MODULE 4 · STUDY NOTES

Securing Your Infrastructure

Building a defense-in-depth network on AWS — VPCs and subnets, routing and gateways, security groups versus network ACLs, load balancers, and the services that assess your compute

Migrating a bank into the cloud raises an obvious question: once the servers belong to someone else, how do you keep the network secure? This module answers it from the customer's side of the **shared responsibility model** — operating system, network, and firewall configuration, and the protection of network traffic. It builds a defense-in-depth network around a three-tier web application. A **virtual private cloud (VPC)** carves out a logically isolated, account-dedicated network; **subnets** split it across Availability Zones and separate public, internet-facing resources from private backend ones; and **internet and NAT gateways, route tables, CIDR ranges, Elastic IPs, and network interfaces** wire it together. Two firewalls then filter the traffic — **security groups** at the instance level and **network ACLs** at the subnet level — while **Elastic Load Balancing** spreads and secures incoming requests. The module closes by pulling every layer into a single workflow, listing best practices for protecting the network, and turning to **Amazon Inspector** and **AWS Systems Manager** to keep the compute behind it assessed and patched.

LEARNING OBJECTIVES

- Define the components of a virtual private cloud (VPC)
- Recognize account boundaries
- Describe AWS services that are available to protect your network and resources

4.1 The shared responsibility model and three-tier architecture

Security on AWS is a partnership. Under the **shared responsibility model**, AWS is responsible for security *of* the cloud — the foundation services for **compute, storage, databases, and networking**, and the **global infrastructure** of Regions, Availability Zones, and edge locations. The customer is responsible for security *in* the cloud. This module concentrates on two of the customer's responsibilities: the **operating system, network, and firewall configuration**, and **network traffic protection** (including encryption, integrity, and identity). In the bank scenario, María's task is exactly this — showing that AnyBank's infrastructure can be properly secured in AWS by using a VPC and applying multiple layers of security.

CUSTOMER – SECURITY IN THE CLOUD	AWS – SECURITY OF THE CLOUD
you configure and protect	*AWS operates and protects*
▪ Customer data ▪ Platform, applications, and identity and access management ▪ **Operating system, network, and firewall configuration** ▪ Client-side data encryption, data integrity, and authentication ▪ **Network traffic protection** — encryption, integrity, identity	▪ Foundation services: compute, storage, databases, networking ▪ Global infrastructure: Regions, Availability Zones, edge locations ▪ The physical and virtualization layers beneath your resources

The infrastructure secured in this module supports a **three-tier architecture** — a software pattern that splits an application into three logical tiers: a **presentation layer**, an **application (business logic) layer**, and a **data storage layer**. It is the classic shape of a client-server web application with a frontend, a backend, and a database. Each tier performs a specific task and can be managed independently of the others, a deliberate shift away from the **monolithic** approach that packs frontend, backend, and database into one place.

TABLE 4.1 The three tiers and how they map onto AWS

TIER	WHAT IT DOES	IN THE AWS REFERENCE DESIGN
Presentation	Frontend UI; receives user traffic	Public subnets (one per AZ) with an Application Load Balancer and a NAT gateway
Application (business logic)	Processes requests	Frontend and backend Auto Scaling groups behind a Network Load Balancer, in private subnets across two AZs
Data storage	Stores and serves data	A primary database in one AZ with a read-only database in a second AZ, in private subnets

WORKED EXAMPLE – READING THE REFERENCE ARCHITECTURE

The module's diagram places a VPC across **two Availability Zones**. In the presentation layer, each AZ has a **public subnet**: one holds a security group with a host instance, the other holds a **NAT gateway**. An **Application Load Balancer** manages traffic between the internet and a frontend **Auto Scaling group** in the business logic layer, which scales frontend instances across both AZs. A **Network Load Balancer** then manages traffic between the frontend group and a backend Auto Scaling group. Finally, the data storage layer keeps a **primary database** in one AZ and a **read-only database** in the second, communicating with the backend group. The takeaway: public, internet-facing components stay thin and forward inward, while the servers and data that matter sit in private subnets spread across AZs.

COMMON PITFALL

Configuring the network, writing firewall rules (security groups and network ACLs), and patching the operating system are **the customer's** responsibility — not AWS's. AWS secures the underlying infrastructure, but a misconfigured, wide-open VPC is on you. Read every scenario for which side of the shared responsibility line it sits on.

4.2 Using a VPC

The **Amazon Virtual Private Cloud (Amazon VPC)** service provisions a **logically isolated section of the AWS Cloud** where you launch your resources in a virtual network that you define, using **both IPv4 and IPv6**. You customize the network — for example, a **public subnet** for web servers that reach the public internet, and a **private subnet** for backend systems such as databases or application servers with no public internet access — and you apply **multiple layers of security** (security groups and network ACLs) to control access to the Amazon EC2 instances in the VPC's subnets.

> **WHAT AMAZON VPC LETS YOU CONTROL**
>
> - The **IP address range** — chosen when you create the VPC
> - **Subnets** that divide the VPC across Availability Zones
> - **Route tables** and **network gateways** that direct traffic
> - The **public/private** split — internet-facing versus isolated resources
> - **Multiple layers of security** — security groups and network ACLs

KEY TERMS

Amazon VPC
A logically isolated section of the AWS Cloud, dedicated to your account, where you launch resources in a virtual network you define

Subnet
A range of IP addresses that divides a VPC; belongs to a single Availability Zone and is classified as public or private

A VPC is **logically isolated** from other virtual networks in the AWS Cloud and is **dedicated to your account** — this isolation is a key **account boundary**: resources in one account's VPC are separated from every other account's. A VPC **belongs to a single AWS Region and can span multiple Availability Zones**. After creating a VPC, you divide it into one or more **subnets**. A subnet is a range of IP addresses that divides the VPC; it **belongs to a single Availability Zone** but you can create subnets in different AZs for high availability, and each is generally classified as **public or private**.

> **COMMON PITFALL**
>
> Watch the scope words. A **VPC is Region-scoped and spans Availability Zones** — it is not tied to one AZ. The **subnet** is the AZ-scoped unit. And because a VPC is dedicated to a single account and isolated from others, it forms an account boundary you can reason about when separating workloads.

4.3 Setting up public and private subnets and internet protocols

An **internet gateway** connects a VPC to the internet. It serves two purposes: it provides a **target in your VPC route tables for internet-routable traffic**, and it performs **network address translation (NAT)** for instances that have been assigned public IPv4 addresses. An internet gateway supports **IPv4 and IPv6**, doesn't cause availability risks or bandwidth constraints, and **incurs no additional charge** in your account.

WORKED EXAMPLE – ENABLING INTERNET ACCESS FOR A SUBNET

To let instances in a subnet reach or be reached from the internet, four things must line up: (1) **create an internet gateway and attach it to your VPC**; (2) **add a route** to the subnet's route table that directs internet-bound traffic to the internet gateway; (3) confirm each instance has a **globally unique IP address** — a public IPv4 address, an Elastic IP address, or an IPv6 address; and (4) confirm your **network ACL and security group rules** allow the relevant traffic to flow to and from the instance. Miss any one and connectivity silently fails.

A **NAT gateway** supports instances in a **private subnet** to connect to the internet or other AWS services, while **preventing the internet from initiating a connection** to those instances. At creation you must specify the **public subnet** in which the NAT gateway will reside and an **Elastic IP address** to associate with it; afterward you **update the route table** of one or more private subnets to point internet-bound traffic at the NAT gateway. You could instead run a **NAT instance** in a public subnet, but a NAT gateway is a **managed** service with better availability, higher bandwidth, and less administrative effort — AWS recommends the NAT gateway for common use cases.

PUBLIC SUBNET	PRIVATE SUBNET
external traffic can reach an interface ▪ An EC2 instance has a **public IP address** ▪ The **route table** has a route to an internet gateway ▪ AWS recommends a **load balancer** here, relaying to web servers in private subnets	*isolated from the public internet* ▪ Interfaces are **not reachable** from outside the parent VPC ▪ Outbound only, via a managed **NAT gateway** (e.g., patching); the response is allowed back ▪ Often hosts **database instances**; subnet IP ranges are contiguous and must not overlap

IP addressing starts when you create the VPC and assign it a **Classless Inter-Domain Routing (CIDR) range** — a range of private addresses. You **cannot change** the range of a VPC or subnet after the fact, but you **can add more CIDR ranges** to a VPC. The block can be as large as **/16** (2^16 = 65,536 addresses) or as small as **/28** (2^4 = 16 addresses). A subnet's CIDR block can equal the VPC's block (the VPC then has a single subnet) or be a subset of it (supporting multiple subnets). The CIDR ranges of subnets **should not overlap**, and you cannot have duplicate IP addresses in the same VPC.

TABLE 4.2 Five addresses AWS reserves in every subnet (a /24 leaves 251 usable)

ADDRESS IN 10.0.0.0/24	RESERVED FOR
10.0.0.0	Network address
10.0.0.1	VPC local router (internal communication)
10.0.0.2	Domain Name System (DNS) resolution
10.0.0.3	Reserved for future use
10.0.0.255	Network broadcast address

Instances get addresses in two flavors. Every instance automatically receives a **private IP address**. A **public IP address** — used to access the internet — can be auto-assigned by modifying the subnet's *auto-assign public IP* property, but public IPs are **dynamic**: stop or start the instance and a new one is assigned, and the address is dissociated when you stop the instance. For production, prefer an **Elastic IP address**: a **static** public IPv4 address that is allocated to your **account**, comes from Amazon's pool of IPv4 addresses, and **doesn't change over time**. You can remap an Elastic IP to

another instance to mask a failure, but note you are **charged for an Elastic IP while it is allocated but unassigned** — release it when you no longer need it.

KEY TERMS

Internet gateway (IGW)
Gives route tables a target for internet-bound traffic and performs NAT for instances with public IPv4 addresses; supports IPv4/IPv6 at no additional charge

NAT gateway
A managed service placed in a public subnet (with an Elastic IP) that lets private-subnet instances make outbound connections while blocking connections the internet tries to initiate

CIDR block
The private IP range assigned to a VPC or subnet; largest /16 (65,536), smallest /28 (16). A range can't be changed, but you can add more to a VPC

Route table
A set of routes (destination + target) that directs a subnet's traffic; includes an undeletable local route, and covers unassociated subnets via the VPC's main route table

An **elastic network interface (ENI)** is a virtual network interface you can attach to an instance, then detach and attach to another instance to **redirect network traffic**; its attributes follow it when it's reattached. Each instance has a **default (primary) network interface** that can be assigned a private IPv4 address from the VPC's range, and you **cannot detach the primary** interface. The number of interfaces you can attach varies by instance type. Attaching a second interface is useful for **forensics** — you can wire an instance to a forensic instance and track an exploit.

A **route table** contains a set of rules, called **routes**, that direct traffic from your subnet — each route pairing a **destination** (a CIDR block) with a **target** (what the traffic is sent through), like a lookup map for forwarding traffic. Every route table has a **local route** for communication within the VPC that you **cannot delete**. Each subnet should have its own route table; a subnet not explicitly associated with one uses the VPC's **main route table**, assigned automatically to control routing for all otherwise-unassociated subnets. A subnet associates with **only one route table at a time**, but one route table can serve **many subnets**. The key takeaways of this section: public subnets carry traffic that must reach an interface such as an EC2 instance, private subnets host resources like databases that need no public access, and route tables determine where traffic is routed.

4.4 Using AWS security groups

A **security group** acts as a **virtual firewall for an EC2 instance** and controls its **inbound and outbound** traffic. Security groups operate at the **instance level, not the subnet level**, so each instance in a subnet can be assigned a different set of security groups. At its most basic, a security group is a way to **filter traffic to your instances**.

HOW SECURITY GROUP RULES BEHAVE

- **Instance/ENI level** — each instance in a subnet can have a different set of groups.
- **Allow rules only** — a security group has no deny rule.
- A **new group has no inbound rules** (inbound is blocked until you add one) and one **allow-all outbound** rule (remove it to restrict outbound; with no outbound rules, outbound is blocked).
- **Stateful** — a response to an allowed request is permitted regardless of the rules in the other direction.
- **All rules are evaluated** before a decision is made to allow traffic.

The key word is **stateful**: state information is kept even after a request is processed. If you send a request *from* your instance, the **response traffic is allowed to flow in regardless of inbound rules**; responses to allowed inbound traffic are allowed back out regardless of outbound rules. A common default posture — deny all inbound, allow all outbound — reflects the fact that a fresh group has no inbound rules and an allow-all outbound rule.

WORKED EXAMPLE – READING A SECURITY GROUP'S DEFAULT RULES

A brand-new security group has **no inbound rules**, so inbound traffic isn't permitted until you add one. It includes one **outbound rule** that allows all traffic — `0.0.0.0/0` for IPv4 and `::/0` for IPv6. A frequently used inbound pattern names the security group **itself** as the source (`sg-xxxxxxxx`, all protocols, all ports), which permits traffic from any interface assigned to the same group — a clean way to let instances in one tier talk to each other, and it **scales automatically** as instances join the group. Because the group is stateful, you never add an outbound rule just to let a response leave.

COMMON PITFALL

A security group is **allow-only** and **stateful**. You cannot write a *deny* rule in a security group, so to block a specific IP address you need a **network ACL**. And don't add an outbound rule merely to let responses to allowed inbound traffic leave — statefulness already handles that.

4.5 Using AWS network ACLs

A **network access control list (network ACL)** is an **optional layer of security** for your VPC that acts as a firewall to control traffic **in and out of one or more subnets** — a coarser complement to security groups, with rules that look similar. **Each subnet in your VPC must be associated with a network ACL**; if you don't explicitly associate one, the subnet is automatically associated with the **default** network ACL. A network ACL can be associated with **multiple subnets**, but a subnet can be associated with **only one network ACL at a time** — associating a new one removes the previous association.

A network ACL has **separate inbound and outbound rules**, and each rule can **allow or deny** traffic. Network ACLs are **stateless**, which means responses to inbound traffic are subject to the rules for outbound traffic (and vice versa) — return traffic is not automatically permitted. Every VPC comes with a **modifiable default network ACL** that, by default, **allows all inbound and outbound** IPv4 traffic (and IPv6 if applicable). A **custom** network ACL you create **denies all inbound and outbound traffic until you add rules**. Rules are **evaluated in number order**, and each network ACL ends with a rule numbered $*$ that **denies any packet matching no numbered rule** — a rule you cannot modify or remove.

TABLE 4.3 Comparing security groups and network ACLs

ATTRIBUTE	SECURITY GROUPS	NETWORK ACLS
Scope	Instance or interface level	Subnet level
Supported rules	Allow rules only	Allow and deny rules
State	Stateful — return traffic is automatically allowed, regardless of rules	Stateless — return traffic must be explicitly allowed by rules
Order of rules	All rules are evaluated before a decision is made to allow traffic	Rules are evaluated in number order before a decision is made to allow traffic

VPC SECURITY FEATURES AND FLOW LOGS

- Layer the controls, closest-to-farthest from the instance: **security group → network ACL → subnet routing / route tables**.
- **Subnets** make networks more efficient and separate application tiers — use multiple Availability Zones.
- **Route tables** control where network traffic is directed.
- **VPC Flow Logs** capture information about IP traffic to and from network interfaces — down to the **interface level** — and publish it to **Amazon CloudWatch Logs** or **Amazon S3**; you can create a flow log for a VPC, a subnet, or a single interface.

COMMON PITFALL

Because network ACLs are **stateless**, allowing a request in does *not* automatically allow its response out — you must add the return rule yourself (often on ephemeral ports). And remember the default asymmetry: the *default* network ACL allows all traffic, but a *custom* network ACL denies everything until you add rules.

4.6 Using AWS load balancers

The **Elastic Load Balancing (ELB)** service automatically distributes incoming application traffic across multiple targets — EC2 instances, containers, IP addresses, and Lambda functions. You configure a load balancer with one or more **listeners**, a listener being a process that checks for connection requests. ELB **scales** the load-balancing device as traffic changes, increasing **availability and fault tolerance**; you can add and remove instances without disrupting the flow of requests, in a **single Availability Zone or across multiple AZs**. It also performs **health checks** on registered targets, routing only to healthy ones and resuming a target once it is healthy again. ELB integrates with services such as EC2 Auto Scaling, Amazon ECS, AWS CloudFormation, and AWS Certificate Manager (ACM).

TABLE 4.4 The three ELB load balancer types

LOAD BALANCER	OPERATING LEVEL	IDEAL FOR	DISTINCTIVE
Application (ALB)	Request level (Layer 7)	Advanced HTTP/HTTPS routing; microservices and containers	Routes on request content; always uses the latest SSL/TLS ciphers
Network (NLB)	Connection level (Layer 4)	TCP and UDP; sudden, volatile traffic	Millions of requests/sec at ultra-low latency; one static IP per AZ
Classic (CLB)	Request and connection level	Applications on the legacy EC2-Classic network	Basic load balancing across EC2 instances

AWS recommends an **Application Load Balancer for Layer 7** traffic and a **Network Load Balancer for Layer 4** traffic in a VPC. Beyond distribution, a load balancer protects data. As the **single point of contact** for clients, it spreads traffic across targets in multiple AZs to raise availability. For **encryption in transit** it terminates **HTTPS and TLS** from clients at the load balancer, so each EC2 instance need not perform TLS termination; an ALB can also carry certificates and optionally terminate SSL there. For **encryption at rest**, with **SSE-S3** enabled on the access-log bucket, ELB encrypts each log file (a unique key per file, itself encrypted by a regularly rotated key).

WORKED EXAMPLE – LOAD BALANCERS IN ACTION

Picture a VPC across **two Availability Zones**, each with a public subnet and multiple private subnets. Internet traffic enters through an **internet gateway** and reaches each AZ. A **load balancer** in each public subnet directs traffic to **web servers** in a private subnet in either AZ. Traffic from the web servers passes to another load balancer, which directs it to **application servers** in a different private subnet. The application servers talk to the **primary database** in a private subnet in the first AZ, and that primary can communicate with a **standby database** in a private subnet in the second AZ. Every tier that matters sits in a private subnet; only the load balancers face the internet.

COMMON PITFALL

A load balancer is **not a firewall**. It distributes traffic, health-checks targets, and can terminate TLS — but it does not filter inbound and outbound traffic the way a security group or network ACL does. On scenario questions, 'apply a firewall to an instance' points to a **security group**, never a load balancer.

4.7 Pulling it all together

All of these features cooperate on the path a packet takes. Consider a VPC with two subnets. In **public subnet A**, an EC2 instance's traffic routes through a **load balancer**, then its **security group**, then the subnet's **network ACL**, then the **route table**, and out through the **internet gateway**. In **private subnet B**, an EC2 instance's traffic routes through a load balancer, its security group, and the network ACL, then through the route table to the **NAT gateway** in the public subnet — which reaches the internet on the instance's behalf without exposing it to inbound connections. The same building blocks, arranged into an inner-to-outer stack, give you defense in depth.

BEST PRACTICES TO PROTECT YOUR NETWORK

- **Control traffic at all layers** — apply controls for both inbound and outbound traffic using security groups, network ACLs, and subnets; use subnets in multiple AZs to separate application layers, and allow only the necessary traffic.
- **Inspect and filter** your traffic at the application level.
- **Automate network protection** — use threat intelligence and anomaly detection for a self-defending network.
- **Limit exposure** — allow only the minimum required access to the internet and internal networks.

WORKED EXAMPLE – THE MODULE'S SAMPLE EXAM QUESTION

*A system administrator created a single EC2 instance and set up network ACLs and the appropriate subnet routing. They now want an extra layer of security by applying a **firewall to control access to and from the EC2 instance**. Which action should they take?* The keywords are **applying a firewall** and **control access to and from the EC2 instance** — instance-level filtering. The answer is **configure a security group**, the virtual firewall for an EC2 instance's inbound and outbound traffic. Not a network ACL (it's a firewall at the **subnet** level), not a route table (it directs traffic, it isn't a firewall), and not a load balancer (it distributes traffic and scales resources, it isn't a firewall).

4.8 Protecting your compute resources

Amazon Inspector is an **automated security assessment service** that helps improve the security and compliance of applications deployed on AWS. It runs automated assessments on **EC2 instances and applications** to identify security vulnerabilities and deviations from best practices — both **before deployment and while running in production** — for example, checking for **unintended network accessibility** of your instances. You can define standards and best practices

and validate adherence to them, and Inspector produces a detailed list of **security findings prioritized by severity**, available as reports through the AWS Management Console or the API.

SECURITY BENEFITS OF AMAZON INSPECTOR

- **Automate responses** — pair Inspector findings with **Amazon EventBridge** events to automate reactions to security issues.
- **Regularly monitor resources** — catch vulnerabilities and departures from best practices before deployment and in production.
- **Benefit from AWS security expertise** — a knowledge base of rules mapped to common best practices and vulnerability definitions, kept current by AWS.
- **Integrate into DevOps** — an API-driven service with an optional agent, so assessments become part of your deployment process.

Amazon Inspector uses the widely deployed **AWS Systems Manager Agent (SSM Agent)** to collect the software inventory and configurations from your EC2 instances, then uses that inventory to assess workloads for vulnerabilities. **AWS Systems Manager** more broadly gives you visibility and control of your infrastructure on AWS: a unified interface to view operational data from multiple services and capabilities to automate management tasks. With it you can **collect system inventory, apply operating system patches, maintain antivirus definitions, and configure operating systems and applications at scale**, keeping your systems compliant with defined configuration policies. The takeaway: **scan your compute resources regularly** for vulnerabilities and patch them, and automate that work with services such as **AWS Lambda** and Systems Manager.

COMMON PITFALL

Assessment is not remediation. **Amazon Inspector finds and prioritizes** issues, but you still have to act — patch and reconfigure — and you can automate the fix with **AWS Lambda** and **Systems Manager**. Also remember Inspector depends on the **SSM Agent** to gather each instance's inventory and configuration.

Module summary

- This module is the customer's side of the shared responsibility model — operating system, network, and firewall configuration, plus network traffic protection — securing a three-tier web application, while AWS secures the underlying cloud infrastructure.
- A VPC is a logically isolated, account-dedicated virtual network in a single Region that spans Availability Zones; subnets divide it, each living in one AZ and classified public or private.
- A subnet is public when an instance has a public IP and the route table has a route to an internet gateway; a private subnet reaches the internet outbound through a NAT gateway (placed in a public subnet, with an Elastic IP) that blocks inbound connections.
- CIDR blocks range from /16 to /28 and can't be changed, though ranges can be added; AWS reserves 5 addresses per subnet (251 usable in a /24); Elastic IPs are static, ENIs are movable virtual NICs, and route tables direct subnet traffic.
- A security group is a stateful, instance-level, allow-only firewall (all rules evaluated together); a network ACL is a stateless, subnet-level firewall with allow and deny rules evaluated in number order (ending in a * deny), and every subnet must have exactly one.

- Elastic Load Balancing distributes traffic across healthy targets — Application (Layer 7, HTTP/HTTPS), Network (Layer 4, TCP/UDP), and Classic — and protects data as the single point of contact, terminating TLS in transit and encrypting access logs at rest.
- Defense in depth stacks route tables/subnets, network ACLs, and security groups; control inbound and outbound at every layer, limit exposure to the minimum, and monitor with VPC Flow Logs.
- Protect compute with Amazon Inspector (automated assessments via the SSM Agent, findings ranked by severity) and AWS Systems Manager (inventory, patching, and configuration at scale); scan and patch regularly.

Review questions

1. **Which parts of the shared responsibility model does this module cover, and who owns them?**

   Answer: Operating system, network, and firewall configuration, plus network traffic protection — the customer's responsibility. AWS is responsible for security of the cloud: the foundation services and the global infrastructure (Regions, AZs, edge locations).

2. **How do the scopes of a VPC and a subnet differ?**

   Answer: A VPC is Region-scoped, dedicated to a single account, and spans multiple Availability Zones; a subnet is a slice of the VPC's IP range that belongs to one AZ and is classified public or private.

3. **What makes a subnet public, and how does a private subnet reach the internet?**

   Answer: Public requires two things: an instance with a public IP and a route-table entry to an internet gateway. A private subnet reaches out through a NAT gateway (in a public subnet, with an Elastic IP), which blocks connections the internet tries to initiate.

4. **Compare security groups and network ACLs on scope, statefulness, and rule types.**

   Answer: Security group: instance/interface level, stateful, allow rules only, all rules evaluated before allowing. Network ACL: subnet level, stateless, allow and deny rules evaluated in number order (with a final $*$ deny).

5. **Why does a /24 subnet provide only 251 usable addresses?**

   Answer: AWS reserves 5 addresses in every subnet — the network address, the VPC local router, DNS resolution, future use, and the network broadcast address — so $256 - 5 = 251$ are usable.

6. **When would you choose an Application Load Balancer over a Network Load Balancer?**

   Answer: Choose an ALB for request-level (Layer 7) HTTP/HTTPS traffic with content-based routing. Choose an NLB for connection-level (Layer 4) TCP/UDP traffic needing ultra-low latency, millions of requests per second, and a static IP per AZ.

7. **Which two services protect your compute resources, and how do they relate?**

   Answer: Amazon Inspector runs automated security assessments and ranks findings by severity, using the AWS Systems Manager SSM Agent to collect each instance's inventory and configuration. Systems Manager gives broader visibility and control to patch and configure resources at scale.