens of ms); don't default everything to TCP — pair **TCP and UDP** where it helps, for example for virtual desktop infrastructure.
- **Cost**: choose the Region delivering the best overall global cost, and place compute **closer to users** for lower latency and strong data sovereignty.

READING A BAD DESIGN – FOUR ANTI-PATTERNS AND THEIR FIXES

A European shoe company selling mostly to US customers deployed its website **and** database servers in the Ireland Region in a **single VPC with one public subnet**, sized as a tiny `/27` VPC (32 addresses) with a `/28` subnet (16 addresses), and a security group that **allows internet traffic to every server** — all while projecting rapid growth. Four fixes follow the four anti-patterns: replace the **small VPC and subnets** with **large public and private subnets** sized for growth; replace **permissive** security groups with **separate, strict** groups layered by server usage; move databases out of **direct internet access** into a **private** subnet (patched via a NAT gateway); and swap the **far-away Region** for a **US Region** near the customers for lower latency and stronger data sovereignty.

Module summary

- An Amazon VPC is a programmatically defined, logically isolated virtual network in one Region that can span AZs; you size it with a private-IP CIDR block (IPv4 /16 max, /28 min), and it stays isolated until you configure access — don't run production in the default VPC.
- Subnets belong to one AZ, take non-overlapping segments of the VPC range (five addresses reserved each), and their route tables set reachability; the auto-created local route can't be deleted. A subnet is public when its table sends 0.0.0.0/0 to an internet gateway (instances need public IPs) and private otherwise.
- A NAT gateway (or NAT instance) in a public subnet gives private-subnet resources outbound-only internet access for tasks like patching; deploy one per AZ, and use an egress-only internet gateway for IPv6. Elastic IPs are static public addresses you can move between instances.
- Layer your defenses: security groups (stateful, resource-level, allow-only) and network ACLs (stateless, subnet-level, allow and deny), plus AWS Network Firewall (a managed IDS/IPS in a firewall subnet) and bastion hosts for administering private resources. A new security group blocks inbound but allows all outbound; a default network ACL allows everything both ways.

- VPC endpoints connect privately to managed services: interface endpoints (PrivateLink, ENI plus private IP, billed, many services) and gateway endpoints (route-table based, free, Amazon S3 and DynamoDB only); Gateway Load Balancer endpoints route traffic through security appliances.
- VPC Flow Logs capture packet-level metadata at the VPC, subnet, or interface level and deliver to CloudWatch Logs, S3 (queried with Athena), or Kinesis Data Firehose; Reachability Analyzer, Network Access Analyzer, and Traffic Mirroring round out troubleshooting.
- The Well-Architected pillars that shape a network — Reliability, Security, Performance Efficiency, Cost Optimization — translate to generous non-overlapping CIDR planning, Zero Trust layered controls, protocol and latency awareness, and cost-driven Region selection.

Review questions

1. **What exactly makes a subnet public rather than private, and why does a subnet live in only one AZ?**

   Answer: A subnet is public when its route table sends 0.0.0.0/0 to an internet gateway (and its instances have public IPs); a private subnet has no such route and falls back to the main route table if none is attached. A subnet belongs to a single AZ (while the VPC spans AZs), so you place subnets in different AZs to spread resources and survive an AZ failure.

2. **A private database instance can't download patches from the internet. How do you fix it without exposing it?**

   Answer: Put a NAT gateway in a public subnet and add a 0.0.0.0/0 route in the private subnet's route table pointing to the NAT gateway. The NAT allows outbound-initiated traffic but blocks unsolicited inbound connections, so the database stays private.

3. **Contrast security groups and network ACLs on scope, state, and rule types.**

   Answer: Security groups act at the resource level, are stateful (return traffic is auto-allowed), and support allow rules only. Network ACLs act at the subnet level, are stateless (return traffic needs its own rule), and support both allow and deny rules evaluated in numbered order.

4. **When would you choose a gateway VPC endpoint over an interface endpoint, and when can't you?**

   Answer: Use a gateway endpoint for Amazon S3 or DynamoDB when you want free, unlimited-throughput private access from within the Region. You can't use it for any other service, or when you need on-premises or cross-Region access — those require an interface endpoint (PrivateLink), which is billed.

5. **Which VPC feature would you enable to investigate why traffic isn't reaching an instance, and where can the data go?**

   Answer: VPC Flow Logs — capture all, accepted, or rejected traffic at the VPC, subnet, or ENI level and deliver it to CloudWatch Logs, Amazon S3 (queried with Athena), or Kinesis Data Firehose. Reachability Analyzer can additionally identify the blocking component.

6. **Name two IP-allocation best practices from the Reliability pillar and why they matter.**

   Answer: Deploy large VPC CIDR blocks and reserve unused CIDR space, because a VPC or subnet IPv4 CIDR can't be changed after creation (you can only add non-overlapping blocks); and leave room for subnets across multiple AZs while accounting for the five reserved addresses per subnet and services that consume extra addresses.