

Cloud Certified Practitioner Essentials note

AWS 4-month Plan Career Development CCP

Notes

AWS Certified Cloud Practitioner Essentials

AWS CCP Essentials suammary notes

AWS CCP

Complete study summary (exhaustive, hierarchical, exam-ready)

Module 1 - Introduction

1) Course intro & presenters

- **Course title:** AWS Cloud Practitioner Essentials (Module 1 — Introduction to the Cloud).
 - **Presenters (who they are & why they matter):**
 - **Morgan Willis** — Principal Cloud Technologist, AWS Training & Certification. Former IT support & teacher; focuses on helping learners get cloud-ready.
 - **Rudy Chetty** — Principal Solutions Architect & “Techfluencer” for AWS Partners. Long experience with customers and education focus.
 - **Alan Meridian** — AWS instructor with lots of classroom delivery experience.
 - **Teaching approach:** Simple analogies, demos, layered learning (concepts → examples → demos) aimed at making complex topics digestible.
-

2) High-level definition: What is cloud computing?

- **One-line definition:** On-demand delivery of IT resources over the internet with pay-as-you-go pricing.
- **Key pieces of that definition:**
 - **On-demand:** Provision resources when needed, release when not needed.
 - **Over the internet:** Services are accessed remotely (browser / API).
 - **Pay-as-you-go:** Costs scale with usage — no large upfront capital for hardware.

- **Why this matters:** removes heavy upfront investment, enables rapid experiments and fast scaling, lowers barriers for startups and small teams.
-

3) Short AWS origin timeline (context)

- AWS began as internal tooling to scale Amazon.com's infrastructure needs.
 - **2003–2006 milestone timeline (as in notes):**
 - Internal tools matured, idea to offer them externally.
 - **Nov 2004:** AWS launched its **first public infra service — Amazon SQS** (message queue).
 - **~2006:** AWS launched **Amazon S3** (storage) and **Amazon EC2** (compute).
 - **Result:** AWS expanded rapidly into databases, networking, analytics and now powers millions of customers (startups → enterprises → governments).
-

4) Client-server model — coffee shop analogy (fundamental computing model)

- **Actors:**
 - **Client (customer)** → makes a request (e.g., “an espresso”).
 - **Server (barista)** → validates request (money, ability to make drink) and returns response (coffee).
 - **Cloud analogy points:**
 - Server = virtual server / cloud service that handles requests (not necessarily physical hardware you manage).
 - Real systems are more complex than single client/server transactions — they're multi-component applications.
 - **Purpose of analogy:** helps internalize request/response, validation, and the concept of services responding to client needs.
-

5) Core AWS concept: pay-only-for-what-you-use (staffing analogy)

- **Staffing analogy:** You pay baristas only while they work; you increase staff when demand spikes and reduce when demand falls.
 - **Cloud parallel:** No need to pre-buy and run servers all the time. Provision when needed, deprovision when done → stop paying immediately.
-

6) The six primary business benefits of using AWS (detailed)

1. **Trade fixed costs for variable costs (pay-as-you-go):**
 - Start small; you pay for what you consume. Use billing & budgeting tools to optimize costs.
 2. **Massive economies of scale:**
 - AWS buys hardware at scale → cost savings passed to customers.
 3. **Don't guess capacity (elastic scaling):**
 - Scale up or down in minutes. Avoid over- or under-provisioning hardware.
 4. **Increase speed and agility:**
 - Rapidly spin up test environments, experiment, then remove resources if they fail — faster innovation cycles.
 5. **Stop spending time running data centers:**
 - Offload racking, powering, maintaining physical servers and focus on product/customer work.
 6. **Go global in minutes:**
 - Deploy to AWS Regions around the world to serve global users quickly (no months of building infrastructure).
-

7) AWS Global Infrastructure — components & concepts

- **Purpose:** global reach + built-in redundancy for availability and latency needs.
 - **Hierarchy:**
 - **Regions:** independent geographic areas (examples: Dublin (eu-west-1), Tokyo, Sao Paulo, Ohio).
 - **Availability Zones (AZs):** 3+ per Region, physically separate data centers within a Region (redundant networking/power).
 - **Data centers:** discrete buildings within an AZ with redundant power, cooling, and connectivity.
 - **Design goals:**
 - **High availability:** keep applications accessible with minimal downtime by using multiple AZs.
 - **Fault tolerance:** design systems to keep operating even if multiple components fail (use multi-AZ and multi-Region patterns).
 - **Multi-Region strategy:** if a Region fails, services can failover to another Region — used for disaster recovery and global resilience.
-

8) AWS Shared Responsibility Model (SRM)

- **Core idea:** security responsibility is split: **AWS is responsible for security of the cloud; you are responsible for security in the cloud.**
 - **AWS responsibilities (Security OF the cloud):**
 - Physical infrastructure (data centers, racks).
 - Network fabric and its security.
 - Hypervisor and virtualization layers.
 - **Customer responsibilities (Security IN the cloud):**
 - Operating systems, patches, and OS hardening (if using IaaS).
 - Applications and application configuration.
 - Data ownership, access controls, encryption keys (you control who accesses data and how it's encrypted).
 - Identity and access management (IAM policies, user accounts) and any configuration that runs in your account.
 - **Analogy:** builder (AWS) constructs a secure house; homeowner (you) locks the doors and manages interior security.
 - **Important note:** responsibility boundary shifts depending on the service model (IaaS vs PaaS vs SaaS). Always check the service-specific responsibility details.
-

9) Cloud in real life — e-commerce example (how concepts combine)

- **Business goal:** expand globally and serve customers with low latency + high availability.
 - **How AWS helps:**
 - Deploy applications in Regions close to customer bases (e.g., Ireland for Europe, Singapore for Asia) to reduce latency.
 - Use multiple AZs within a Region to design for **high availability** and failover.
 - Optionally use multiple Regions for disaster recovery and global failover (customer-facing continuity).
 - **Security & SRM in practice:**
 - AWS secures infrastructure; the e-commerce company secures application code, payment data, compliance requirements (e.g., PCI/DSS).
 - The company must implement encryption, IAM, and secure configuration to meet regulatory obligations.
 - **Business advantage:** small teams can reach global markets quickly with lower capital outlay and built-in resiliency.
-

10) Key terms & exam focus (what to memorize / understand)

- **Definition:** cloud computing = *on-demand delivery of IT resources over the internet with pay-as-you-go pricing*.
 - **Client-server model** (request → validate → response) and how it maps to cloud services.
 - **Pay-as-you-go principle + staffing analogy.**
 - **Six benefits of the cloud** (list them and be able to explain each).
 - **AWS origin timeline:** internal tooling → SQS (2004) → S3 & EC2 (~2006) → growth into many services.
 - **Global infrastructure hierarchy:** Region → Availability Zone → Data center.
 - **High availability vs fault tolerance** (definitions & why use multiple AZs/Regions).
 - **Shared Responsibility Model:** AWS = security *of* the cloud; customer = security *in* the cloud. Be ready to map examples to each side (physical, network, hypervisor vs OS, apps, data, IAM).
 - **Real-world application:** e-commerce example showing Regions/AZs + SRM in practice.
-

11) Common analogies to recall (useful in exam answers & interviews)

- **Coffee shop** — client (customer) and server (barista) as a request/response model.
 - **Staffing** — pay-as-you-go; increase/decrease staff = scale resources.
 - **Chain of shops** — availability and resiliency using multiple locations (AZs/Regions).
 - **House builder/homeowner** — AWS builds the house (cloud) vs you secure the interior (your data and apps).
-

12) Quick study checklist (practical actions to review this module)

- Memorize the **cloud computing definition** and **six benefits**.
 - Draw the **global infra hierarchy:** Region → AZ → Data center (sketch it).
 - Be able to **explain SRM** with three concrete examples of AWS responsibilities and three concrete examples of customer responsibilities.
 - Rehearse the **coffee shop** and **staffing** analogies in one or two sentences each.
 - Remember **SQS (2004)** and **S3/EC2 (~2006)** as the start of public AWS services — know why that origin matters (internal tools → public offering).
 - Practice a short CV/answer describing how an e-commerce company uses Regions, AZs, and SRM for global expansion and compliance.
-

13) Final practical notes (PM angle / how to use this knowledge)

- As a PM preparing for CCP, you must be able to **translate business needs into cloud capabilities**: cost model, speed to market, scaling, and availability.
 - Use the analogies to explain cloud trade-offs to stakeholders (execs care about cost, availability, speed).
 - When planning a migration or global rollout, map customer geography → target Region(s) → AZ usage → SRM tasks (who patches, who encrypts, who owns DR runbooks).
-

Definition (core)

- **Cloud computing:** On-demand delivery of IT resources **over the internet** with **pay-as-you-go** pricing.
 - **AWS:** Global cloud provider offering compute, storage, DB, networking, AI, etc., built from Amazon's internal tooling.
-

Core concepts (short)

- **Pay-as-you-go:** Consume → pay; deprovision → stop paying.
 - **On-demand / elastic:** Provision in minutes; scale up/down automatically.
 - **Global infra hierarchy: Region → Availability Zone (AZ) → Data center.**
 - Region = geographic area (multiple AZs).
 - AZ = isolated location within Region (redundant power/network).
 - **High availability (HA):** Design so service stays up with minimal downtime (use multi-AZ).
 - **Fault tolerance:** Continue operating despite multiple component failures (design redundancy in depth).
 - **Client-server model:** Client makes request → server validates → server responds (coffee shop analogy).
-

6 Business benefits (memorize & be able to explain)

1. **Pay only for what you use** (variable vs fixed costs).
 2. **Economies of scale** (AWS buys hardware at volume; cost advantage).
 3. **No guessing capacity** (elastic scaling; avoid over/under-provisioning).
 4. **Speed & agility** (spin up test/dev instantly; iterate fast).
 5. **Reduce ops burden** (less time managing data centers).
 6. **Global reach in minutes** (deploy to Regions worldwide).
-

Shared Responsibility Model (simple)

- **AWS = security *OF* the cloud:** Physical data centers, network, hypervisor, hardware.

- **Customer = security IN the cloud:** OS patches, app config, IAM, data protection, encryption keys.
 - **Note:** Responsibility boundary shifts by service type (IaaS vs PaaS vs SaaS).
-

Quick exam-ready facts

- **First public AWS service:** SQS (early AWS history context).
 - **Examples of services to name:** EC2 (compute), S3 (object storage), RDS (managed DB), SQS (messaging).
 - **Design pattern:** Use multi-AZ for HA; multi-Region for DR and global failover.
 - **Billing tools:** Cost Explorer, budgets, tagging for chargeback (know these exist).
-

Study & exam tips (practical)

- Use analogies (coffee shop, staffing, builder/homeowner) to explain concepts concisely.
- Be able to map **who owns** specific security tasks (patch OS vs secure hypervisor).
- Practice: draw Region → AZ → datacenter and explain failover flow.
- Timebox study: CCP achievable in ~2–4 weeks with daily focused study + hands-on labs.

Module 2 - Compute in the Cloud (Amazon EC2)

Here is a comprehensive summary of Module 2 based on your notes, structured for AWS CCP exam preparation.

1. Introduction to Amazon EC2

Amazon Elastic Compute Cloud (EC2) provides secure, resizable compute capacity in the cloud. It allows you to provision virtual servers (Instances) in place of physical on-premises servers.

- **Client/Server Model:** EC2 acts as the **Server** (delivering resources/data) in response to requests from a **Client** (user/computer).
- **Benefits:**
 - **Flexible:** Launch resources only when needed; stop/terminate when finished.
 - **Cost-Effective:** Pay only for running instances (not stopped/terminated ones).
 - **Speed:** Provision resources in minutes vs. weeks for on-prem hardware.
- **Virtualization & Multi-tenancy:**
 - **Virtual Machines (VMs):** EC2 instances are VMs.
 - **Multi-tenancy:** Multiple VMs share the same underlying physical host hardware.
 - **Hypervisor:** The software layer on the host that shares resources and isolates VMs from one another for security.

- **Configuration:** You have full control over the Operating System (Windows/Linux), software stack, and networking (public/private access).
 - **Vertical Scaling:** You can resize instances (add more RAM/CPU) if workloads increase.
-

2. Amazon EC2 Instance Families

AWS groups instances based on hardware resources (CPU, Memory, Storage, Networking). Choosing the right type optimizes performance and cost.

Family	Optimized For	Use Cases
General Purpose	Balance of compute, memory, and networking.	Web servers, code repositories, diverse workloads. (Good starting point).
Compute Optimized	High-performance processors. CPU	Batch processing, media transcoding, high-performance web servers, scientific modeling, gaming servers.
Memory Optimized	Fast performance for large datasets in memory. RAM	High-performance databases, real-time analytics, caching.
Accelerated Computing	Hardware accelerators (GPUs , FPGAs).	Graphics processing, floating-point number calculations, data pattern matching.
Storage Optimized	High read/write access to large datasets on local storage HARD	NoSQL databases, data warehousing, distributed file systems.

3. Interacting with AWS (Provisioning)

All interactions with AWS are API calls. There are three ways to invoke these APIs:

1. AWS Management Console:

- Browser-based graphical interface.
- Best for: Learning, testing, viewing bills, and non-technical management.
- **Cons:** Manual, prone to human error, hard to repeat.

2. Command Line Interface (CLI):

- Text-based commands via terminal (e.g., `aws ec2 run-instances`).
- Best for: Automation, scripting, and repetitive tasks.
- **AWS CloudShell:** A browser-based terminal with CLI pre-installed.

3. Software Development Kit (SDK):

- Interact with resources using programming languages (Python, Java, etc.).
- Best for: Integrating AWS logic directly into application code.

4. Configuring and Launching an EC2 Instance

To launch an instance, you must configure the following components:

- **1. AMI (Amazon Machine Image):** A template containing the software configuration (OS, App Server, Applications).
 - **Quick Start AMIs:** Pre-configured by AWS (e.g., Amazon Linux, Windows).
 - **Custom AMIs:** Created by you for consistency and faster boot times.
 - **Marketplace AMIs:** Sold by third-party vendors with specific software installed.
 - **2. Instance Type:** Determines the hardware power (e.g., `t2.micro` - 1 vCPU, 1 GB RAM).
 - **3. Key Pair:** A security credential (private key) used to securely connect to the instance (SSH for Linux, RDP for Windows).
 - **4. Network Settings:** Security Groups (firewalls) to allow traffic (e.g., allow HTTP port 80).
 - **5. Storage:** EBS volumes (virtual hard drives).
 - **6. User Data:** A script entered during launch that runs **only once** when the instance first boots. Used for **Bootstrapping** (installing updates/software automatically upon launch).
-

5. Amazon EC2 Pricing Models

There are different payment options to optimize costs based on usage patterns.

Model	Description	Best For	Discount
On-Demand	Pay by the second/hour. No commitment.	Spiky, short-term, or unpredictable workloads. Testing/Dev.	None (Base Price)
Savings Plans	Commit to a specific usage amount (\$/hr) for 1 or 3 years.	Flexible workloads. Applies across instance families, regions, and OS. Also applies to Fargate/Lambda.	Up to 72%
Reserved Instances (RI)	Commit to specific instance attributes for 1 or 3 years.	Steady-state, predictable usage (e.g., database always running). Less flexible than Savings Plans.	Up to 75%
Spot Instances	Bid on unused AWS capacity. Can be interrupted with 2-minute warning.	Fault-tolerant, flexible, stateless workloads (e.g., batch processing).	Up to 90%
Dedicated Hosts	Physical server dedicated to your use.	Compliance requirements or software licensing that requires socket/core visibility.	Varies

*Note: **Dedicated Instances** run on hardware dedicated to you, but you do not control instance placement on the specific physical server like you do with Dedicated Hosts.*

6. Scaling and Elasticity

- **Scalability:** The ability to handle increased load by adding resources.
 - **Vertical Scaling (Scaling Up):** Increasing the power of an individual instance (more RAM/CPU). Limitation: You eventually hit a hardware ceiling.
 - **Horizontal Scaling (Scaling Out):** Adding more instances to the pool. Preferred for cloud architectures.
- **Elasticity:** The ability to automatically acquire resources when demand increases and release them when demand decreases (scaling in/out).
- **Amazon EC2 Auto Scaling:**
 - Ensures you have the correct number of instances to handle current load.
 - **Dynamic Scaling:** Responds to real-time metrics (e.g., CPU utilization spikes).
 - **Predictive Scaling:** Schedules capacity based on historical usage patterns.
 - **High Availability:** Deploys instances across multiple Availability Zones (AZs) to prevent a single point of failure(SPOF).

7. Elastic Load Balancing (ELB)

The ELB acts as the "host" of your application, directing incoming traffic to the correct downstream resources (EC2 instances).

- **Function:** Distributes incoming application traffic across multiple targets (instances) in one or more AZs.
- **Health Checks:** ELB monitors instances. If an instance fails a health check (Unhealthy Threshold), ELB stops sending traffic to it.
- **Decoupling:** A frontend web server doesn't need to know which backend servers exist; it just points to the ELB.
- **Routing Methods:** Round Robin, Least Connections, IP Hash, Least Response Time.

8. Messaging and Queuing (Decoupling Architectures)

In a cloud architecture, components should be **Loosely Coupled** to prevent cascading failures.

- **Tightly Coupled:** If Component A talks directly to Component B, and B fails, A also fails.
- **Loosely Coupled:** A buffer (queue) is placed between components. If B fails, A can still do its work, and messages are stored until B recovers.

Comparison of Messaging Services

Feature	Amazon SQS (Simple Queue Service)	Amazon SNS (Simple Notification Service)	Amazon EventBridge
Concept	A message buffer/queue.	A notification topic (Pub/Sub).	A serverless event bus.

Mechanism	Pull-based: Consumers poll the queue to retrieve messages.	Push-based: Pushes messages immediately to subscribers.	Event-based: Uses rules to route events to targets.
Persistence	Durable (stores messages up to 14 days).	Not durable (fire and forget).	Not durable (real-time routing).
Use Case	Decoupling applications, batch processing, handling traffic bursts.	Sending emails, SMS, mobile push, or triggering Lambda functions.	Building event-driven architectures, routing between SaaS apps and AWS.

Module 2: Cram Cheat Sheet

Compute (EC2)

- **EC2 (Elastic Compute Cloud):** Resizable, virtual servers. Acts as the "Server" in Client/Server model.
- **Multi-tenancy:** VMs share physical hardware but are isolated by the **Hypervisor**.
- **Vertical Scaling:** Changing instance size (e.g., adding more RAM/CPU to an existing server).
- **Horizontal Scaling:** Adding *more* instances (e.g., going from 1 server to 3).
- **User Data:** Script entered during launch to install software/updates. Runs **only once** on first boot.
- **AMI (Amazon Machine Image):** Template for the instance (OS, App Server, Applications).

Instance Types

- **General Purpose:** Balanced resources. Good for web servers/code repos.
- **Compute Optimized:** High performance media transcoding, gaming, scientific modeling, machine learning.
- **Memory Optimized:** Large datasets in memory. Good for databases, real-time analytics.
- **Accelerated Computing:** Hardware accelerators (GPU/FPGA). Good for graphics, floating-point math.
- **Storage Optimized:** High I/O for local storage. Good for Data Warehousing, NoSQL.

Pricing Models

- **On-Demand:** No commitment, pay by second/hour. Best for short-term/spiky workloads.
- **Savings Plans:** Commit to **\$ usage/hr** (1 or 3 yrs). Up to 72% off. Flexible (applies to EC2, Fargate, Lambda).
- **Reserved Instances (RI):** Commit to **specific instance attributes** (1 or 3 yrs). Up to 75% off. Steady-state workloads.
- **Spot Instances:** Spare capacity. Up to 90% off. Can be **interrupted** (2-min warning).
- **Dedicated Hosts:** Physical server dedicated to you. Best for **licensing** (BYOL) and compliance.

Architecture & Networking

- **ELB (Elastic Load Balancing):** Distributes traffic to healthy instances. Regional construct.
 - **Auto Scaling:** Adds/removes instances based on demand (Dynamic) or schedule (Predictive).
 - **Tightly Coupled:** Components talk directly. Failure in one causes failure in the other.
 - **Loosely Coupled:** Uses a queue/buffer. Isolates failures.
 - **SQS (Simple Queue Service):** Message Queue. Pull-based. Durable (stores messages). Decouples systems.
 - **SNS (Simple Notification Service):** Pub/Sub. Push-based. Sends to endpoints (Email, SMS, Mobile).
 - **EventBridge:** Serverless Event Bus. Routes events using rules.
-

Module 2: Exam-Style Practice Questions

1. A company needs to run a background data processing job that can withstand interruptions. The job is not time-critical and the company wants to minimize costs. Which EC2 pricing option should they choose?

- A) On-Demand
- B) Reserved Instances
- C) Spot Instances
- D) Dedicated Hosts

Correct Answer: C

Explanation: The text states Spot Instances offer up to 90% off for spare capacity but "AWS can reclaim the instance at any time." This fits workloads that can "tolerate being interrupted."

2. A developer wants to ensure that a script runs automatically to install a web server as soon as a new EC2 instance is launched. Where should this script be placed during configuration?

- A) In the AMI settings
- B) In the User Data section
- C) In the Security Group settings
- D) In the Elastic Load Balancer configurations

Correct Answer: B

Explanation: The text notes that "User Data allows us to paste in a script... which will go ahead and install and activate" software upon launch.

3. Which AWS service allows for "loose coupling" by acting as a buffer that stores messages between applications until they can be processed?

- A) Amazon SNS
- B) Elastic Load Balancing
- C) Amazon SQS
- D) Amazon EventBridge

Correct Answer: C

Explanation: The text defines SQS as a "message queue" where messages are placed "until they are processed," preventing cascading failures (loose coupling). SNS is described as "Push" and not holding messages for pickup.

4. A company is running a database that requires processing large datasets specifically in memory to deliver fast performance. Which EC2 instance family is best suited for this workload?

- A) Compute Optimized
- B) Storage Optimized
- C) Memory Optimized
- D) General Purpose

Correct Answer: C

Explanation: The text explicitly states Memory-optimized instances are "Best for real-time analytics" and workloads that "process large data sets in memory."

5. You are designing an architecture where a failure in one component should not cause the entire system to fail. What architectural principle are you applying?

- A) Vertical Scaling
- B) Tightly Coupled Architecture
- C) Multi-tenancy
- D) Loosely Coupled Architecture

Correct Answer: D

Explanation: The notes describe "Tightly Coupled" as an architecture where a single failure causes cascading issues. The solution to avoid this is a "Loosely Coupled" architecture (using queues like SQS).

Module 3 - Exploring Compute Services

Here is the comprehensive summary, cheat sheet, and practice questions for **Module 3: Exploring Compute Services**, based strictly on the provided text.

1. Management Models: Unmanaged vs. Managed vs. Serverless

AWS services offer varying degrees of control and convenience (The "Coffee Shop" Metaphor).

- **Unmanaged Services (e.g., EC2):**
 - **Analogy:** A high-end espresso machine where you grind beans and pull the shot yourself.
 - **Responsibility:** You manage scaling, patching, OS, and security *in the cloud*.
 - **Benefit:** Maximum control and customization.
- **Managed Services (e.g., ELB, RDS):**
 - **Analogy:** A coffee pod machine. Press a button, get coffee.
 - **Responsibility:** AWS manages underlying infrastructure. You configure settings to meet requirements.
 - **Benefit:** Convenience; reduces operational overhead.
- **Serverless Computing:**
 - You cannot see or access the underlying infrastructure.

- AWS handles provisioning, scaling, high availability, and maintenance automatically.
 - **Benefit:** You focus entirely on the application code.
-

2. AWS Lambda (Serverless Compute)

Lambda is a **Function as a Service (FaaS)** that runs code in response to events without provisioning servers.

- **How it works:**
 - You upload code (Function).
 - You configure a **Trigger** (e.g., an image upload, an SQS message, or an HTTP request).
 - Code runs only when triggered.
 - **Key Features:**
 - **Pricing:** Pay only for compute time used (down to the millisecond) and number of requests.
 - **Scaling:** Automatically scales up or down based on demand (from 1 request to thousands).
 - **Maintenance:** AWS handles patching and security of the underlying environment.
 - **Constraints:** Maximum execution duration is **15 minutes**.
 - **Languages:** Supports Java, Python, Node.js, etc., via **Runtimes**. You can also build Custom Runtimes.
 - **AWS Lambda Power Tuning:** An open-source tool using AWS Step Functions to optimize Lambda configuration for cost vs. performance.
-

3. Containers and Orchestration

Containers solve the "It works on my machine" problem by packaging code, runtime, and dependencies into a single portable unit.

- **Benefits:** Consistent environment, fast start times, and improved resource efficiency.
- **Container Orchestration:**
 - Manages the lifecycle of containers (start, stop, scale, recover from failure).
 - **Amazon ECS (Elastic Container Service):** AWS-native, streamlined tool. Simple integration with other AWS services.
 - **Amazon EKS (Elastic Kubernetes Service):** Managed **Kubernetes**. Offers high flexibility and control. Great for open-source or hybrid standards.
- **Container Registry:**
 - **Amazon ECR (Elastic Container Registry):** Fully managed registry to store, manage, and deploy container images (OCI standard).

Container Compute Options (Launch Types)

Once you choose an orchestrator (ECS or EKS), you must choose where the containers run:

1. **EC2 Launch Type:** You manage the VMs (instances) that host the containers. Full control over hardware/networking.
2. **AWS Fargate (Serverless):** AWS manages the underlying server infrastructure. You focus only on the container.
 - **Scaling:** Scales by launching/stopping containers based on CPU/memory metrics.
 - **Billing:** Based on vCPU and Memory allocated to the container.

Comparison: Lambda vs. Fargate

Feature	AWS Lambda	AWS Fargate
Type	Function-as-a-Service (FaaS)	Container-as-a-Service (CaaS)
Workload	Short-lived, event-driven.	Long-running processes (APIs, Web Servers).
Duration Limit	15 Minutes	No limit (24/7).
Scaling	Instant execution per event.	Scales based on metrics (CPU/RAM).

4. Additional Purpose-Built Compute Services

AWS Elastic Beanstalk

- **Function:** fully managed service for deploying web applications.
- **How it works:** You upload code; Beanstalk handles deployment, capacity provisioning, load balancing, and auto-scaling.
- **Control:** You retain visibility and control over the underlying AWS resources (e.g., EC2 instances).
- **Use Case:** Web apps, RESTful APIs, microservices.

AWS Batch

- **Function:** Run batch computing workloads (massive datasets, simulations).
- **How it works:** Dynamically provisions resources (EC2) based on the volume and requirements of submitted jobs.
- **Use Case:** Financial risk analysis, media transcoding, genomics research.

Amazon Lightsail

- **Function:** Virtual Private Server (VPS) made simple.
- **Key Feature:** Predictable monthly pricing.
- **Use Case:** Simple workloads, blogs (WordPress), small business sites, dev/test environments.

AWS Outposts

- **Function:** Hybrid cloud solution.
- **How it works:** Extends AWS infrastructure (hardware) to your **on-premises** data center.

- **Use Case:** Low-latency requirements, data residency compliance, local data processing.
-

Module 3: Cram Cheat Sheet

Serverless & Event-Driven

- **Serverless:** No infrastructure visibility. AWS handles provisioning/scaling.
- **AWS Lambda:** Event-driven compute. **15-minute** max timeout. Pay per millisecond.
- **Trigger:** The event that causes a Lambda function to execute (e.g., SQS message).
- **Runtime:** The language-specific environment for Lambda (Python, Node.js, etc.).

Containers

- **Container:** Package of code + dependencies. Solves portability issues.
- **Amazon ECR: Registry.** Stores container images.
- **Amazon ECS: Orchestrator.** AWS-native, simple, streamlined.
- **Amazon EKS: Orchestrator.** Managed **Kubernetes**. Complex, open-source standard.
- **AWS Fargate: Serverless Compute engine** for containers. Works with ECS and EKS.
Removes server management.

Other Compute

- **Elastic Beanstalk:** Platform for web apps. You upload code → AWS provisions infra (EC2/ELB). Retain control.
 - **AWS Batch:** Plans and schedules batch jobs (simulations, analysis).
 - **Amazon Lightsail:** Easy VPS. **Predictable monthly price.** Beginners/Simple apps.
 - **AWS Outposts:** Hybrid. Runs AWS infrastructure **on-premises**.
-

Module 3: Exam-Style Practice Questions

1. A developer needs to deploy a piece of code that will run strictly in response to an image being uploaded to a database. The code execution will take less than 2 minutes. Which service is the most cost-effective and requires the LEAST infrastructure management?

- A) Amazon EC2
- B) AWS Lambda
- C) Amazon Lightsail
- D) AWS Batch

Correct Answer: B

Explanation: Lambda is "event-driven," runs code without provisioning servers, and charges only for the compute time consumed. The duration fits within the 15-minute limit.

2. A company wants to deploy a containerized application using Kubernetes but does not want to manage the underlying control plane or clusters manually. Which AWS service should they use?

- A) Amazon ECS
- B) Amazon ECR
- C) Amazon EKS
- D) AWS Elastic Beanstalk

Correct Answer: C

Explanation: The text defines Amazon EKS (Elastic Kubernetes Service) as the service that makes it convenient to run Kubernetes clusters on AWS.

3. You have a requirement to process massive datasets and run scientific simulations. You need a service that automatically schedules and manages compute resources for these specific jobs. Which service is best suited?

- A) AWS Batch
- B) Amazon Lightsail
- C) AWS Outposts
- D) AWS Fargate

Correct Answer: A

Explanation: AWS Batch is explicitly described as a service designed for "heavy-duty tasks like processing massive datasets, running simulations, or performing complex calculations."

4. A startup wants to launch a simple WordPress blog. They want a low, predictable monthly price and an interface that is easier to use than the standard AWS console. Which service should they choose?

- A) AWS Lambda
- B) Amazon EC2
- C) Amazon Lightsail
- D) AWS Elastic Beanstalk

Correct Answer: C

Explanation: Amazon Lightsail is ideal for "basic workloads" like blogs and offers a "predictable monthly price" with a simplified experience.

5. Which compute option allows you to run containers on Amazon ECS without having to provision or manage the underlying EC2 instances?

- A) AWS Outposts
- B) AWS Fargate
- C) Amazon ECR
- D) Amazon EKS

Correct Answer: B

Explanation: AWS Fargate is a "serverless compute engine for containers." When using Fargate, you "do not need to provision or manage servers."

Module 4 - Going Global

Here is the comprehensive summary, cheat sheet, and practice questions for **Module 4: Going Global**, based strictly on the provided text.

1. Introduction to Global Expansion

Expanding a business globally requires strategic decisions regarding location, speed, and consistency.

- **Regions:** Similar to choosing where to open a physical shop, you must decide where to deploy resources based on demand and regulations.

- **Edge Locations:** Similar to small "coffee carts," these provide fast, localized delivery of frequently accessed content (caching) to users without going back to the central server.
- **Consistency:** To ensure the same experience everywhere (standardization), AWS uses **Infrastructure as Code (IaC)** to automate and replicate setups.

[Image of AWS Global Infrastructure map]

2. Choosing AWS Regions

A Region is a physical location in the world where AWS has multiple Availability Zones. When choosing a Region, there are **four key considerations**, usually evaluated in this order:

1. Compliance (Priority #1):

- Regions are isolated; data does not leave a Region unless explicitly moved.
- You must adhere to local data governance laws (e.g., financial data in Frankfurt must stay in Germany).
- If you have a legal requirement to keep data within specific borders, you **must** choose that Region.

2. Proximity:

- Distance impacts **latency** (the time it takes for data to travel).
- Choose a Region close to your customer base to reduce delay.

3. Feature Availability:

- Not all AWS Regions have every feature or service immediately. New innovations are rolled out over time.
- *Example:* AWS GovCloud Regions are specific to US government compliance needs.

4. Pricing:

- Costs vary by Region due to local taxes and energy costs.
 - Some Regions are more cost-effective to operate in than others.
-

3. High Availability and Redundancy

To ensure stability and zero downtime, you should build **Redundant Architectures**.

- **Multi-AZ (Availability Zone) Deployment:**

- If one AZ fails, the application automatically fails over to a backup AZ.
- Benefits: Disaster recovery, business continuity, and transparency to the user.

- **Multi-Region Deployment:**

- Going a step further by deploying in entirely different Regions.
 - If a whole Region has an interruption, you can failover to another Region.
-

4. Edge Services (Speed and Content Delivery)

For users who need data delivered faster than a standard Region can provide, AWS uses the **Amazon Global Edge Network**.

- **Edge Locations:** Physical sites separate from Regions designed to cache content and accelerate delivery.
 - **Amazon CloudFront:**
 - A **Content Delivery Network (CDN)**.
 - Delivers data (videos, images, APIs) to users from the nearest Edge Location.
 - Reduces latency by serving cached content locally rather than retrieving it from the origin.
 - **Amazon Route 53:**
 - A **DNS (Domain Name System)** service hosted at edge locations.
 - Translates human-readable URLs (e.g., www.example.com) into machine-readable IP addresses.
 - **AWS Outposts:**
 - Used when you need speed *faster* than a Region or CloudFront.
 - Extends AWS infrastructure to your **on-premises** facility (hybrid cloud).
-

5. Infrastructure and Automation (IaC)

Managing resources manually (clicking through the console) across multiple Regions is slow, error-prone, and hard to replicate.

- **Infrastructure as Code (IaC):** Defines infrastructure in a file (blueprint) to automate provisioning.
 - **AWS CloudFormation:**
 - An IaC service that uses text-based **Templates**.
 - **Declarative:** You define *what* you want (resources), and CloudFormation handles the *how* (API calls).
 - **Benefits:**
 - **Consistency:** Creates identical environments across different accounts or Regions.
 - **Speed:** Deploys complex stacks with a single command.
 - **Safety:** Reduces human error and allows infrastructure changes to be tracked via source control.
-

Module 4: Cram Cheat Sheet

Global Infrastructure

- **Region:** A physical location around the world containing multiple Availability Zones. Data is isolated here for **Compliance**.

- **Availability Zone (AZ):** One or more discrete data centers. Used for **High Availability** and failover *within* a Region.
- **Edge Location:** Site used to **Cache** content. separate from Regions.

Selection Factors (in order)

1. **Compliance:** Legal/Data sovereignty (e.g., data cannot leave the country).
2. **Proximity:** Distance to users (Latency).
3. **Feature Availability:** Does the region have the service you need?
4. **Pricing:** Tax rates and operational costs vary by region.

Services

- **Amazon CloudFront:** CDN. Uses **Edge Locations** to serve cached content (images/video) quickly.
 - **Amazon Route 53:** DNS service. Translates Names to IP addresses.
 - **AWS Outposts:** Runs AWS infrastructure **On-Premises**. Hybrid.
 - **AWS CloudFormation:** Infrastructure as Code (IaC). Uses **Templates** (blueprints) to provision resources. Ensures consistency and repeatability.
-

Module 4: Exam-Style Practice Questions

1. A company requires that all their financial data remain physically located within the borders of Germany due to local regulations. Which factor should be the primary consideration when choosing an AWS Region?

- A) Proximity
- B) Compliance
- C) Pricing
- D) Feature Availability

Correct Answer: B

Explanation: The text states that before other factors, you must look at compliance requirements. If laws dictate data must stay in a specific area, that is the deciding factor.

2. Which AWS service allows you to define your infrastructure as a text-based template, enabling you to deploy consistent and identical environments across multiple AWS Regions?

- A) Amazon CloudFront
- B) Amazon Route 53
- C) AWS Outposts
- D) AWS CloudFormation

Correct Answer: D

Explanation: AWS CloudFormation is the Infrastructure as Code (IaC) service that uses text-based documents called "templates" to provision resources consistently.

3. Users in Europe are experiencing slow load times for high-resolution images hosted on an application in the US. Which service should be used to cache these images closer to the users to reduce latency?

- A) AWS CloudFormation
- B) Amazon Route 53
- C) Amazon CloudFront
- D) AWS Outposts

Correct Answer: C

Explanation: CloudFront is a Content Delivery Network (CDN) that uses Edge locations to cache frequently accessed content (like images) close to users to reduce latency.

4. A company needs to run AWS infrastructure within their own on-premises data center to achieve lower latency than is possible with a standard AWS Region. Which service should they use?

- A) AWS Outposts
- B) Amazon CloudFront
- C) Availability Zones
- D) AWS GovCloud

Correct Answer: A

Explanation: The text describes AWS Outposts as the service that "essentially makes it possible for you to run AWS services on-premises."

5. What is the primary benefit of deploying an application across multiple Availability Zones (Multi-AZ)?

- A) Lower pricing due to tax incentives.
- B) Reducing the need for Infrastructure as Code.
- C) Automatic failover and high availability if one AZ experiences an interruption.
- D) Caching content closer to users.

Correct Answer: C

Explanation: A multi-AZ architecture allows an application to "automatically switch over to the backup AZ" if an interruption occurs, ensuring redundancy.

Module 5 - Networking

Here is the comprehensive summary, cheat sheet, and practice questions for **Module 5: Networking**, based strictly on the provided text.

Module 5: Networking

1. Amazon VPC (Virtual Private Cloud)

Amazon VPC allows you to provision a logically isolated section of the AWS Cloud. You define the virtual network space where you launch resources.

- **Scope:** A VPC spans an entire **Region**.
- **Isolation:** It logically isolates your network from other customers.
- **CIDR Block:** Defines the IP address range for the VPC (e.g., `10.0.0.0/16`).

2. Subnets (Organizing Resources)

A subnet is a segment of a VPC's IP address range where you place resources. Subnets are restricted to a single **Availability Zone (AZ)**.

- **Public Subnet:** Used for resources that need direct internet access (e.g., Customer-facing website, Load Balancer).
 - **Requirement:** Must have an **Internet Gateway** attached to the VPC and a route in the **Route Table** pointing to it.
 - **Auto-assign IP:** You often enable "Auto-assign public IPv4 address" here.
 - **Private Subnet:** Used for resources that should *not* be directly exposed to the internet (e.g., Database, Backend application).
 - **Security:** Better for sensitive data; no direct route to the Internet Gateway.
-

3. Gateways and Connectivity

Gateways act as "doors" controlling traffic into and out of your VPC.

- **Internet Gateway (IGW):** Allows traffic from the public internet to enter/exit your VPC. Required for Public Subnets.
 - **Virtual Private Gateway (VGW):** The VPN endpoint on the AWS side. Allows encrypted traffic from an approved private network (on-premises) to enter the VPC.
 - **NAT Gateway:** Allows instances in a **Private Subnet** to connect to the internet (e.g., for updates) but prevents the internet from initiating connections to them.
-

4. Connecting to AWS (Hybrid & Remote)

There are four main ways to connect remote networks or users to AWS:

1. **AWS Client VPN:**
 - **What:** Managed client-based VPN service.
 - **Use Case:** Connecting remote workers (roaming employees) securely to AWS.
2. **AWS Site-to-Site VPN:**
 - **What:** Encrypted tunnel between a data center/branch office and AWS VPC.
 - **Use Case:** Connecting a corporate office to the cloud. Uses the public internet (shared bandwidth).
3. **AWS Direct Connect (DX):**
 - **What:** Physical, dedicated fiber connection from your data center to AWS.
 - **Use Case:** High bandwidth, low latency, large data transfers, and regulatory compliance. Bypasses the public internet.
 - **Note:** Can use VPN as a *failover* for Direct Connect.
4. **AWS PrivateLink:**

- **What:** Connects your VPC to services (AWS services or other VPCs) privately.
- **Use Case:** Accessing third-party SaaS or internal services without exposing traffic to the public internet. Traffic stays on the AWS backbone.

Comparison: PrivateLink vs. Direct Connect

Feature	AWS PrivateLink	AWS Direct Connect
Connection Type	Virtual (VPC Endpoint)	Physical (Fiber cable)
Scope	VPC-to-Service (or VPC-to-VPC)	On-Premises-to-AWS
Traffic Path	Stays on AWS backbone	Dedicated line (bypasses internet)

5. Security: ACLs vs. Security Groups

You secure your network with two layers of firewalls.

Feature	Network ACL (NACL)	Security Group (SG)
Scope	Subnet Level	Instance Level (EC2)
Analogy	Passport Control (Border)	Doorman (Building Entry)
State	Stateless (Checks every packet, in & out)	Stateful (If allowed in, return traffic is allowed automatically)
Rule Types	Allow AND Deny	Allow only (Everything else is denied by default)
Default	Allows all traffic (Default NACL)	Denies all inbound traffic (Default SG)

6. Global Networking & Traffic

Services designed to route external users to your application.

- **Amazon Route 53:**
 - **What:** Highly available **DNS** (Domain Name System) service.
 - **Function:** Translates domain names (www.example.com) into IP addresses.
 - **Routing Policies:** Latency-based, Geolocation (routes user based on their location), Weighted.
- **Amazon CloudFront:**
 - **What:** **CDN** (Content Delivery Network).
 - **Function:** Caches content (images, videos) at **Edge Locations** close to users.
 - **Use Case:** Static web content, video streaming.

- **AWS Global Accelerator:**
 - **What:** Network service optimizing the path to your app over the AWS global network.
 - **Function:** Provides **Static IP addresses**. Routes traffic through the AWS backbone (not public internet).
 - **Use Case:** Non-HTTP traffic (UDP/Gaming), rapid failover, or when Static IPs are required.

Comparison: CloudFront vs. Global Accelerator

Feature	Amazon CloudFront	AWS Global Accelerator
Primary Goal	Cache Content	Optimize Network Path
IP Address	Dynamic	Static (Anycast)
Protocol	HTTP/HTTPS (Web)	TCP/UDP (Gaming, IoT, VoIP)

Module 5: Cram Cheat Sheet

VPC Basics

- **VPC:** Isolated network in the cloud.
- **Subnet:** Division of a VPC. Maps to **one AZ**.
- **Public Subnet:** Has route to **Internet Gateway**.
- **Private Subnet:** No direct route to Internet Gateway.
- **NAT Gateway:** Allows **Private Subnet** outbound access only.

Connectivity

- **Site-to-Site VPN:** Encrypted tunnel over public internet.
- **Direct Connect:** **Physical** dedicated line. No public internet. High speed/security.
- **PrivateLink:** Private connection to services via **VPC Endpoints**. Traffic stays on AWS network.

Security

- **Security Group:** **Stateful**. Instance level. Allow rules only.
- **Network ACL:** **Stateless**. Subnet level. Allow & Deny rules.

Global Traffic

- **Route 53:** DNS Service. Translates names to IPs.
- **CloudFront:** CDN. Caches content at Edge Locations. Low latency for web content.
- **Global Accelerator:** Uses AWS global network for performance. Provides **Static IPs**. Good for Gaming/UDP.

Module 5: Exam-Style Practice Questions

1. You have a requirement to block a specific IP address from accessing your subnets. Which security feature should you use?

- A) Security Group
- B) Internet Gateway
- C) Network Access Control List (Network ACL)
- D) Route Table

Correct Answer: C

Explanation: Network ACLs allow both Allow **and** Deny rules. Security Groups only support Allow rules; you cannot explicitly deny a specific IP in a Security Group.

2. A company requires a private, dedicated network connection between their on-premises data center and AWS that does not use the public internet. Which service meets this requirement?

- A) AWS Site-to-Site VPN
- B) AWS Client VPN
- C) AWS Direct Connect
- D) Amazon CloudFront

Correct Answer: C

Explanation: AWS Direct Connect provides a "dedicated private connection" (fiber) that bypasses the internet. VPNs use the public internet.

3. You are hosting a multiplayer game that uses the UDP protocol. You need a static IP address for your users to connect to, and you want to ensure the lowest possible latency by routing traffic over the AWS global backbone. Which service should you use?

- A) Amazon Route 53
- B) AWS Global Accelerator
- C) Amazon CloudFront
- D) AWS PrivateLink

Correct Answer: B

Explanation: Global Accelerator supports non-HTTP protocols (like UDP for gaming) and provides static IP addresses, whereas CloudFront is dynamic and optimized for HTTP content.

4. Which networking component is REQUIRED to make a subnet public?

- A) NAT Gateway
- B) Virtual Private Gateway
- C) Internet Gateway
- D) VPC Endpoint

Correct Answer: C

Explanation: The text states: "To allow traffic from the public internet... you must attach what is called an internet gateway... Without it, no one can reach the resources."

5. Your application is running on EC2 instances. You want to ensure that if traffic is allowed inbound to the instance, the response traffic is automatically allowed outbound without additional configuration. Which firewall feature provides this capability?

- A) Network ACL
- B) Security Group
- C) Route Table
- D) Subnet Mask

Correct Answer: B

Explanation: Security Groups are **stateful**. "With security groups, you allow specific traffic in, and by default, all [return] traffic is allowed out." Network ACLs are stateless and check traffic both ways.

Module 6 - AWS Storage Services

Here is the comprehensive summary, cheat sheet, and practice questions for **Module 6: Storage**, based strictly on the provided text.

1. Storage Categories

Data requires different storage solutions based on how it is accessed and used.

- **Block Storage:**

- Data is split into manageable "blocks."
- Allows editing small parts of a file (blocks) without rewriting the whole thing.
- **Ideal for:** Databases, applications requiring frequent updates, boot volumes.

- **Object Storage:**

- Data is stored as **Objects** in a flat address space (Buckets).
- Each object contains: Data + Metadata + Unique ID (Key).
- You cannot edit a part of an object; you must rewrite the entire object to update it.
- **Ideal for:** Unstructured data (images, videos), backups, logs, static websites.

- **File Storage:**

- Hierarchical structure (folders/directories).
- Shared access over a network (NFS/SMB).
- **Ideal for:** Shared content repositories, home directories, collaborative workloads.

2. Block Storage Services

Amazon EC2 Instance Store

- **Type:** Unmanaged, Ephemeral (Temporary).
- **Location:** Physically attached to the EC2 host computer.
- **Durability:** Data is LOST if the instance is stopped or terminated.
- **Performance:** Extremely high I/O (low latency) because it's local.
- **Use Case:** Buffers, caches, scratch data.

Amazon EBS (Elastic Block Store)

- **Type:** Managed, Persistent.
- **Location:** Network-attached (separate from the host).
- **Durability:** Data persists even if the instance is stopped. Replicated within the **same AZ** for availability.

- **Features:**

- **Snapshots:** Point-in-time backups stored in S3. They are **incremental** (only backup changed blocks).
- **Elastic:** Can be resized and modified while in use.
- **Use Case:** Databases, file systems, persistent application storage.

Amazon Data Lifecycle Manager: Automates the creation, retention, and deletion of EBS snapshots.

3. Object Storage: Amazon S3 (Simple Storage Service)

A fully managed service for storing any amount of data as objects in **Buckets**.

- **Durability:** 11 9s (99.999999999%).
- **Scalability:** Virtually unlimited storage. (Max object size: 5TB).
- **Security:** Private by default.
 - **Bucket Policies:** Resource-based policies (JSON) attached to the bucket.
 - **ACLs:** Legacy method for access.
 - **Block Public Access:** Feature to prevent accidental public exposure.

S3 Storage Classes

Class	Access Freq.	Speed	Use Case
S3 Standard	Frequent	Millisecond	Dynamic websites, active data.
S3 Standard-IA	Infrequent	Millisecond	Backups, DR. Cheaper storage, higher retrieval fee.
S3 Express One Zone	Most Frequent	Single-digit ms	AI/ML, HPC. Single AZ durability.
S3 One Zone-IA	Infrequent	Millisecond	Secondary backups. Single AZ .
S3 Intelligent-Tiering	Unknown/Changing	Millisecond	Automatically moves data between tiers to save cost.
S3 Glacier Instant	Rare (Archive)	Millisecond	Medical records needing fast access.
S3 Glacier Flexible	Very Rare	Mins to Hours	Tape replacement, long-term archives.
	Extremely Rare	12-48 Hours	

S3 Glacier Deep Archive

Lowest cost.
Compliance
archives.

S3 Lifecycle Policies: Rules to automate moving objects to cheaper tiers or deleting them after a set time (Transition & Expiration actions).

4. File Storage Services

Amazon EFS (Elastic File System)

- **Protocol:** NFS (Linux).
- **Scope: Regional** (Data replicated across multiple AZs).
- **Scaling:** Elastic. Grows/shrinks automatically as files are added/removed.
- **Access:** Simultaneous access by thousands of EC2 instances.
- **Use Case:** Linux shared storage, web serving, analytics.

Amazon FSx Family

Feature-rich, managed file systems for specialized workloads.

- **FSx for Windows File Server:** Native Windows compatibility (SMB). Good for AD integration.
- **FSx for Lustre:** High-performance computing (HPC), Machine Learning.
- **FSx for NetApp ONTAP:** Advanced data management (dedup, snapshots).
- **FSx for OpenZFS:** Managed ZFS file system.

Comparison: EBS vs. EFS

Feature	EBS	EFS
Type	Block (Hard Drive)	File (Network Share)
Access	Single Instance (mostly)	Thousands of Instances concurrently
Scope	Zonal (Single AZ)	Regional (Multi-AZ)
Scaling	Manual (Provisioned size)	Automatic (Elastic)

5. Hybrid Storage: AWS Storage Gateway

Bridges on-premises environments with AWS Cloud storage.

1. S3 File Gateway:

- Present cloud storage (S3) as a local file server (NFS/SMB) to on-prem apps.
- Uses local caching for low latency.

2. Volume Gateway:

- Presents cloud storage as block volumes (iSCSI).

- **Cached Mode:** Data in cloud, frequent data cached locally.
- **Stored Mode:** All data local, snapshots sent to cloud.

3. Tape Gateway:

- Replaces physical tape libraries.
 - Uses standard tape backup software to write to virtual tapes in S3/Glacier.
-

6. Disaster Recovery Service

AWS Elastic Disaster Recovery:

- Replicates block-level data from physical/virtual servers to AWS continuously.
 - Minimizes downtime and data loss (RPO/RTO).
 - **Cost:** Pay only for what you use (no standby infrastructure costs).
-

Module 6: Cram Cheat Sheet

Block Storage

- **Instance Store:** Ephemeral (Temporary). **Local** to host. Highest speed. **Data lost** on stop.
- **EBS:** Persistent. Network drive. **Zonal**. Backed up via **Snapshots** (stored in S3).
- **Snapshot:** Incremental backup. Only saves changed blocks.

Object Storage (S3)

- **S3 Standard:** General purpose.
- **S3 Intelligent-Tiering:** Auto-moves data based on access patterns. Good for **unknown** usage.
- **S3 Glacier Deep Archive:** **Cheapest** storage. 12hr retrieval. Compliance.
- **Bucket Policy:** JSON resource policy to control access (public/private).
- **Versioning:** Protects against accidental deletion/overwrite.

File Storage

- **EFS:** Linux (NFS). **Regional** (Multi-AZ). Auto-scales. Shared by many EC2s.
- **FSx for Windows:** Windows (SMB). Native Active Directory support.
- **FSx for Lustre:** HPC, AI/ML speeds.

Hybrid & DR

- **Storage Gateway:** Connects On-Prem to Cloud.
 - **File Gateway:** S3 as local file share.
 - **Tape Gateway:** Virtual Tape Library.
 - **Elastic Disaster Recovery:** Continuous replication of servers to AWS for failover.
-

Module 6: Exam-Style Practice Questions

1. A company needs to run a high-performance database on an EC2 instance. The data must persist even if the instance is stopped and restarted. Which storage option should they choose?

- A) Amazon S3
- B) Amazon EC2 Instance Store
- C) Amazon EBS
- D) Amazon EFS

Correct Answer: C

Explanation: EBS provides persistent block-level storage suitable for databases. Instance Store is ephemeral (data loss on stop). S3 is object storage (not for high-speed DBs).

2. You have a large dataset that is accessed infrequently but must be immediately available when requested. You want to minimize storage costs. Which S3 storage class is the BEST fit?

- A) S3 Standard
- B) S3 Glacier Deep Archive
- C) S3 Standard-Infrequent Access (Standard-IA)
- D) S3 Intelligent-Tiering

Correct Answer: C

Explanation: Standard-IA offers lower storage costs for infrequent access but ensures **millisecond (immediate) retrieval**. Glacier options have retrieval delays (minutes to hours).

3. A media company needs a shared file system that can be accessed simultaneously by hundreds of Linux EC2 instances across multiple Availability Zones. The storage must scale automatically as files are added. Which service should they use?

- A) Amazon EBS
- B) Amazon EFS
- C) Amazon S3
- D) Amazon Instance Store

Correct Answer: B

Explanation: EFS is a **Regional**, managed file system (NFS) that supports concurrent access by thousands of instances and scales automatically. EBS is Zonal and typically single-attach.

4. A hospital needs to archive patient records for 10 years to meet regulatory compliance. The records will likely never be accessed, but if they are, a retrieval time of 12 hours is acceptable. Which solution is the most cost-effective?

- A) S3 Standard
- B) S3 Glacier Instant Retrieval
- C) S3 Glacier Deep Archive
- D) AWS Storage Gateway

Correct Answer: C

Explanation: Glacier Deep Archive is the **lowest-cost** storage class designed for long-term retention where retrieval times of 12-48 hours are acceptable.

5. You want to automate the management of your EBS snapshots to ensure you are meeting your backup retention policies and optimizing costs. Which tool should you use?

- A) S3 Lifecycle Policies
- B) Amazon Data Lifecycle Manager
- C) AWS Elastic Disaster Recovery
- D) AWS Config

Correct Answer: B

Explanation: Amazon Data Lifecycle Manager is specifically designed to automate the creation, retention, and deletion of **EBS snapshots** based on schedules you define.

Module 7 - Databases

Here is the comprehensive summary, cheat sheet, and practice questions for **Module 7: Databases**, based strictly on the provided text.

1. Relational Databases (SQL)

Relational databases store data with predefined relationships (rows and columns) and use **Structured Query Language (SQL)** for querying.

- **Concept:** Data is stored in tables (e.g., Customer table, Order table) and linked via common attributes.
- **Lift-and-Shift:** Moving an existing on-premises database to an **Amazon EC2** instance.
 - **Control:** You retain full control over the OS, memory, CPU, and storage.
 - **Responsibility:** You manage backups, patching, and scaling manually.
- **Amazon RDS (Relational Database Service):** A fully managed service that automates administrative tasks.
 - **Managed Features:** AWS handles hardware provisioning, database setup, patching, and backups.
 - **Supported Engines:** Amazon Aurora, MySQL, PostgreSQL, MariaDB, Oracle, Microsoft SQL Server.
 - **Multi-AZ Deployment:** Automatically replicates data to a standby instance in a different Availability Zone for high availability and failover.
 - **Read Replicas:** Offloads read traffic from the primary instance to improve performance.
- **Amazon Aurora:** A cloud-native, fully managed relational database.
 - **Compatibility:** MySQL and PostgreSQL.
 - **Performance:** Up to **5x faster** than standard MySQL and **3x faster** than PostgreSQL.
 - **Storage:** Automatically scales from 10 GB up to 128 TB.
 - **Resilience:** Replicates data 6 times across 3 Availability Zones. Can handle the loss of up to 2 copies of data without affecting write availability.

2. NoSQL Databases

NoSQL databases use flexible schemas (non-relational) and are designed for high performance and scalability.

- **Amazon DynamoDB:** A fully managed, serverless, key-value NoSQL database.
 - **Structure:** Uses **Tables**, **Items** (rows), and **Attributes** (columns). No rigid schema (items can have different attributes).
 - **Performance:** Delivers **single-digit millisecond** latency at any scale.
 - **Scaling:** Automatically scales throughput up or down based on traffic.
 - **Global Tables:** Replicates data globally across AWS Regions for local access performance.
 - **Use Cases:** High-traffic web apps, gaming, e-commerce shopping carts.
-

3. In-Memory Caching

Caching improves database performance by storing frequently accessed data in high-speed memory (RAM) rather than retrieving it from a disk-based database every time.

- **Amazon ElastiCache:** Fully managed in-memory caching service.
 - **Engines:** Compatible with **Redis** and **Memcached**.
 - **Benefit:** Provides **microsecond** latency. Reduces load on the primary database (RDS).
 - **Use Cases:** Session management, gaming leaderboards, real-time analytics.
- **DynamoDB Accelerator (DAX):**
 - A purpose-built caching layer specifically for **DynamoDB**.
 - Improves read times from milliseconds to microseconds.

Comparison: Database with vs. without Cache

Scenario	Workflow	Performance
Without Cache	App queries Database directly for every request.	Slower (Disk I/O). High load on DB.
With ElastiCache	App checks Cache first. If data exists (Hit), return instantly. If not (Miss), query DB and update Cache.	Faster (Microsecond). Low load on DB.

4. Purpose-Built Databases

AWS offers specialized databases designed for specific data structures and use cases.

- **Amazon DocumentDB:**
 - **Type:** Document database.
 - **Compatibility:** **MongoDB**.
 - **Data Structure:** JSON-like documents.
 - **Use Cases:** Content management, catalogs, user profiles.

- **Amazon Neptune:**
 - **Type:** Graph database.
 - **Function:** optimized for storing and navigating highly connected datasets.
 - **Use Cases:** Social networks, fraud detection, recommendation engines.
 - **Amazon Managed Blockchain:**
 - **Type:** Blockchain.
 - **Function:** Creates and manages scalable blockchain networks.
 - **Use Cases:** Supply chain tracking, cryptographically verifiable ledgers.
-

5. Database Management Tools

- **AWS Database Migration Service (DMS):**
 - Helps migrate databases to AWS securely.
 - **Key Feature:** The source database remains fully operational during the migration (minimal downtime).
 - Supports homogeneous (Oracle to Oracle) and heterogeneous (Oracle to Aurora) migrations.
 - **AWS Backup:**
 - A centralized service to automate and manage data protection across AWS services.
 - **Scope:** Manages backups for RDS, DynamoDB, EBS volumes, EFS, and Storage Gateway.
 - **Features:** Cross-Region backup, compliance reporting, and centralized backup policies.
-

Module 7: Cram Cheat Sheet

Relational (SQL)

- **Amazon RDS:** Managed SQL. Engines: MySQL, Postgres, Oracle, SQL Server, MariaDB, Aurora. Automates patching/backups.
- **Amazon Aurora:** AWS-native. **MySQL/Postgres compatible.** High speed (5x MySQL). **Auto-scaling storage.** High availability (6 copies across 3 AZs).
- **Multi-AZ:** feature of RDS for **Disaster Recovery** (failover).
- **Read Replica:** feature of RDS for **Performance** (scaling reads).

NoSQL & Specialized

- **Amazon DynamoDB:** **Key-Value.** Serverless. **Millisecond** latency. Flexible schema.
- **Amazon DocumentDB:** **MongoDB** compatible. JSON documents.
- **Amazon Neptune:** **Graph** database. Complex relationships (Social sets).

Caching

- **Amazon ElastiCache:** **Redis/Memcached.** **Microsecond** latency. Stores data in **RAM**.
- **DAX:** Cache specifically for **DynamoDB**.

Tools

- **AWS DMS:** Migrates DBs with **minimal downtime**.
 - **AWS Backup:** Centralized backup dashboard for multiple AWS services.
-

Module 7: Exam-Style Practice Questions

1. A company wants to migrate an on-premises MongoDB database to AWS. They want a fully managed service that supports their existing MongoDB workloads with minimal code changes. Which service should they use?

- A) Amazon DynamoDB
- B) Amazon Aurora
- C) Amazon DocumentDB
- D) Amazon Neptune

Correct Answer: C

Explanation: Amazon DocumentDB is explicitly described as a "MongoDB-compatible database" designed to handle semi-structured data (JSON-like documents).

2. You are building a social networking application that needs to track complex relationships between users, such as friends of friends and interaction patterns. Which database service is optimized for this workload?

- A) Amazon RDS
- B) Amazon Neptune
- C) Amazon ElastiCache
- D) Amazon Redshift

Correct Answer: B

Explanation: Amazon Neptune is a "purpose-built graph database" that excels at understanding "highly connected data sets" like social networks.

3. An e-commerce application using Amazon RDS is experiencing performance issues due to a high volume of read traffic for product details. You need a solution to reduce the load on the database and improve retrieval speed to microseconds. What should you implement?

- A) Enable Multi-AZ for RDS
- B) Migrate to Amazon Glacier
- C) Implement Amazon ElastiCache
- D) Use AWS Database Migration Service

Correct Answer: C

Explanation: ElastiCache is an "in-memory caching service" designed to alleviate database pressure by storing frequently accessed data in RAM, providing microsecond latency.

4. A gaming company requires a database that can handle millions of requests per second with single-digit millisecond latency. The schema needs to be flexible to allow for changing game attributes. Which service meets these requirements?

- A) Amazon DynamoDB
- B) Amazon Aurora
- C) Amazon RDS for MySQL
- D) Amazon Neptune

Correct Answer: A

Explanation: DynamoDB is a "fully managed serverless NoSQL database" known for "single-digit millisecond performance," flexible schemas, and high scalability.

5. You need to migrate a mission-critical production database to AWS. It is imperative that the source database remains online and operational to serve users during the migration process.

Which service allows this?

- A) AWS Snowball
- B) Amazon S3
- C) AWS Database Migration Service (DMS)
- D) Amazon Connect

Correct Answer: C

Explanation: A key benefit of AWS DMS is that "the source database remains fully operational during migration, which minimizes downtime."

Module 8 - AI/ML and Data Analytics

Here is the comprehensive summary, cheat sheet, and practice questions for **Module 8: AI/ML and Data Analytics**, based strictly on the provided text.

1. Introduction to AI and Machine Learning

- **Artificial Intelligence (AI):** A broad field focused on developing intelligent computer systems capable of performing humanlike tasks.
- **Machine Learning (ML):** A subset of AI.
 - **Concept:** Training machines to perform complex tasks without explicit instructions (code rules).
 - **Training:** Finding patterns in historical data to create an **ML Model**.
 - **Inference:** Applying the trained model to new data to make predictions.
 - **Benefit:** More adaptable than "classical programming" (rule-based systems) which fail to account for individual preferences.

2. The AWS AI/ML Stack (Three Tiers)

AWS organizes its AI/ML offerings into three tiers based on customization and ease of use.

Tier 1: AWS AI Services (Pre-built)

Managed services with pre-trained models. No ML expertise required. Accessible via API.

- **Language Services:**

- **Amazon Comprehend:** NLP service. Extracts insights (sentiment, key phrases) from text. Used for content classification.
- **Amazon Polly:** Converts **Text to Speech**. Lifelike voices.
- **Amazon Transcribe:** Converts **Speech to Text**. Used for subtitles and call logs.

- **Amazon Translate:** Text translation across multiple languages.
- **Computer Vision & Search:**
 - **Amazon Rekognition:** Image and video analysis. Identifies objects, people, and text.
 - **Amazon Textract:** OCR. Extracts text, handwriting, and tables from documents/forms.
 - **Amazon Kendra:** Enterprise search service using NLP. Returns specific answers from documents rather than just links.
- **Conversational & Personalization:**
 - **Amazon Personalize:** Real-time product/content recommendations (ML-powered personalization).
 - **Amazon Lex:** Conversational interfaces (Chatbots/Voice bots) using ASR and NLU.

Tier 2: AWS ML Services (SageMaker)

For developers wanting to build their own models without managing infrastructure.

- **Amazon SageMaker AI:** A fully managed service to **Build, Train, and Deploy** ML models.
 - **IDE:** Integrated environment to track experiments, visualize data, and debug.
 - **Features:** Includes pre-trained models and tools for MLOps (standardizing workflows).

Tier 3: Frameworks & Infrastructure

For experts requiring complete control.

- **Frameworks:** PyTorch, TensorFlow, Apache MXNet.
 - **Infrastructure:** ML-optimized **EC2** instances, **Amazon EMR**, and **Amazon ECS**.
-

3. Generative AI (GenAI)

- **Deep Learning:** A subset of ML using artificial neural networks (mimicking the human brain) to solve complex problems.
- **Generative AI:** A type of deep learning capable of creating *new* content (images, stories, code).
 - **Foundation Models (FMs):** Extremely large models pre-trained on vast data. Capable of multi-tasking (unlike traditional ML models which are single-task).
 - **Large Language Models (LLMs):** FMs trained on text.

AWS GenAI Services:

1. **Amazon SageMaker JumpStart:** An ML hub with pre-built solutions and FMs. Deploy models with a few clicks and fine-tune them.
2. **Amazon Bedrock:** Fully managed service. Provides access to FMs (from Amazon and startups like AI21/Anthropic) via a **single API**. Serverless experience for GenAI.
3. **Amazon Q:** Interactive generative AI assistant.
 - **Amazon Q Business:** Connects to company repositories to answer questions and solve problems based on enterprise data.

- **Amazon Q Developer:** Code recommendations and generation (formerly CodeWhisperer functionality).

Comparison: SageMaker vs. Bedrock

Feature	Amazon SageMaker	Amazon Bedrock
Model	You build/train custom models OR use JumpStart.	Use pre-trained Foundation Models (FMs).
Management	Managed infrastructure (you control the instance).	Serverless (API access only).
Use Case	Specific predictive tasks (Fraud detection, churn).	Content generation, Chatbots, Summarization.

4. Data Analytics Fundamentals

Analytics transforms raw data into insights.

- **Data Lake:** A centralized repository (like **Amazon S3**) storing vast amounts of raw data (structured and unstructured).
- **ETL (Extract, Transform, Load):**
 - **Extract** data from sources.
 - **Transform** into a usable format (e.g., JSON to CSV).
 - **Load** into a destination (Data Warehouse).
- **Data Pipeline:** Automates the flow of ingesting, processing, and analyzing data.

5. Analytics Services (The Pipeline)

Ingestion (Getting Data In)

- **Amazon Kinesis Data Streams:** Real-time ingestion of streaming data. Low latency.
- **Amazon Data Firehose:** Near real-time ingestion. Aggregates data and loads it into destinations (S3, Redshift).
 - Note: Can invoke Lambda to transform data before delivery.

Cataloging & Processing (Preparing Data)

- **AWS Glue:** Managed ETL service.
 - **AWS Glue Data Catalog:** Central metadata repository. Crawls data in S3 to define schemas (tables) so other services (Athena) can understand the data structure.
- **Amazon EMR:** Big data processing using open-source frameworks (Spark, Hadoop, Hive). For large-scale data processing.

Analysis (Querying Data)

- **Amazon Athena: Serverless.** interactive analytics.
 - Runs **SQL queries** directly on data stored in **Amazon S3**.
 - Uses the Glue Data Catalog to understand data structure.
 - No infrastructure to manage; pay per query.
- **Amazon Redshift:** Fully managed **Data Warehouse**.
 - Optimized for complex SQL queries on massive structured datasets (Petabytes).
 - Uses columnar storage and parallel processing.

Visualization (Seeing Data)

- **Amazon QuickSight:** Business Intelligence (BI) service.
 - Creates interactive dashboards.
 - **Amazon Q in QuickSight:** Allows natural language queries (e.g., "Show me sales trends for Q4").
 - **Amazon OpenSearch Service:** Search and log analytics. Used for application monitoring and observability.
-

6. Architecture Example: E-Commerce Pipeline

A practical flow described in the text:

1. **Source:** App data stored in **Amazon DynamoDB**.
 2. **Ingest:** DynamoDB streams changes to **Kinesis Data Streams**.
 3. **Buffer/Load:** **Amazon Data Firehose** reads from Kinesis.
 4. **Transform:** Firehose triggers **Lambda** to convert JSON to CSV.
 5. **Storage:** Firehose delivers CSV to **Amazon S3** (Data Lake).
 6. **Catalog:** **AWS Glue** crawls S3 to create a metadata table.
 7. **Analyze:** Data Scientists use **Amazon Athena** to query S3 using SQL.
 8. **Train:** **SageMaker AI** reads from S3 to train ML models.
-

Module 8: Cram Cheat Sheet

AI Services (Tier 1)

- **Polly:** Text → Speech.
- **Transcribe:** Speech → Text.
- **Comprehend:** NLP/Sentiment analysis.
- **Rekognition:** Image/Video analysis.
- **Textract:** Extract text/data from documents (OCR).
- **Kendra:** Intelligent Enterprise Search.
- **Lex:** Chatbots/Conversational interfaces.

- **Personalize:** Recommendation engine.

GenAI

- **Bedrock:** API access to Foundation Models (FMs). Serverless.
- **SageMaker JumpStart:** Hub for pre-trained models.
- **Amazon Q:** GenAI Assistant (Business & Developer).

Analytics Pipeline

- **Kinesis Data Streams:** Real-time streaming.
 - **Data Firehose:** Near real-time delivery to S3/Redshift.
 - **AWS Glue:** ETL + Data Catalog (Metadata).
 - **Amazon Athena:** Serverless SQL queries on S3.
 - **Amazon Redshift:** Data Warehouse (Structured/Complex queries).
 - **Amazon QuickSight:** BI Dashboards.
 - **Amazon EMR:** Big Data frameworks (Spark/Hadoop).
-

Module 8: Exam-Style Practice Questions

1. A company wants to create a chatbot for their customer service website that utilizes natural language understanding (NLU) and automatic speech recognition (ASR). They do not have deep machine learning expertise. Which service should they use?

- A) Amazon Polly
- B) Amazon SageMaker AI
- C) Amazon Lex
- D) Amazon Connect

Correct Answer: C

Explanation: Amazon Lex is the AI service specifically designed for building conversational interfaces (chatbots) using voice and text.

2. A business analyst needs to run ad-hoc SQL queries on terabytes of data stored in an Amazon S3 data lake. They want a serverless solution with no infrastructure to manage.

Which service is the best fit?

- A) Amazon Redshift
- B) Amazon Athena
- C) Amazon EMR
- D) AWS Glue

Correct Answer: B

Explanation: Athena allows you to run SQL queries directly on data in S3. It is serverless and requires no infrastructure setup, unlike Redshift (which is a warehouse) or EMR (which requires clusters).

3. You need to ingest real-time streaming data from thousands of IoT sensors. The data must be processed with low latency. Which service should be the entry point for this data?

- A) Amazon Data Firehose
- B) Amazon S3
- C) Amazon Kinesis Data Streams
- D) AWS Glue

Correct Answer: C

Explanation: Kinesis Data Streams is designed for "real-time ingestion" of streaming data. Firehose is considered "near real-time" (seconds delay).

4. A developer wants to build a generative AI application that generates marketing copy. They want to access high-performing Foundation Models (FMs) from multiple providers via a single API without managing infrastructure. Which service should they choose?

- A) Amazon SageMaker
- B) Amazon Bedrock
- C) Amazon Kendra
- D) Amazon Q

Correct Answer: B

Explanation: Amazon Bedrock is the fully managed service that offers access to Foundation Models from Amazon and other startups via a unified API, specifically for GenAI apps.

5. Which service acts as a centralized metadata repository that allows Amazon Athena and Amazon EMR to discover and understand the schema of data stored in a Data Lake?

- A) Amazon QuickSight
- B) Amazon Redshift
- C) AWS Glue Data Catalog
- D) Amazon DynamoDB

Correct Answer: C

Explanation: The AWS Glue Data Catalog provides a centralized repository to store metadata (schemas/tables) about data, making it discoverable for analytics services like Athena.

Module 9 - Security

Here is the comprehensive summary, cheat sheet, and practice questions for **Module 9: Security**, based strictly on the provided text.

1. Introduction to Security Concepts

Security is a shared responsibility. To secure an environment, you must understand two core components:

- **Authentication:** Verifying identity (Who you are). *Example: Logging in with a username and password.*
- **Authorization:** Determining permissions (What you can do). *Example: Accessing only your specific employee records.*

Goals: Safeguard personal information, maintain trust, and prevent financial fraud/identity theft. AWS provides tools to prevent incidents, proactively address issues, and detect/respond to events.

2. Identity and Access Management (IAM)

This service controls access to AWS resources.

The Root User

- Created when you first open an AWS account.
- Has complete access to do anything (owner).
- **Best Practices:** Enable Multi-Factor Authentication (MFA), use a strong password, and **do not** use for daily tasks.

IAM Components

- **IAM Users:** Represent individual identities. By default, they have **zero permissions** (Implicit Deny).
- **Principle of Least Privilege:** You must explicitly grant only the permissions needed for a job, and nothing more.
- **IAM Policies:** JSON documents defining permissions (Effect: Allow/Deny, Action: API call, Resource: ARN).
- **IAM Groups:** Collections of users. Policies attached to a group apply to all members.
- **IAM Roles:**
 - Identities with **temporary** credentials (no password).
 - Assumed by users, applications, or services for a limited time.
 - Used for federation (corporate identity integration) and cross-account access.

IAM Identity Center

- Centralizes identity management across multiple AWS accounts.
 - Enables **Single Sign-On (SSO)** and Federated Identity Management (linking corporate credentials to AWS).
-

3. Secrets and Configuration Management

Tools to manage sensitive data and configuration settings securely.

AWS Secrets Manager

- Securely stores database credentials, API keys, and OAuth tokens.
- **Key Feature: Automatic Rotation** of secrets (e.g., changing DB passwords automatically).
- **Encryption:** Mandatory/Default encryption.

AWS Systems Manager (Parameter Store)

- Stores configuration data (AMI IDs, URLs) and simple secrets.
- **Difference from Secrets Manager:** No built-in automatic rotation; focuses on configuration management.

4. Network and Application Protection

Protecting infrastructure from attacks, specifically **DDoS (Distributed Denial of Service)** where "zombie bots" overwhelm an application.

Defensive Services

1. **Security Groups:** Acts as a firewall at the **AWS Network level** (not the instance level). Filters traffic before it reaches the instance, shrugging off massive attacks like UDP floods.
 2. **Elastic Load Balancing (ELB):** Scales to absorb traffic and shields backend servers.
 3. **AWS Shield:**
 - **Shield Standard: Free.** Protects against common/frequent DDoS attacks. Built into CloudFront and Route 53.
 - **Shield Advanced: Paid.** Protects against sophisticated attacks, offers diagnostics, and access to the Shield Response Team (SRT).
 4. **AWS WAF (Web Application Firewall):**
 - Filters web traffic based on rules (Web ACLs).
 - Blocks exploits like SQL injection or traffic from specific IP addresses.
 - Can block malicious requests while letting legitimate ones through.
-

5. Data Protection (Encryption)

Securing data prevents unauthorized access even if data is stolen.

Types of Encryption

- **At Rest:** Data is idle (e.g., on a hard drive).
- **In Transit:** Data is moving across a network.

Encryption Services

- **AWS KMS (Key Management Service):**
 - Creates and manages **Cryptographic Keys** used to encrypt/decrypt data.
 - Keys never leave KMS. You control who manages/uses the keys.
 - Integrated with EBS, S3, DynamoDB, etc.
- **AWS Certificate Manager (ACM):**
 - Manages **SSL/TLS certificates**.
 - Used for **Encryption In Transit (HTTPS)** to secure network connections.
- **Amazon Macie:**
 - Uses Machine Learning to discover **sensitive data (PII)** stored in **Amazon S3**.
 - Helps meet compliance by assessing security posture.

6. Detection and Response

Services that monitor your environment for issues *after* they occur or to find potential risks.

Amazon Inspector

- **Focus:** Vulnerabilities and Software exposures.
- **Action:** Runs automated assessments on **EC2 instances**, Containers, and Lambda functions.
- **Result:** Lists security findings (e.g., open ports, vulnerable software versions) with remediation steps.

Amazon GuardDuty

- **Focus:** Threat Detection.
- **Action:** Continuously monitors **metadata** and network activity (logs). Uses ML and anomaly detection.
- **Result:** Identifies compromised instances, malicious IPs, or unauthorized access.

Amazon Detective

- **Focus:** Root Cause Investigation.
- **Action:** Collects log data to build interactive visualizations.
- **Result:** Helps visualize user/resource interactions over a timeline to understand *how* a breach happened.

AWS Security Hub

- **Focus:** Centralization.
 - **Action:** Aggregates findings from multiple services (GuardDuty, Inspector, Macie) into a single dashboard.
 - **Result:** Provides a comprehensive view of security and compliance state; automates remediation.
-

15 Hard Questions - Data Pipelines

CCP AWS

Done — I removed the **15 advanced practice questions** from the canvas document and left the rest (diagrams, cheat sheet, mnemonics, etc.) intact.

Below are the **15 advanced practice questions** (medium→high intensity), provided separately as you requested. Answers are included at the end.

Advanced Practice Questions (Q1–Q15)

Q1. A campaign analytics team must compute 1-minute tumbling window aggregates on clickstream events with exactly-once semantics. They also want to enrich events with user profile data stored in DynamoDB. Which architecture best satisfies latency, ordering, and enrichment requirements?

- A. Kinesis Data Streams → Lambda consumer (updates DynamoDB per event) → write aggregates to DynamoDB
 - B. Kinesis Data Streams → Kinesis Data Analytics (Flink) with stateful joins to DynamoDB via lookup function → write results to S3
 - C. Firehose → S3 → Athena scheduled queries
 - D. Kinesis Data Streams → Firehose → Redshift
-

Q2. A security team wants to index CloudTrail logs for near real-time search and also run long-term forensic analytics. They need sub-second search of new logs and cost-efficient long-term storage. Choose the best hybrid architecture.

- A. Firehose → OpenSearch for indexing; Firehose also archives raw logs to S3 (Parquet) for Athena/Glue
 - B. Kinesis Streams → EMR → Redshift
 - C. Firehose → S3 only, and use Athena for searching
 - D. S3 → OpenSearch via Glue
-

Q3. You have small JSON files in S3 with inconsistent schemas. Analysts want fast interactive SQL and predictable Athena performance. Which two-step transformation minimizes scan cost and schema issues?

- A. Use Glue crawler then query raw JSON with Athena
 - B. Run Glue jobs to normalize schema and write Parquet partitioned files; register tables in Glue Catalog
 - C. Move data into Redshift using COPY of JSON files
 - D. Re-index into OpenSearch and query
-

Q4. A dataset requires complex graph processing and iterative ML training across many node-to-node shuffles. Which service is best suited to this workload?

- A. Athena
 - B. EMR with Spark / Hadoop
 - C. Glue
 - D. Firehose
-

Q5. Your company uses Redshift for dashboards but wants to query colder historical partitions on S3 without copying data into Redshift. What feature enables this integration?

- A. Redshift Spectrum
 - B. Redshift Snapshot Export
 - C. Athena Federated Query
 - D. Glue Streaming
-

Q6. A pipeline must transform streaming telemetry in Python, preserve ordering, and allow consumers to rewind processing to an earlier offset for reprocessing. Which service combination gives you the best control?

- A. Kinesis Data Streams with consumer applications (KCL or enhanced fan-out) + custom checkpointing (e.g., DynamoDB offsets)
 - B. Firehose with Lambda transforms
 - C. Kinesis Analytics SQL only
 - D. SQS FIFO queues with Lambda
-

Q7. A data product team needs a serverless, discoverable metadata layer that integrates with Athena and EMR and supports schema versioning. They also want automated schema inference. Which combination is correct?

- A. Glue Data Catalog + Glue Crawlers
 - B. Redshift Catalog + Glue Jobs
 - C. Athena internal catalog only
 - D. OpenSearch mappings
-

Q8. A BI team complains that QuickSight dashboards are slow when querying Redshift during peak hours. Which option reduces latency and protects Redshift concurrency without changing dashboards?

- A. Move QuickSight to use Athena instead
 - B. Use QuickSight SPICE to import dataset into in-memory engine
 - C. Increase Redshift node count only
 - D. Re-write queries to use Glue
-

Q9. A compliance requirement mandates isolation of workloads on physical hosts due to licensing restrictions. Which EC2-level option gives you full host control and license visibility while still using EC2 for Redshift or EMR nodes?

- A. Dedicated Hosts for EC2 instances backing your cluster
 - B. Spot Instances only
 - C. Savings Plans
 - D. Kinesis Data Streams
-

Q10. A data team wants low-cost, large-scale ad-hoc analytics. They have massive cold datasets on S3 and want pay-per-query. However, many SQLs scan whole files repeatedly. Which combined strategy is best to lower cost and improve performance?

- A. Convert files to columnar Parquet + partition appropriately; use Glue Catalog; employ Athena with partition projection if needed
 - B. Move everything to Redshift and stop using Athena
 - C. Run EMR to precompute all views daily and store results only
 - D. Index everything into OpenSearch
-

Q11. You need to run Spark jobs that use a custom native library requiring privileged OS access. Which service offers the greatest control to install and run native dependencies on cluster nodes?

- A. EMR (provisioned EC2 nodes)
 - B. Glue (serverless)
 - C. Athena
 - D. Firehose
-

Q12. An architecture currently uses Firehose → OpenSearch. During peaks, OpenSearch cluster is overwhelmed by indexing throughput causing throttling and increased latency. What mitigation is the best next step to absorb bursts and protect OpenSearch while maintaining near real-time indexing?

- A. Add a buffering S3 stage: Firehose delivers to S3 and a separate worker batch-indexes to OpenSearch with backpressure controls.
 - B. Increase OpenSearch instance types immediately.
 - C. Switch Firehose to Kinesis Data Streams.
 - D. Stop ingesting logs during peak hours.
-

Q13. A data scientist wants to perform exploratory joins between a large S3 data lake and an operational RDS database without moving data. Which modern serverless tools can accomplish cross-source querying?

- A. Athena with federated query connectors (for RDS)
 - B. Redshift COPY only
 - C. Glue ETL moving RDS to S3 first
 - D. OpenSearch cross-cluster search
-

Q14. A pipeline processes clickstream events and writes both to a real-time analytics dashboard (seconds latency) and a historical S3 data lake for batch ML. Which pattern is most efficient and cost-effective?

- A. Producer → Kinesis Data Streams; one consumer = KDA/real-time dashboard; another consumer = Firehose to S3 (or KCL consumer that writes to Firehose)
 - B. Producer → Firehose only to S3; dashboards read S3 via Athena
 - C. Producer → direct PUT to Redshift and OpenSearch simultaneously
 - D. Producer → SQS → Lambda → write to S3 and OpenSearch
-

Q15. A company wants to run ML model training on a curated dataset in Redshift, but the training framework requires Spark on many nodes and GPU acceleration. What is a good approach?

- A. Unload curated dataset from Redshift to S3 (Parquet), spin up EMR or SageMaker training cluster (GPU instances), and run distributed training on data in S3.
 - B. Run training inside Redshift using UDFs.
 - C. Stream data via Firehose to OpenSearch and train there.
 - D. Use Athena to train models directly.
-

Answers (Q1–Q15)

1. B
 2. A
 3. B
 4. B
 5. A
 6. A
 7. A
 8. B
 9. A
 10. A
 11. A
 12. A
 13. A
 14. A
 15. A
-

Would you like these converted into a printable PDF, a spaced-repetition flashcard deck, or another 15 *very high* intensity questions?

Module 9: Cram Cheat Sheet

Identity (IAM)

- **Root User:** Owner. Unlimited power. Enable MFA.
- **IAM User:** Individual. Default = **Deny All**.
- **IAM Role:** **Temporary** credentials. Used for EC2, Services, and Federation.
- **Least Privilege:** Grant only necessary permissions.
- **Identity Center:** SSO / Federation.

Network Defense

- **AWS Shield: DDoS** protection. Standard = Free. Advanced = Paid/Support.
- **AWS WAF: Web App Firewall**. Blocks SQL injection, XSS, bad IPs (Layer 7).
- **Security Group:** Stateful firewall. Filters traffic at the **Network level**.

Data Security

- **KMS:** Manages **Encryption Keys** (At Rest).
- **ACM:** Manages **SSL/TLS Certificates** (In Transit).
- **Secrets Manager:** Rotates DB credentials/API keys **automatically**.
- **Macie:** Finds sensitive data (PII) in **S3**.

Monitoring & Response

- **Inspector:** Finds **Vulnerabilities** (software bugs/network exposure) in EC2.
 - **GuardDuty:** Finds **Threats** (intelligent detection) using logs/ML.
 - **Detective:** visualizes **Root Cause** of security issues.
 - **Security Hub:** Single dashboard for all security findings.
-

Module 9: Exam-Style Practice Questions

1. A company wants to ensure that their database credentials are rotated automatically every 30 days without custom coding. Which service should they use?

- A) AWS Systems Manager Parameter Store
- B) AWS Secrets Manager
- C) AWS Key Management Service (KMS)
- D) AWS IAM Identity Center

Correct Answer: B

Explanation: While both Parameter Store and Secrets Manager can store secrets, the text explicitly states that Secrets Manager "provides a secure way to manage, rotate, and retrieve database credentials" and has built-in automated rotation, whereas Parameter Store does not.

2. Which service uses machine learning to automatically discover and classify sensitive data, such as personally identifiable information (PII), stored in Amazon S3 buckets?

- A) Amazon Inspector
- B) Amazon GuardDuty
- C) Amazon Macie
- D) AWS Shield

Correct Answer: C

Explanation: Amazon Macie uses machine learning and automation to discover sensitive data stored in Amazon S3.

3. You need to protect your web application from common web exploits like SQL injection and cross-site scripting. You also want to block traffic from specific IP addresses. Which service is best suited for this?

- A) AWS Shield Standard
- B) AWS WAF
- C) Security Groups
- D) Amazon Inspector

Correct Answer: B

Explanation: AWS WAF (Web Application Firewall) monitors network requests and checks them against a web access control list (web ACL) to block common web exploits and specific IPs.

4. A security team needs a centralized view of the security state of all their AWS accounts and wants to aggregate findings from services like GuardDuty and Inspector into a single dashboard. Which service should they use?

- A) Amazon Detective
- B) AWS IAM
- C) AWS Security Hub
- D) AWS Artifact

Correct Answer: C

Explanation: AWS Security Hub brings multiple security services together into a single place, automatically aggregating security findings into one comprehensive view.

5. Which AWS service provides intelligent threat detection by continuously monitoring network activity and account metadata for malicious behavior, such as communication with known malicious IP addresses?

- A) Amazon GuardDuty
- B) Amazon Inspector
- C) AWS Shield
- D) AWS KMS

Correct Answer: A

Explanation: Amazon GuardDuty identifies threats by continuously monitoring streams of account metadata and network activity, using anomaly detection and machine learning. Inspector focuses on software vulnerabilities, not active threat monitoring.

30 CCP-style exam questions - Security

CCP AWS

Got it — here are **30 CCP-style exam questions** about AWS security: **15 “normal”** (typical CCP difficulty) and **15 “hard”** (scenario / multi-step). Each question is multiple choice (one best answer). **Answers and short explanations follow each section** so you can self-test then check reasoning.

Section A — 15 Normal Questions (CCP level)

N1. Which statement about the AWS root user is correct?

- A. Use the root user for daily administrative tasks.
- B. Root user has unlimited privileges and you should enable MFA and lock away its credentials.
- C. Root user cannot be used to create IAM users.
- D. Root user permissions are limited to billing only.

N2. By default, an IAM user in AWS has which permissions?

- A. Full administrator permissions.
- B. Read-only across all AWS services.
- C. Deny all until you explicitly grant permissions.
- D. Access to billing and IAM only.

N3. Which of the following best describes an IAM role?

- A. A permanent user account for developers.
- B. Temporary credentials assumed by services, applications, or federated users.
- C. A password manager for IAM users.
- D. A service that issues X.509 certificates.

N4. What does “least privilege” mean in IAM policy design?

- A. Give full admin rights to everyone to avoid permission errors.
- B. Grant users only the permissions they need to perform their job and no more.
- C. Remove all permissions and rely on root for everything.
- D. Grant all permissions but restrict by time.

N5. AWS Identity Center (formerly AWS SSO) is primarily used for:

- A. Managing VPC networks.
- B. Single Sign-On and federation to multiple AWS accounts and apps.
- C. Storing encryption keys.
- D. Performing network intrusion detection.

N6. Which service provides automatic, always-on DDoS protection at no extra cost?

- A. AWS Shield Advanced
- B. AWS WAF
- C. AWS Shield Standard
- D. AWS GuardDuty

N7. AWS WAF is best used to:

- A. Encrypt S3 objects at rest.
- B. Block web attacks such as SQL injection and XSS at Layer 7.
- C. Replace Security Groups for VPC-level traffic filtering.
- D. Rotate database credentials.

N8. A Security Group in AWS is:

- A. A stateless packet filter applied at the subnet level.
- B. A stateful virtual firewall that controls inbound and outbound traffic to instances.
- C. A tool to manage IAM groups.
- D. A DNS firewall.

N9. Which AWS service manages encryption keys used to encrypt data at rest?

- A. ACM
- B. Secrets Manager
- C. KMS (Key Management Service)
- D. Macie

N10. Which service issues and manages SSL/TLS certificates for your load balancers and websites?

- A. KMS
- B. ACM (AWS Certificate Manager)
- C. Secrets Manager
- D. GuardDuty

N11. Where should you store database credentials so they can be automatically rotated?

- A. In an S3 object without encryption
- B. In a text file on an EC2 instance
- C. In AWS Secrets Manager
- D. As a parameter in Route 53

N12. Amazon Macie is primarily used for:

- A. Detecting vulnerabilities in EC2 AMIs.
- B. Discovering and classifying sensitive data (like PII) in S3.
- C. Managing IAM policies.
- D. Blocking invalid HTTP requests.

N13. Which service continuously analyzes events and logs to detect suspicious activity and threats?

- A. Inspector
- B. GuardDuty
- C. Security Hub
- D. Detective

N14. AWS Inspector is used to:

- A. Centrally manage security findings from other services.
- B. Visualize event graphs and root cause.
- C. Scan EC2 instances for vulnerabilities and misconfigurations.
- D. Replace WAF for web filtering.

N15. Which service aggregates and presents security findings across multiple AWS services in a single dashboard?

- A. Detective
 - B. GuardDuty
 - C. Security Hub
 - D. Inspector
-

Answers & short explanations — Normal

1. **N1: B.** Root has full privileges; best practice is enable MFA and avoid daily use.
 2. **N2: C.** IAM users have no permissions until granted (default deny).
 3. **N3: B.** Roles provide temporary credentials for services, EC2 instance profiles, federated users.
 4. **N4: B.** Least privilege = give only necessary permissions.
 5. **N5: B.** Identity Center = SSO/federation across accounts/apps.
 6. **N6: C.** Shield Standard is always-on and free; Advanced is paid with extra protections.
 7. **N7: B.** WAF protects at Layer 7 (web app attacks).
 8. **N8: B.** Security Groups are stateful virtual firewalls at instance/network interface level.
 9. **N9: C.** KMS manages keys for encryption at rest.
 10. **N10: B.** ACM issues/manages SSL/TLS certs for AWS services.
 11. **N11: C.** Secrets Manager stores and can rotate credentials.
 12. **N12: B.** Macie discovers/classifies sensitive data in S3.
 13. **N13: B.** GuardDuty uses logs & ML to detect threats.
 14. **N14: C.** Inspector scans EC2/ECR/etc. for vulnerabilities.
 15. **N15: C.** Security Hub consolidates findings from many services.
-

Section B — 15 Hard Questions (scenario-based, higher difficulty)

H1. Your organization enforces that the root user credentials must never be used and MFA is mandatory. Which of the following is the MOST secure operational control?

- A. Delete the root user.
- B. Store root credentials in Secrets Manager and rotate them monthly.
- C. Remove root user password, enable MFA, and store root credentials offline in a secure vault with strict access controls.
- D. Create an IAM user with AdministratorAccess and use that instead.

H2. A developer needs temporary access to upload objects to an S3 bucket for a CI job running on EC2 without embedding long-term credentials on the instance. Best practice approach:

- A. Store access keys in a file on the EC2 instance.
- B. Create an IAM user, place keys in Secrets Manager, and pull them from the EC2 instance.
- C. Attach an IAM role to the EC2 instance with least-privilege S3 permissions.
- D. Use the root user credentials for the EC2 job.

H3. You must allow employees to sign into all AWS accounts using corporate credentials (Okta). You also want centralized permission control and just-in-time access. Which service combination should you implement?

- A. Create IAM users in each account and synchronize passwords.
- B. Use Identity Center (AWS SSO) integrated with your corporate IdP (Okta) plus permission sets (least privilege).
- C. Configure public key authentication for all users.
- D. Share the same IAM user across accounts.

H4. A public web application receives Layer-7 attacks. You need to block SQL injection and rate-limit traffic from abusive IPs before it hits the application but still allow legitimate users. Which combination is best?

- A. Security Group + Shield Standard
- B. WAF (with rules for SQLi and rate limiting) in front of CloudFront or ALB + Shield Standard
- C. KMS + ACM
- D. GuardDuty + Inspector

H5. An S3 bucket contains sensitive PII. You need to detect if objects containing PII are accidentally stored there and alert the security team. Which service helps detect and classify PII in S3 and integrates with Security Hub?

- A. Macie
- B. GuardDuty
- C. Inspector
- D. WAF

H6. Your compliance team requires automatic rotation of RDS credentials used by an application. Which AWS service provides built-in rotation for database credentials and integrates with IAM/KMS?

- A. SSM Parameter Store (unencrypted)
- B. Secrets Manager (automatic rotation)
- C. KMS (manages keys only)
- D. ACM

H7. A compromised EC2 instance may perform reconnaissance by calling AWS APIs. Which service monitors AWS account activity (API calls) and can generate findings about unusual API behavior?

- A. GuardDuty (using CloudTrail data)
- B. WAF
- C. KMS
- D. Macie

H8. Inspector finds a set of vulnerable packages on an EC2 fleet. You want an automated workflow: collect findings, open tickets, and trigger remediation runbooks. Which AWS service should you use to centralize findings and orchestrate responses?

- A. Security Hub to aggregate findings, AWS Systems Manager Automation for remediation, and SNS/ITSM for tickets.
- B. KMS only.
- C. ACM only.
- D. WAF only.

H9. You need to give an external partner temporary access to a single S3 prefix in one account without creating an IAM user. The partner uses their own AWS account. Best solution:

- A. Create an IAM user for the partner in your account.
- B. Create an IAM role in your account that the partner can assume (cross-account role) with least privilege to the prefix.
- C. Share root credentials.
- D. Put the objects in a public S3 bucket.

H10. You suspect unusual lateral movement inside your VPC. You want a service that analyzes VPC flow logs, CloudTrail, and DNS logs to surface suspicious activity and provide root-cause graphs. Which service is the best fit?

- A. AWS Detective
- B. AWS Inspector
- C. AWS WAF
- D. Secrets Manager

H11. Your external web application uses a TLS cert from ACM attached to an Application Load Balancer. A compliance audit requires central rotation tracking and no private key export. Why is ACM a good choice?

- A. ACM stores private keys centrally and allows exporting them to anyone.
- B. ACM issues/manages certificates with automatic renewal and the private keys are not exportable when ACM-provided certs are used.
- C. ACM cannot be used with ALB.
- D. ACM only supports self-signed certs.

H12. A critical data breach detected by GuardDuty triggers an alert. To investigate further, you want an aggregated view of all security alerts across accounts and automate cross-account alert forwarding. Which AWS service is purpose-built for aggregating findings and enabling cross-account, cross-region views?

- A. Security Hub
- B. Inspector
- C. ACM
- D. KMS

H13. Your team wants to block suspicious IP addresses at the edge (CloudFront) globally and also log all blocked requests. Which services should you combine?

- A. ACLs only on the VPC subnet.
- B. AWS WAF (IP block list + managed rules) associated with CloudFront + log to S3 or Kinesis Data Firehose.
- C. Security Groups + Inspector.
- D. GuardDuty + Macie.

H14. You must ensure encryption of S3 objects at rest with a key you control and audit all key usage. Which services and features do you use?

- A. Server-side encryption with S3-managed keys only (SSE-S3).
- B. SSE-KMS using a KMS CMK (customer-managed key) with CloudTrail logging of KMS API calls.
- C. Encrypt objects on the client only (no KMS).
- D. Store objects unencrypted and rely on network controls.

H15. A developer requests the ability to read secrets from Secrets Manager but should NOT be able to list, delete, or change rotation. Which policy approach enforces least privilege?

- A. Attach AdministratorAccess to the developer.
 - B. Grant an IAM policy allowing only `secretsmanager:GetSecretValue` on the specific secret resource and deny other secretsmanager actions.
 - C. Give access to all secrets in the account.
 - D. Store the secret in SSM Parameter Store (unencrypted) instead.
-

Answers & brief explanations — Hard

- H1: C.** You can't delete root. Best practice: remove root password or store offline and enable MFA; strict vault storage.
- H2: C.** Attach an instance role (instance profile) to EC2 with least-privilege S3 put permissions — no long-term keys.
- H3: B.** Identity Center + corporate IdP provides SSO/federation and centralized permission sets (just-in-time).
- H4: B.** WAF (SQLi rules + rate limiting) fronting CloudFront or ALB plus Shield Standard for DDoS protection.
- H5: A.** Macie discovers and classifies PII in S3 and integrates with Security Hub.
- H6: B.** Secrets Manager supports automatic rotation for RDS and integrates with KMS to encrypt secrets.
- H7: A.** GuardDuty analyzes CloudTrail and other logs to detect suspicious API activity.
- H8: A.** Use Security Hub to aggregate findings, then Systems Manager Automation (or Lambda/Step Functions) + SNS/ITSM to create tickets and remediate.
- H9: B.** Create a cross-account role that the partner can assume for a specific S3 prefix — no user needed.
- H10: A.** Detective builds visualizations/graphs from VPC flow, CloudTrail, DNS logs to investigate lateral movement.
- H11: B.** ACM issues and renews certs, and ACM-managed certs keep private keys non-exportable — good for compliance.
- H12: A.** Security Hub aggregates findings from GuardDuty, Inspector, Macie, etc., across accounts/regions.
- H13: B.** WAF associated with CloudFront blocks at edge; logs can be pushed to S3/Firehose for auditing.
- H14: B.** Use SSE-KMS with a customer-managed CMK and enable CloudTrail logging of KMS APIs for auditing.
- H15: B.** Grant only `GetSecretValue` on the specific resource (least privilege); deny or omit list/delete/rotate actions.
-

How to use these

- Take the **normal 15** as warm-up; time yourself (about 1.5–2 minutes per question).
- Use the **hard 15** to practice multi-step, architecture and control reasoning — annotate which services appear in each solution.
- If you want, I can:
 - Turn these into a **printable PDF**,
 - Make a **flashcard deck** ($Q \rightarrow$ hide answer until reveal),
 - Generate **detailed explanations** and AWS console links for any question, or
 - Produce **15 more very-high intensity** (deep architecture / policy writing) questions.

Which output would you like next?

Module 10 - Monitoring, Compliance and Governance

Here is the comprehensive summary, cheat sheet, and practice questions for **Module 10: Monitoring, Compliance, and Governance**, based strictly on the provided text.

1. Monitoring (Observability)

Monitoring involves observing systems, collecting metrics, and using that data to make decisions. It ensures reliability, availability, and performance.

Amazon CloudWatch

A centralized service to monitor infrastructure and applications in real-time.

- **Metrics:** Variables tied to resources (e.g., CPU utilization, "Espresso Count").
- **Alarms:** Triggers notifications (via **Amazon SNS**) or actions (e.g., Auto Scaling) when a metric crosses a defined threshold.
- **Dashboards:** Customizable home pages providing a single view of resources with auto-refreshing graphs.
- **Logs:** Centralizes log files from various AWS resources for searching, filtering, and analysis (e.g., looking for specific error codes).

AWS CloudTrail

A service that audits **API calls** and user activity.

- **Function:** Logs every request made to AWS (who, when, what, from where).
- **Event History:** Viewable, searchable record of the past **90 days** of management events.
- **CloudTrail Insights:** Detects anomalies in API usage (e.g., spikes in error rates).
- **Integrity:** Logs are stored in S3 and can be validated to prove they haven't been tampered with.

Comparison: CloudWatch vs. CloudTrail

- **CloudWatch:** Monitors **Performance** (Health, Metrics, Utilization).
- **CloudTrail:** Monitors **Activity** (API calls, Who did what).

2. Compliance and Auditing

AWS manages the security of the cloud, but customers must ensure their workloads comply with regulations (GDPR, HIPAA, PCI).

AWS Artifact

A self-service portal for on-demand access to compliance documents.

- **Artifact Reports:** Third-party audit reports (e.g., ISO, SOC) verifying AWS security controls.
- **Artifact Agreements:** Manage legal agreements with AWS (e.g., BAA for HIPAA).

AWS Config

A service to assess, audit, and evaluate resource **configurations**.

- **Function:** Continuously records configuration changes (e.g., "Was this S3 bucket encrypted yesterday?").
- **Rules:** Evaluates configurations against desired policies. Can trigger remediation for non-compliant resources.

AWS Audit Manager

Automates evidence collection for audits.

- **Function:** Maps AWS usage to pre-built compliance frameworks.
 - **Relationship:** It takes findings from **AWS Config** and Security Hub and packages them into audit-ready reports for external auditors.
-

3. Management and Governance

As organizations grow from single accounts to multiple accounts, they need tools for centralized management.

AWS Organizations

Account management service to consolidate multiple AWS accounts into a central hierarchy.

- **Structure:** Root Account → Organizational Units (OUs) → Member Accounts.
- **Consolidated Billing:** All bills roll up to the primary account; enables volume discounts.
- **Service Control Policies (SCPs):**
 - JSON policies that specify the **maximum permissions** for member accounts.
 - Applied to the Root, OUs, or specific accounts.
 - **Key Distinction:** SCPs affect the **Root User** of the member account.

AWS Control Tower

The easiest way to set up and govern a secure, multi-account AWS environment.

- **Landing Zone:** Automatically sets up a well-architected multi-account environment with best practices (logging, security) pre-configured.
- **Guardrails:** High-level rules that provide ongoing governance (can be Preventive or Detective).
- **Blueprints:** Automates setup with federated access and identity management.

AWS Service Catalog

Allows organizations to create and manage a catalog of **approved** IT services.

- **Benefit:** Enables self-service for employees to launch resources without violating governance rules.
- **Use Case:** A developer needs an EC2 instance; they select a pre-approved configuration from the catalog.

AWS License Manager

Helps manage software licenses (e.g., Microsoft, Oracle) on AWS.

- **BYOL:** Supports the "Bring Your Own License" model to track usage and prevent non-compliance.
-

4. Health and Optimization

Tools to maintain the health of the AWS environment and optimize costs and security.

AWS Health Dashboard

- **Personal Health Dashboard:** Provides a personalized view of the performance and availability of the AWS services underlying your specific AWS resources.
- **Service Health:** Notifications about general AWS service events.

AWS Trusted Advisor

An automated tool that evaluates your account against AWS best practices across **five categories**:

1. **Cost Optimization:** (e.g., Idle RDS instances, unattached EBS volumes).
2. **Performance:** (e.g., High utilization on EC2).
3. **Security:** (e.g., Security groups allowing 0.0.0.0/0, Root user MFA status).
4. **Fault Tolerance:** (e.g., EBS snapshots missing, Single-AZ deployments).
5. **Service Limits:** (e.g., Approaching the limit of VPCs per region).

IAM Access Analyzer

Analyzes resource-based policies to identify external access. Helps refine permissions to achieve **Least Privilege**.

Module 10: Cram Cheat Sheet

Monitoring & Logging

- **CloudWatch:** Metrics, Alarms, Dashboards, Logs. Focus = **Performance**.
- **CloudTrail:** API Logs, Event History (90 days). Focus = **Auditing/User Activity**.

Compliance Tools

- **AWS Artifact:** Download third-party reports/agreements (SOC, HIPAA).
- **AWS Config:** Records configuration changes over time. Checks rules.
- **AWS Audit Manager:** Automates **evidence collection** for external auditors.

Multi-Account Management

- **AWS Organizations:** Central billing, grouping (OUs), and **SCPs**.
- **SCP (Service Control Policy):** Restricts what the **Root User** in a member account can do.
- **AWS Control Tower:** Automates the setup of a **Landing Zone** (secure baseline). Uses Guardrails.

Governance & Optimization

- **Service Catalog:** Pre-approved list of services for employees (Self-service).
 - **Trusted Advisor:** Checks 5 Pillars: **Cost, Performance, Security, Fault Tolerance, Service Limits**.
 - **License Manager:** Manages BYOL (Bring Your Own License).
 - **AWS Health Dashboard:** Alerts on service outages affecting *your* resources.
-

Module 10: Exam-Style Practice Questions

1. A company needs to provide a detailed report to an external auditor proving that they complied with PCI-DSS regulations over the past year. They need to collect evidence from AWS Config and CloudTrail automatically to minimize manual effort. Which service should they use?

- A) AWS Artifact
- B) AWS Audit Manager
- C) AWS Trusted Advisor
- D) AWS Control Tower

Correct Answer: B

Explanation: While AWS Artifact provides AWS's compliance reports, **AWS Audit Manager** is used to automate the collection of evidence for your audit and map it to compliance frameworks.

2. An administrator wants to ensure that no AWS account in their organization can disable AWS CloudTrail, including the root users of those accounts. Which feature of AWS Organizations can accomplish this?

- A) IAM Policy
- B) AWS Config Rule
- C) Service Control Policy (SCP)
- D) AWS Artifact Agreement

Correct Answer: C

Explanation: SCPs allow you to control the maximum permissions for member accounts and affect the **Root User**. IAM policies cannot restrict the Root user.

3. You have a requirement to receive an alert via text message (SMS) whenever the CPU utilization of your EC2 production fleet exceeds 80%. Which AWS service handles the metrics and the alarm triggering?

- A) AWS CloudTrail
- B) Amazon CloudWatch
- C) AWS Systems Manager
- D) AWS Trusted Advisor

Correct Answer: B

Explanation: Amazon **CloudWatch** collects metrics (like CPU utilization) and uses Alarms to trigger notifications when thresholds are breached.

4. A startup wants to optimize their AWS costs and security. They are looking for a tool that will automatically check their environment and recommend deleting unattached EBS volumes and enabling MFA on the root account. Which service provides these checks?

- A) AWS Trusted Advisor
- B) AWS Cost Explorer
- C) AWS Control Tower
- D) AWS License Manager

Correct Answer: A

Explanation: **AWS Trusted Advisor** evaluates the account against best practices in five categories, including Cost Optimization (unattached volumes) and Security (MFA).

5. Your company is expanding to a multi-account architecture. You need a service that will set up a "Landing Zone" with pre-configured security guardrails and a well-architected multi-account environment automatically. Which service should you choose?

- A) AWS Organizations
- B) AWS Service Catalog
- C) AWS Control Tower
- D) AWS Config

Correct Answer: C

Explanation: While Organizations manages the accounts, **AWS Control Tower** is the service that automates the setup of a "Landing Zone" and applies guardrails.

Module 11 - Pricing and Support

Here is the comprehensive summary, cheat sheet, and practice questions for **Module 11: Pricing and Support**, based strictly on the provided text.

1. AWS Pricing Fundamentals

There are three fundamental principles for how AWS charges:

1. **Pay-as-you-go:** No upfront costs or long-term contracts. Pay only for what you use (ideal for variable workloads).
2. **Save when you commit:** Discounts for committing to a specific usage level for **1 or 3 years** (e.g., Savings Plans, Reserved Instances).
3. **Pay less by using more:** Volume-based discounts. The more you use (economies of scale), the lower the per-unit cost becomes.

Main Drivers of Cost

- **Compute:** Charged by processing power and time (e.g., EC2 hours/seconds).
- **Storage:** Charged by amount stored and duration (e.g., GB/month in S3 or EBS).
- **Outbound Data Transfer:**
 - **Inbound** data transfer is usually **free**.
 - **Outbound** data transfer (moving data *out* of AWS) is **charged**.
 - Transfer between services within the same Region is usually free.

2. Billing and Cost Management Tools

AWS provides tools to help you track, forecast, and control spending.

AWS Billing Dashboard

- **Function:** The starting point. Shows current month-to-date spend and top services by cost.
- **Use Case:** Viewing your invoice and payment status.

AWS Budgets

- **Function:** Set custom budgets for cost, usage, or reservation coverage.

- **Key Feature:** Sends **Alerts** (email/SNS) when you exceed (or are forecasted to exceed) your defined threshold.
- **Use Case:** "Notify me if my spend goes over \$500."

AWS Cost Explorer

- **Function:** Visualizes and analyzes historical cost and usage over time.
- **Key Feature: Forecasting** future costs for the next 12 months based on past trends.
- **Tags:** You can use **Cost Allocation Tags** (metadata) to organize costs by project or department (e.g., "Project: Dev").

AWS Pricing Calculator

- **Function:** Web-based tool to estimate costs *before* you build.
 - **Use Case:** You want to know how much an architecture will cost before launching it.
-

3. Account Structure & Governance

AWS Organizations

- **Function:** Central management of multiple AWS accounts.
 - **Consolidated Billing:** Combines bills from all member accounts into the **Management Account** (Primary).
 - **Benefit:** You get one bill. You aggregate usage across accounts to reach **volume discount tiers** faster.
-

4. AWS Support Plans

All customers get **Basic Support** for free (access to documentation, Service Health Dashboard, and Trusted Advisor core checks). Paid plans offer more features:

Plan	Use Case	Response Time (Critical)	Key Features
Developer	Experimenting / Testing	< 12 hours (System Impaired)	Email access to support during business hours.
Business	Production Workloads	< 1 hour (System Down)	24/7 Phone/Chat/Email. Full Trusted Advisor checks. Infrastructure Event Management (for a fee).
Enterprise On-Ramp	Business Critical	< 30 mins (System Down)	Access to a pool of Technical Account Managers (TAMs).
Enterprise	Mission Critical	< 15 mins (System Down)	Designated Technical Account Manager (TAM) .

- **Technical Account Manager (TAM):** Your primary point of contact. Provides proactive guidance, architectural reviews, and cost optimization. (Enterprise plans only).
-

5. AWS Marketplace & Partners

- **AWS Marketplace:** A digital catalog to find, buy, and deploy third-party software (e.g., Security tools, OS images) that runs on AWS.
 - **Benefits:** Flexible pricing (pay-as-you-go), consolidated billing (software charges appear on your AWS bill).
 - **AWS Partner Network (APN):** Global community of technology and consulting partners who build solutions on AWS.
-

6. Cost Optimization Strategies

Compute

- **Rightsizing:** Using the smallest instance that meets your performance needs.
- **AWS Compute Optimizer:** Service that analyzes usage and recommends optimal instance types to save money.
- **Spot Instances:** Use spare capacity for **up to 90% discount** (fault-tolerant workloads).
- **Savings Plans:** Commit to usage (\$/hr) for 1 or 3 years for significant discounts.

Storage & Data

- **S3 Storage Classes:** Use **Intelligent-Tiering** or **Glacier** for infrequently accessed data.
 - **Lifecycle Policies:** Automate moving/deleting old data.
 - **Compression:** Compress data before storing to reduce size.
 - **VPC Endpoints:** Keep traffic within the AWS network to avoid internet data transfer charges.
-

Module 11: Cram Cheat Sheet

Pricing Principles

- **Pay-as-you-go:** No upfront cost.
- **Save when you commit:** 1 or 3 year contract (Savings Plans/RIs).
- **Pay less by using more:** Volume discounts.

Cost Tools

- **Budgets:** **Alerts** when spending exceeds limits.
- **Cost Explorer:** **Visualizes** history and **Forecasts** future spend.
- **Pricing Calculator:** **Estimates** costs *before* deployment.
- **Consolidated Billing:** One bill for multiple accounts. Aggregates usage for discounts.

Support Plans

- **Basic:** Free. No human support.

- **Developer:** Email support (biz hours).
- **Business:** **24/7 Phone/Chat.** < 1 hr response (Prod Down).
- **Enterprise:** TAM included. < 15 min response.

Optimization

- **Compute Optimizer:** Recommends rightsizing.
 - **Spot Instances:** Spare capacity (90% off).
 - **Tags:** Metadata used to categorize costs (e.g., by Department).
-

Module 11: Exam-Style Practice Questions

1. A company wants to receive an email notification whenever their monthly AWS spending exceeds a specific dollar amount. Which service should they use?

- A) AWS Cost Explorer
- B) AWS Budgets
- C) AWS Pricing Calculator
- D) AWS Trusted Advisor

Correct Answer: B

Explanation: AWS Budgets is specifically designed to set custom budgets and send alerts (notifications) when thresholds are breached.

2. Which AWS Support Plan is the minimum required to receive 24/7 access to customer service via phone, chat, and email?

- A) Basic Support
- B) Developer Support
- C) Business Support
- D) Enterprise Support

Correct Answer: C

Explanation: Developer support only offers email access during business hours. **Business Support** is the first tier to offer 24/7 access via phone, chat, and email.

3. A company has multiple AWS accounts and wants to receive a single bill for all of them. They also want to combine the usage across accounts to qualify for volume pricing discounts. Which feature should they use?

- A) AWS Cost Explorer
- B) AWS Organizations (Consolidated Billing)
- C) AWS Budgets
- D) AWS Control Tower

Correct Answer: B

Explanation: AWS Organizations enables Consolidated Billing, which combines bills and aggregates usage to achieve volume discounts.

4. You are planning to migrate a workload to AWS and want to estimate the monthly cost of the architecture before you provision any resources. Which tool should you use?

- A) AWS Cost Explorer
- B) AWS Budgets
- C) AWS Pricing Calculator
- D) AWS Billing Dashboard

Correct Answer: C

Explanation: The AWS Pricing Calculator is a web-based planning tool used to create cost estimates *before deployment*.

5. Which AWS Support Plan includes access to a designated Technical Account Manager (TAM) to provide proactive guidance and architectural reviews?

- A) Business Support
- B) Developer Support
- C) Enterprise Support
- D) Basic Support

Correct Answer: C

Explanation: A Technical Account Manager (TAM) is a premium feature included only with the **Enterprise** (and Enterprise On-Ramp) Support plans.

Module 12 - Migration to AWS Cloud

Here is the comprehensive summary, cheat sheet, and practice questions for **Module 12: Migrating to the AWS Cloud**, based strictly on the provided text.

1. The Migration Process

Migrating to AWS involves a three-phase process to ensure success.

- **Phase 1: Assess:** Evaluate readiness, identify business outcomes, and build the business case.
- **Phase 2: Mobilize:** Create a migration plan, address gaps, and understand application dependencies.
- **Phase 3: Migrate and Modernize:** Execute the migration, validate applications, and optimize.

2. AWS Cloud Adoption Framework (AWS CAF)

A framework providing guidance and best practices to accelerate migration and reduce risk. It organizes guidance into **six perspectives**, split between business and technical capabilities.

Business Capabilities

- **Business Perspective:** Ensures IT aligns with business needs. Creates the business case. (Roles: Finance, Budget Owners).
- **People Perspective:** Focuses on organizational change management, staffing, and training. (Roles: HR, People Managers).
- **Governance Perspective:** Focuses on risk management and aligning IT strategy with business strategy. (Roles: CIO, Program Managers).

Technical Capabilities

- **Platform Perspective:** Decides *how* to build/migrate solutions (architecture/patterns). (Roles: CTO, Solution Architects).
 - **Security Perspective:** Ensures visibility, auditability, and control. (Roles: CISO, Security Analysts).
 - **Operations Perspective:** Defines how to run, use, and recover IT workloads daily. (Roles: IT Operations Managers).
-

3. Seven Migration Strategies (The 7 Rs)

Every application falls into one of seven options when moving to the cloud.

1. **Rehost (Lift and Shift):** Move applications "as-is" (e.g., Physical to Virtual Machine). Fast, low effort, immediate cost savings (up to 30%).
 2. **Relocate:** Move containers or VMs (e.g., VMware) to the cloud without changing the hosting environment.
 3. **Replatform (Lift, Tinker, and Shift):** Move with minor optimizations but **no core code changes** (e.g., moving a self-managed DB to Amazon RDS).
 4. **Refactor (Rearchitect):** Writing **new code** to add features or performance (Cloud-native). Highest initial cost/effort, but highest long-term benefit.
 5. **Repurchase (Drop and Shop):** Switch to a different product, usually **SaaS** (Software-as-a-Service). (e.g., dropping a legacy license for a new SaaS subscription).
 6. **Retain:** Keep the application on-premises for now (do not migrate yet).
 7. **Retire:** Decommission applications that are no longer needed (turn off the lights).
-

4. Migration Tools

Planning and Assessment

- **AWS Application Discovery Service:** The "Detective." Automates discovery of on-premises inventory (servers/DBs) and maps dependencies/connections.
- **Migration Evaluator:** The "Cost Consultant." Builds a data-driven **business case** with projected cost estimates.
- **AWS Migration Hub:** The "Command Center." Central place to track progress of migrations across different tools.

Execution

- **AWS Application Migration Service:** The "Movers." Automates the "Lift and Shift" (Rehost) process. Minimizes downtime.
-

5. Database Migration

Tools specifically designed for moving data and databases.

AWS Database Migration Service (AWS DMS)

- **Function:** Migrates databases to AWS quickly and securely.
- **Key Feature:** The source database remains **operational (live)** during migration.
- **Use Case:** Homogeneous (Same engine) or Heterogeneous (Different engine) migrations.

AWS Schema Conversion Tool (AWS SCT)

- **Function:** Converts the **Schema** (structure) and code objects when changing database engines (Heterogeneous).
 - **Use Case:** Converting Oracle schema to Amazon Aurora before using DMS to move the data.
-

6. Data Transfer Services

Moving files and data storage to the cloud.

Online Transfer

- **AWS DataSync:** Automates/accelerates moving large data between on-premises storage and AWS (S3/EFS). Handles throttling and scheduling.
- **AWS Transfer Family:** Fully managed support for file transfers using **FTP, SFTP, and FTPS** directly into S3 or EFS.
- **AWS Direct Connect:** Dedicated, private fiber connection (bypasses internet).

Offline Transfer

- **AWS Snowball Edge (Storage Optimized):**
 - **Type:** Physical rugged device shipped to you.
 - **Use Case:** Offline migration of **petabyte-scale** data where internet bandwidth is limited or unavailable.
 - **Features:** High-performance NVMe storage and local computing capabilities.
-

Module 12: Cram Cheat Sheet

CAF Perspectives

- **Business:** Strategy, Finance, ROI.
- **People:** HR, Training, Culture.
- **Governance:** Risk, Policy, CIO.
- **Platform:** Architecture, CTO.
- **Security:** Compliance, CISO.
- **Operations:** Daily health, Support.

The 7 Rs

- **Rehost:** Lift & Shift (No changes).
- **Replatform:** Lift, Tinker, Shift (Optimization, e.g., EC2 → RDS).
- **Refactor:** Rearchitect (New code).
- **Repurchase:** Switch to SaaS.

- **Retire:** Delete/Turn off.

Migration Services

- **Migration Hub:** Central tracking dashboard.
- **Application Discovery Service:** Maps dependencies/inventory.
- **Migration Evaluator:** Builds the **Business Case** (Cost).
- **Application Migration Service:** Tool for **Rehosting**.

Data & DB Tools

- **AWS DMS:** Moves DB data. **Source stays live.**
 - **AWS SCT:** Converts **Schema** for different engines (Oracle → Aurora).
 - **AWS DataSync:** Automated online data transfer.
 - **AWS Transfer Family:** SFTP/FTP to S3.
 - **Snowball Edge:** Offline physical device. No internet needed.
-

Module 12: Exam-Style Practice Questions

1. A company wants to move a legacy application to AWS. They decide to move the application to Amazon EC2 instances without making any changes to the application code or configuration. Which migration strategy is this?

- A) Refactor
- B) Rehost
- C) Replatform
- D) Repurchase

Correct Answer: B

Explanation: Rehost (Lift and Shift) involves moving applications "as-is" without changes.

2. Which AWS Cloud Adoption Framework (CAF) perspective focuses on ensuring that IT strategies align with business strategies and involves the CIO and Program Managers?

- A) Business Perspective
- B) Governance Perspective
- C) Platform Perspective
- D) Operations Perspective

Correct Answer: B

Explanation: The Governance perspective focuses on skills and processes to align IT strategy with business strategy and manage risk. The Business perspective focuses on the business case and finance.

3. You need to migrate a 500 TB database from an on-premises Oracle server to Amazon Aurora. The schema structures are different. Which combination of tools should you use?

- A) AWS DataSync and AWS Transfer Family
- B) AWS Application Discovery Service and Migration Hub
- C) AWS Schema Conversion Tool (SCT) and AWS Database Migration Service (DMS)
- D) AWS Snowball Edge and AWS Config

Correct Answer: C

Explanation: AWS SCT is required to convert the schema (heterogeneous migration), and AWS DMS is used to migrate the actual data.

4. A company needs to transfer 500 Petabytes of data to the AWS Cloud. Their current internet connection is slow and unreliable. Which service is the most practical solution for this migration?

- A) AWS Direct Connect
- B) AWS DataSync
- C) AWS Snowball Edge Storage Optimized
- D) AWS Transfer Family

Correct Answer: C

Explanation: For offline migration where bandwidth is limited and data volumes are massive (petabytes), Snowball Edge devices are the correct solution.

5. Which service allows an organization to automate the discovery of on-premises server inventory and application dependencies to create a detailed migration plan?

- A) AWS Application Discovery Service
- B) AWS Application Migration Service
- C) AWS Migration Hub
- D) AWS Systems Manager

Correct Answer: A

Explanation: The Application Discovery Service is the "detective" that gathers configuration, performance, and connection details (dependencies) from on-premises servers.

Module 13 - Well-Architected Solutions

Here is the comprehensive summary, cheat sheet, and practice questions for **Module 13: Well-Architected Solutions**, based strictly on the provided text.

1. AWS Specialized Services

AWS offers services purpose-built for specific use cases, ranging from development to end-user computing.

Development Services

Tools to automate pipelines, debug applications, and build APIs.

- **CI/CD (Continuous Integration/Continuous Delivery):** Automates the release of code.
 - **AWS CodePipeline:** The **orchestrator**. Fully managed service that automates the build, test, and deploy phases of a release process.
 - **AWS CodeBuild:** The **executor**. Compiles source code, runs tests, and produces software packages. Scales automatically; pay only for build time.
- **AWS X-Ray:** A tracing and **debugging** tool. Helps visualizes application behavior to pinpoint performance bottlenecks and troubleshoot issues (especially in distributed/microservice architectures).

- **AWS AppSync:** Fully managed **GraphQL** service. Allows developers to create a single API to securely access and combine data from multiple sources (connects frontend to backend).
- **AWS Amplify:** Streamlines building, deploying, and managing **full-stack** (frontend and backend) web and mobile applications. Handles auth, storage, and hosting complexity.

Business Application Services

- **Amazon Connect:** AI-powered, cloud-based **contact center** service. Handles call routing, chats, and analytics.
- **Amazon Simple Email Service (SES):** Cost-effective, scalable **email** service for high-volume transactional (e.g., password resets) or marketing emails.

End-User Computing (EUC)

Services providing remote access to desktops and applications.

- **Amazon AppStream 2.0:** Streams specific **desktop applications** to a browser on any device. (SaaS-like experience for desktop apps).
- **Amazon WorkSpaces:** Fully managed **Virtual Desktop Infrastructure (VDI)**. Users get a full cloud-based desktop environment accessible from anywhere.
- **Amazon WorkSpaces Secure Browser:** (Formerly WorkSpaces Web). A managed **remote browser** for employees who only need access to internal websites and SaaS apps, without needing a full desktop or VPN.

IoT (Internet of Things)

- **AWS IoT Core:** Managed cloud service to securely **connect physical devices** (sensors, smart cameras) to the cloud. Handles data ingestion and processing.
-

2. The AWS Well-Architected Framework

A framework providing a consistent approach to evaluating architectures against best practices. It consists of **Six Pillars**.

1. Operational Excellence:

- **Focus:** Running and monitoring systems to deliver business value and continuously improving processes.
- **Keywords:** Automating deployments, responding to events, code pipelines.

2. Security:

- **Focus:** Protecting information, systems, and assets.
- **Keywords:** Least privilege, data integrity, locking down access, encryption.

3. Reliability:

- **Focus:** Recovery planning and ability to withstand/recover from failures.
- **Keywords:** Recovery from disruptions, adapting to demand, Multi-AZ.

4. Performance Efficiency:

- **Focus:** Using computing resources efficiently.

- **Keywords:** Rightsizing instances, selecting the right resource type for the workload.

5. Cost Optimization:

- **Focus:** Controlling and reducing expenses.
- **Keywords:** Eliminating waste, de-provisioning unused resources, using Spot Instances.

6. Sustainability:

- **Focus:** Minimizing environmental impact.
- **Keywords:** Energy-efficient design, reducing carbon emissions, using shared/serverless resources (Lambda) instead of always-on hardware.

AWS Well-Architected Tool:

- A free service in the console.
 - You define your workload and answer questions based on the 6 pillars.
 - It generates a report with potential issues and remediation steps based on best practices.
-

3. Architecture Optimization Example

The text uses an online florist to demonstrate how to apply the pillars to a standard architecture (EC2, RDS, S3):

- **Operational Excellence:** Implement **Infrastructure as Code** and auto-rollback.
 - **Security:** Enable **Encryption** and least-privilege IAM policies.
 - **Reliability:** Use **Auto Scaling** and deploy across multiple **Availability Zones**.
 - **Performance:** Use **Compute Optimizer** to ensure instances are rightsized.
 - **Cost:** Switch to **Spot Instances** for variable traffic or **Savings Plans** for steady traffic. Use **Cost Explorer**.
 - **Sustainability:** Optimize workloads to reduce waste (e.g., use serverless).
-

4. Serverless Architecture Patterns

Examples of how services connect in real-world scenarios:

- **Serverless Web Backend:**
 - **Flow:** Client → **Amazon API Gateway** (Validation) → **AWS Lambda** (Compute) → **Amazon DynamoDB** (Storage).
 - **Tracing:** **AWS X-Ray** is used to trace requests through this distributed map.
- **Serverless Contact Form:**
 - **Flow:** Static site on **S3** → **API Gateway** → **AWS Lambda** → **Amazon SES** (sends email).
- **Intelligent Contact Center:**
 - **Flow:** **Amazon Connect** (Phone/Chat) → **AWS Lambda** (Logic) → **Amazon CloudFront** (Content Delivery). Provides options to switch from voice to chat/email if wait times are long.

Module 13: Cram Cheat Sheet

Development Tools

- **CodePipeline:** Orchestrates the release workflow (Build → Test → Deploy).
- **CodeBuild:** Compiles code and runs tests.
- **X-Ray:** Debugging and performance tracing for distributed apps.
- **AppSync:** GraphQL API creator.
- **Amplify:** Tools for Full-stack web/mobile apps (Front-end + Back-end).

End-User & Business

- **WorkSpaces:** Full Virtual Desktop.
- **AppStream 2.0:** Streams specific Applications to a browser.
- **Connect:** Cloud Contact Center.
- **SES:** Bulk/Transactional Email.

The 6 Pillars

1. **Operational Excellence:** Continuous improvement, automation.
 2. **Security:** Risk mitigation, encryption, least privilege.
 3. **Reliability:** Recovering from failure, high availability.
 4. **Performance Efficiency:** Right-sizing, efficient resource use.
 5. **Cost Optimization:** Removing waste, analyzing spend.
 6. **Sustainability:** Environmental impact, energy efficiency.
-

Module 13: Exam-Style Practice Questions

1. A developer needs to debug a distributed serverless application composed of API Gateway, Lambda, and DynamoDB. The developer needs to trace user requests as they travel through these services to identify latency issues. Which service should be used?

- A) AWS CloudTrail
- B) AWS X-Ray
- C) AWS CodeBuild
- D) Amazon Inspector

Correct Answer: B

Explanation: AWS X-Ray is a tracing and debugging tool specifically designed to visualize application behavior and identify bottlenecks in distributed systems.

2. A company wants to provide its remote employees with a secure, fully managed cloud-based virtual desktop experience that they can access from any device. Which service is the best fit?

- A) Amazon AppStream 2.0
- B) Amazon Connect
- C) Amazon WorkSpaces
- D) AWS Amplify

Correct Answer: C

Explanation: Amazon WorkSpaces is a fully managed virtual desktop infrastructure (VDI) service. AppStream streams *applications*, whereas WorkSpaces provides a full *desktop*.

3. During an architecture review, a solutions architect recommends replacing always-on EC2 instances with AWS Lambda functions to reduce energy consumption and the organization's carbon footprint. Which pillar of the Well-Architected Framework is this recommendation addressing?

- A) Cost Optimization
- B) Performance Efficiency
- C) Reliability
- D) Sustainability

Correct Answer: D

Explanation: The Sustainability pillar focuses on minimizing environmental impact and designing energy-efficient systems, such as using serverless resources instead of always-on hardware.

4. A startup wants to build a mobile application that requires authentication, data storage, and backend APIs. They want a service that streamlines the development and deployment of these full-stack features with minimal infrastructure management. Which service should they use?

- A) AWS CodePipeline
- B) AWS Amplify
- C) AWS IoT Core
- D) AWS X-Ray

Correct Answer: B

Explanation: AWS Amplify makes it convenient to develop, deploy, and manage full-stack (frontend and backend) applications, handling the complexity of auth, storage, and hosting.

5. Which AWS service helps you evaluate your existing workloads against the six pillars of the AWS Well-Architected Framework and provides a report with remediation steps?

- A) AWS Trusted Advisor
- B) AWS Cost Explorer
- C) AWS Well-Architected Tool
- D) AWS Config

Correct Answer: C

Explanation: The AWS Well-Architected Tool is a free service in the console specifically designed to assess workloads against the 6 pillars and provide improvement plans.

A MediVault EU - A realworld scenario



medivault_architecture

This is a comprehensive, real-world migration strategy for "MediVault EU," a fictional HealthTech startup based in Germany.

The Profile:

- **Business:** A SaaS platform allowing doctors and patients to upload, view, and analyze sensitive medical records and high-resolution imaging.
 - **Requirements:** HIPAA Compliance (for US clients), GDPR Compliance (for EU clients), High Availability (99.99%), Secure Payments, Disaster Recovery.
 - **Constraints:** Data residency (EU data stays in EU), Strict Audit Logs.
-

Phase 1: The Foundation – Global Infrastructure & Network Security

1. Requirement: Data Residency and High Availability.

We need to ensure European customer data never leaves the EU (GDPR) while ensuring the application survives the loss of a physical data center.

- **AWS Solution: Multi-Region Strategy with Multi-AZ.**
 - **Primary Region:** `eu-central-1` (Frankfurt). Germany has strict data protection laws, building trust.
 - **Secondary Region (DR):** `eu-west-1` (Ireland). For Disaster Recovery backups only.
- **Why this is best:** AWS Regions are physical locations. By selecting Frankfurt, we meet data residency laws. By utilizing **3 Availability Zones (AZs)** within Frankfurt, we ensure that if one data center floods or loses power, the other two take over instantly.

2. Requirement: Network Isolation.

The database containing patient data must never be accessible directly from the public internet.

- **AWS Solution: Amazon VPC with Public/Private Subnet Tiering.**
 - **Action:** Create a **Virtual Private Cloud (VPC)**.
 - **Public Subnets:** Only for Load Balancers and NAT Gateways.
 - **Private Subnets (App Layer):** For the application servers (API).
 - **Private Subnets (Data Layer):** For Databases and Caching.
 - **Why this is best:** This creates a layered defense. Even if a hacker penetrates the outer layer, the data is locked deep inside a network that has no route to the open internet.
-

Phase 2: Identity & Access Management (The Human Firewall)

3. Requirement: Strict control over who touches the data.

In a HIPAA environment, every access event must be attributable to a specific human. No shared "Admin" passwords.

- **AWS Solution: AWS IAM Identity Center (Successor to SSO).**
 - **Action:** Enable **AWS Organizations** to manage the Billing and Prod/Dev accounts. Enable **IAM Identity Center** to federate with the startup's existing corporate email (e.g., Google Workspace or Active Directory).
 - **Why this is best:** We enforce **MFA (Multi-Factor Authentication)** at the identity provider level. Developers log in using their email. We use **IAM Roles** (temporary credentials) rather than long-term IAM User Access Keys, eliminating the risk of hard-coded credentials being leaked on GitHub.
-

Phase 3: Compute & Application Hosting (Modernization)

4. Requirement: Scalability and Patching Compliance.

The startup cannot afford downtime to patch servers manually. If traffic spikes (e.g., flu season), the app must handle it automatically.

- **AWS Solution: Amazon ECS (Elastic Container Service) with AWS Fargate.**
 - **Action:** Containerize the application using Docker. Deploy it on **AWS Fargate**.
 - **Why this is best:** Fargate is "Serverless Compute for Containers." AWS manages the underlying OS and security patching. The startup only manages the application code. It scales automatically based on CPU/Memory usage. This removes the "Patching OS" burden from the Shared Responsibility Model.

5. Requirement: Traffic Distribution and Encryption in Transit.

Users connect via the web. We need to decrypt their traffic securely and distribute it to the containers.

- **AWS Solution: Application Load Balancer (ALB).**
 - **Action:** Place the ALB in the Public Subnets. Attach **ACM (AWS Certificate Manager)** SSL/TLS certificates to the ALB.
 - **Why this is best:** The ALB handles the SSL termination (offloading encryption work from the app). It performs health checks; if a container fails, the ALB stops sending traffic to it and directs it to a healthy one.
-

Phase 4: Data Storage & Caching (The Crown Jewels)

6. Requirement: Relational Data (Patient Records) with HIPAA Compliance.

We need a database that supports complex queries, transaction integrity (ACID), and encryption.

- **AWS Solution: Amazon Aurora PostgreSQL (Serverless v2).**
 - **Action:** Deploy Aurora in the Private Data Subnet. Enable **Encryption at Rest** using **AWS KMS**.

- **Why this is best:** Aurora is 3x faster than standard PostgreSQL. It replicates data 6 times across 3 Availability Zones. If a drive fails, it heals itself. "Serverless v2" scales the database capacity up/down instantly based on demand, saving money during low-traffic nights.

7. Requirement: Unstructured Data (X-Rays, MRIs, PDFs).

We need cheap, unlimited storage for large files that must be encrypted and immutable (to prevent tampering).

- **AWS Solution: Amazon S3 (Simple Storage Service).**
- **Action:** Create Private Buckets. Enable **Server-Side Encryption (SSE-KMS)**. Enable **S3 Object Lock** (WORM - Write Once Read Many) for medical records to ensure they cannot be deleted or altered for a set retention period (e.g., 7 years).
- **Why this is best:** S3 offers 11 9s of durability. It integrates with Macie (security) and is infinitely scalable.

8. Requirement: High Performance/Low Latency.

Doctors need patient profiles to load instantly.

- **AWS Solution: Amazon ElastiCache (Redis).**
 - **Action:** Store user sessions and frequently accessed non-critical data (like hospital lists) in memory.
 - **Why this is best:** Reduces load on the database (Aurora) and delivers data in microseconds.
-

Phase 5: Security & Edge Protection

9. Requirement: DDoS Protection and Web Exploits.

We must protect patient login pages from brute force attacks and SQL injection.

- **AWS Solution: Amazon CloudFront + AWS WAF (Web Application Firewall).**
- **Action:** Serve the application through CloudFront (CDN) to cache static assets (logos, CSS) at the edge. Attach **AWS WAF** to CloudFront.
- **WAF Rules:** Block traffic from non-serviced countries (Geo-blocking). Rate-limit login attempts (e.g., max 5 requests per minute from one IP). Block SQL injection patterns.
- **Why this is best:** CloudFront absorbs volumetric DDoS attacks (AWS Shield Standard is included). WAF blocks application-layer attacks before they reach the ALB.

10. Requirement: Secret Management.

Database passwords and API keys must not be written in the application code.

- **AWS Solution: AWS Secrets Manager.**
- **Action:** Store DB credentials in Secrets Manager.
- **Why this is best:** The application retrieves the password programmatically at runtime. Secrets Manager **automatically rotates** the password every 30 days, a key requirement for high-security compliance.

11. Requirement: Data Privacy Scanning.

We need to ensure no PII (Personally Identifiable Information) is accidentally uploaded to a public or wrong bucket.

- **AWS Solution: Amazon Macie.**

- **Action:** Enable Macie on all S3 buckets.
 - **Why this is best:** Macie uses Machine Learning to scan objects. If it detects a credit card number or Patient ID in a bucket that is not flagged as "Confidential," it alerts the security team.
-

Phase 6: Monitoring, Auditing & Governance

12. Requirement: Full Auditability (HIPAA).

We need to know exactly who accessed a patient record and when.

- **AWS Solution: AWS CloudTrail.**
 - **Action:** Enable CloudTrail across all regions. Create a "Trail" that logs data events (S3 object access) and management events.
- **Why this is best:** This is the "Black Box" flight recorder. It provides the legal proof of access required by auditors.

13. Requirement: Real-time Performance Monitoring.

- **AWS Solution: Amazon CloudWatch + X-Ray.**

- **Action:** Use CloudWatch for metrics (CPU, Latency). Use **AWS X-Ray** to trace a request from the user → ALB → Fargate → Database to find bottlenecks.
- **Why this is best:** Provides "Observability." We can set CloudWatch Alarms to page the on-call engineer (via SNS) if error rates spike.

14. Requirement: Compliance Automation.

We need to prove we are compliant continuously, not just once a year.

- **AWS Solution: AWS Config + AWS Audit Manager.**
 - **Action:** Use AWS Config rules (e.g., "ensure all EBS volumes are encrypted"). If a developer creates an unencrypted volume, Config flags it or auto-remediates it. Use Audit Manager to map these checks to the HIPAA framework.
 - **Why this is best:** It automates the collection of evidence for auditors.
-

Phase 7: Backup & Disaster Recovery

15. Requirement: Business Continuity.

What if the Frankfurt region goes down completely?

- **AWS Solution: AWS Backup (Cross-Region).**
 - **Action:** Configure AWS Backup to take daily snapshots of Aurora and S3. Replicate these backups to the **Ireland (eu-west-1)** region.
 - **Why this is best:** It provides a centralized console for all backups. Copying to Ireland ensures data stays in the EU but is safe from a disaster affecting Germany.
-

Review & Optimization

Is this Cost Optimized?

- **Yes:** We used **Aurora Serverless v2** (scales to zero/low when not used). We used **Fargate** (no paying for idle EC2 capacity). We will use **S3 Intelligent-Tiering** to automatically move old X-rays to cheaper storage classes (Archive) after 90 days. We will purchase **Compute Savings Plans** for a 1-year commitment to reduce Fargate costs by ~30%.

Is this Secure?

- **Yes:** Data encrypted at rest (KMS) and in transit (TLS/ACM). Private Subnets for all compute/data. WAF at the edge. Macie scanning for leaks.

Is this Fault Tolerant?

- **Yes:** Multi-AZ architecture for Database and Compute. If an AZ dies, the system auto-heals.
-

Service List by Category

1. **Compute:** Amazon ECS (Fargate).
 2. **Networking:** Amazon VPC, Application Load Balancer (ALB), Amazon CloudFront, Amazon Route 53 (DNS), AWS PrivateLink (for internal service connection).
 3. **Storage:** Amazon S3 (Files), Amazon Aurora PostgreSQL (Relational DB), Amazon ElastiCache (Redis).
 4. **Security & Identity:** AWS IAM Identity Center, AWS KMS (Encryption), AWS WAF (Firewall), AWS Shield (DDoS), AWS Secrets Manager, Amazon Macie, AWS Certificate Manager (ACM).
 5. **Management & Governance:** AWS Organizations, AWS CloudTrail (Audit), Amazon CloudWatch (Metrics/Logs), AWS Config (Compliance), AWS Audit Manager, AWS Trusted Advisor.
 6. **Developer Tools:** AWS CodePipeline, AWS CodeBuild (for CI/CD).
 7. **Recovery:** AWS Backup.
-

Architecture Diagram

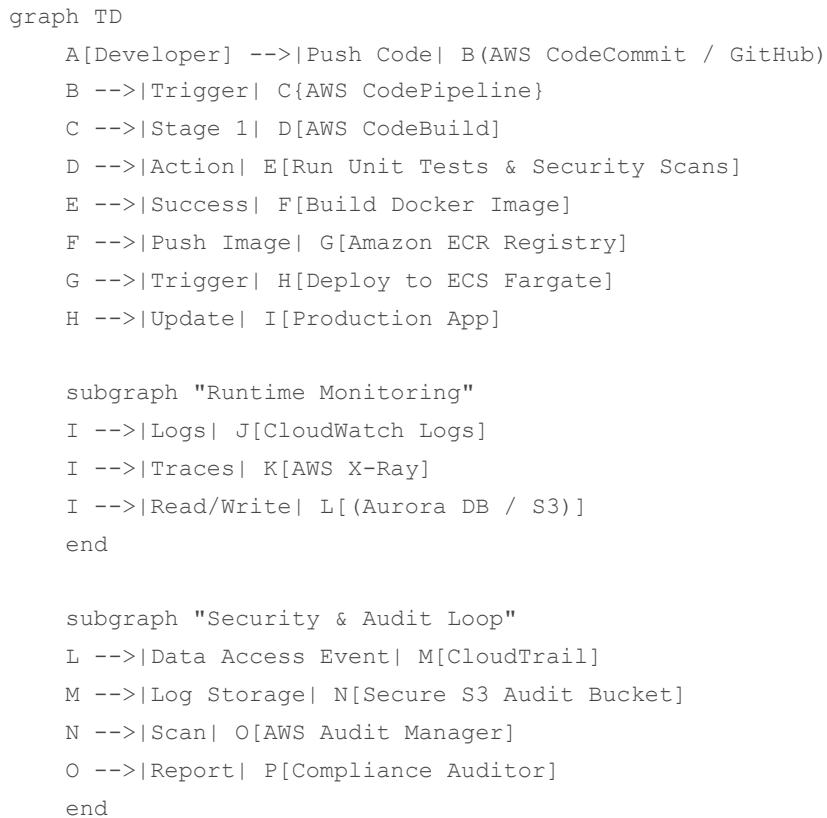
If the image above doesn't load, visualize this flow:

1. **Edge:** User → Route 53 → CloudFront (WAF/Shield)
 2. **Public Subnet:** → Application Load Balancer (ACM SSL)
 3. **Private App Subnet:** → ECS Fargate Cluster (across 3 AZs)
 4. **Private Data Subnet:** → Aurora PostgreSQL (Primary + Replica) & ElastiCache
 5. **Sidecar Services:** S3 (connected via VPC Endpoint), KMS, Secrets Manager.
 6. **Management Wrapper:** CloudTrail, Config, GuardDuty watching everything.
-

Workflow Diagram (Automated CI/CD & Audit)

This workflow ensures that every code change is scanned for security before it reaches production, and every action is logged.

Text



Module 1 - Introduction to the Cloud

▼ Introduction to AWS Cloud Practitioner Essentials

Rudy: Hey everyone! Welcome to the course. We hope you're ready to dive in and learn the fundamentals of the AWS cloud.

Alan: We have a lot of material to cover. And we're going to start with defining; what is the cloud?

Morgan: But, before we do that, let's start with some introductions. Hi, I'm Morgan Willis, a Principal Cloud Technologist with AWS Training and Certification. I started in the IT world about 15 years ago. And along the way, I decided that I was missing something.

I missed the helping and teaching aspect that I had in my first job in IT support. So, I went into teaching at a software development bootcamp. And then I eventually landed here at Amazon Web Services, or AWS, where, as a Cloud Technologist, I get to support others in their cloud learning journey every day.

Rudy: Howzit learners! My name is Rudy Chetty and I'm a Chief Techfluencer and a Principle Solutions Architect for AWS Partners. I'm originally from sunny Cape Town, South Africa...the home of bunny chow, biltong and boerewors. I've been in the technology space for over two decades helping customers worldwide realize their application and cloud dreams. Education is my passion. Therefore, I cannot wait for you to dive into this course and learn all about the wonders of the cloud and AWS.

Alan: And I'm Alan Meridian. I'm an instructor in AWS Training and Certification.

Over the last eight plus years with AWS, I've delivered hundreds of trainings introducing learners like you to AWS concepts, services, and solutions-sometimes involving ukuleles. I'm glad you're here. We're going to learn a lot together.

Rudy: Look, cloud computing can be complex. But don't worry. This is why we're here! Our curriculum provides analogies, examples, and use cases that will help you better understand these concepts. We'll cover all the essential information you need to be comfortable discussing AWS. Additionally, we'll throw in some AWS service demos to show you how things work in action. This layered approach to learning will really reinforce concepts. Oh, and we promise to keep it simple.

Morgan: AWS offers a massive range of services for every business. AWS is the world's most comprehensive and broadly adopted cloud, and millions of customers use AWS to be more agile, lower costs, and innovate faster. Whether you're looking for compute power, generative AI, databases, storage, content delivery, or specialized services for some other functionality, AWS has the services you need to help you build sophisticated applications for your business.

Alan: That is massive range of services, and I wish we had time to cover them all. But, we did promise to keep it simple. So, let's start this conversation with a fundamental concept of computing: the client-server model.

Rudy: And to get things percolating in your brain, we are going to brew up a coffee shop analogy. This coffee shop is going to act as a real-world metaphor to show you the different parts of the cloud computing model. And, the analogy will help you understand how AWS can change the way your IT operates.

First, let's think about the coffee shop and how it can represent the client-server model.

Alan: Let's have Morgan be the server, the barista. And I'll be the client, or the customer. I'm gonna make a request. In this case, it's for coffee. In the computing world, the request could be for anything. It could be rain pattern analysis in South Africa, or it could be the latest x-rays of your knee, or videos of kittens. Whatever the business needs is where the request starts. Basically, a customer makes a request, and with permissions, the server responds to that request. All I want is a caffeinated beverage. Morgan represents the server part of the client-server model. In AWS, she would be a virtual server. So, from a cloud computing architectural point of view, the coffee shop transaction is pretty straightforward. I, the user, made a request to Morgan, the server. Morgan validated that the request was legitimate. In this case, did I give her money? Is the item that I ordered something that they can make? Then she returned a response, which in this example, is a triple mocha with extra caramel shots.

Morgan: Now, in the real world, applications are usually more complicated than just a single transaction with a single server. In a business solution that is more mature, it can get beautifully complex. But again, we're going to start with the basics. As the curriculum progresses, these basic concepts will continue to build on each other. And hopefully, by the end, those beautifully complex concepts will make a lot more sense. You've already tackled your first cloud computing concept. So now, let's continue with a key concept of AWS, and that is, you only pay for what you use.

Rudy: Think about this principle as it applies to staffing a coffee shop. Employees are only paid for the hours they work in the actual shop. So, if say Morgan and I are off the clock, well, then we don't get paid because we didn't work any hours. Moreover, the store owner decides how many baristas they need for any given day of the week. Busy days require more employees and slower days require fewer. Let's take an example. Say the coffee shop is releasing a brand-new drink called Rudy's Rhubarb Refresher. Trademark. The shop anticipates increased demand due to the beverage being very tasty, so they staff the shop with 10 baristas all day long. Although this works with a sudden surge of customers, it isn't great for slower periods during the day. Some baristas could be idle, and it would be hard to justify paying them to just be there in case they are needed. The monetary cost just doesn't make good financial sense.

Alan: And yet, this is exactly what happens in an on-premises data center where you can't just snap your fingers and triple your capacity. At AWS, you don't pre-pay for anything. And you don't have to worry about capacity constraints.

Rudy: When you need more resources, you just provision them right then and there. How cool is that? Then, when you don't need those resources anymore, you can deprovision just as quickly. You just set up the right configuration and it's done automatically. And the best part is that when you deprovision the resources, you stop paying for them immediately. Just like you only pay your employees for the hours they're working, you only pay for the AWS resources that you consume.

Morgan: Right. So, paying only for what you use is the first AWS specific concept we covered—and it's just one of the many benefits when it comes to running your business on AWS. And that's really why we're here: so you can understand how AWS is built to help you run your business better.

Alan: Exactly. You've already learned some foundational concepts about the cloud...and that's just the start!

Rudy: Yep, things are only going to get more interesting, folks, as we dive deeper! Thanks for joining us on this learning journey, future AWS Cloud Practitioner. See you soon.

▼ What is Cloud Computing?

Before we dive into cloud computing, let's rewind the clock and set some context for how Amazon grew to include Amazon Web Services. In the early 2000s, Amazon.com was an ecommerce site that customers used to buy books and other consumer goods. As more people started to use the site, the Amazon IT team had to continually make upgrades to keep things running smoothly. More servers, more storage, more compute. You name it. They were deploying it!

The team eventually decided to develop various standardized tools, mechanisms, and ways to make things more efficient and scalable. These methods proved to be quite effective, and in 2003, employees thought, "Maybe this knowledge would be valuable to other companies facing similar challenges." Thus, Amazon started to envision a service that would allow businesses to rent computing power, storage, and other resources, on-demand. This business model could eliminate the need for upfront investment in hardware.

Just a year later, in November 2004, AWS launched its first public infrastructure service: Amazon **Simple Queue Service**. Two years later, AWS launched Amazon **Simple Storage Service** for data storage and Amazon **Elastic Compute Cloud** for scalable compute. Initially, AWS was used by smaller start-ups and developers. However, its scalability, cost-effectiveness, and ease of use quickly attracted larger enterprises.

Over the next few years, AWS rapidly expanded its offerings by adding **databases**, **networking**, **analytics**, and many other cloud-based services. Fast forward to the present. AWS powers a significant portion of the internet, serving millions of customers worldwide. From small start-ups and businesses, to large corporations, government agencies, and more. What started as an internal solution for Amazon's own IT needs has grown into a global cloud computing leader.

With that quick history lesson in mind, let's look at a working definition of cloud computing. Cloud computing is essentially the **on-demand delivery of IT resources** over the internet with **pay-as-you-go** pricing. Shall we break this down a bit? I think we shall. On demand means you use resources as needed. Let's say your business needs 2,000 TB of storage. Open an AWS account, throw some files into Amazon S3, and Bob's your uncle. You're good to go. If you don't need to store those files anymore? Delete them, and stop paying immediately.

The idea of the delivery of these on-demand IT resources, as you learned, is the concept that got AWS started in the first place. Let's say you have an application that you have to run, content you need stored, or data you need analyzed. Basically, you have stuff that uses IT resources that have to live and operate somewhere. All of that data is stored in a data center, which is essentially a building or set of buildings devoted to housing servers that contain all this data. Data centers are designed with redundant power, cooling, and security measures to ensure secure and continuous operation. Historically, businesses would run applications in their own data centers or co-locate with other customers in a shared facility. There was no alternative. Once AWS became available, companies could now run their applications in other data centers they didn't actually own. No more infrastructure to manage. No more repetitive tasks and time-consuming ones--goodbye. By using AWS, teams could now focus on innovation.

Oh, and the over-the-internet part means you can access those resources remotely. You can be in your house, place of business, or visiting family around the world. Hi Mom! In fact, all you need is an internet connection. Log into your AWS Account and manage your infrastructure right from your web-browser.

And with pay-as-you-go pricing, if you don't need a particular part of your infrastructure, just deprovision it. Simple as that. No contracts. No need to call a sales rep.

To reiterate: cloud computing is the on-demand delivery of IT resources over the internet with pay-as-you-go pricing. So, as you tackle the rest of this content, keep this definition in mind. Happy learning!

▼ Benefits of the Cloud

Okay. You have a good working definition of cloud computing, and you've also touched on why it's beneficial from a business perspective. Now, you'll learn even more about the advantages of using a cloud computing model. We'll cover six primary benefits of the AWS Cloud.

Let's start by talking a bit more about that first concept we've mentioned: the ability to pay as you go. With the cloud, you can trade fixed expenses for variable expenses. Remember when we talked about how data is housed in a data center? When you're building a traditional on-premises facility, you typically need to invest a large amount of money **upfront**. This includes investments in physical space, hardware, staff, and upkeep to make sure everything is running smoothly. For some businesses, this can be hundreds of thousands or even millions of dollars. Then, regardless of the data center's utilization rate, you have a **fixed cost**, and you have to set up your budget around that fixed cost.

Billing with AWS is fundamentally different. Your bill with AWS will be variable from month to month as you consume different amounts of resources. The great thing about this model is that if you're just getting started, there's no need to come up with a big investment to build out your AWS environment. Instead, you can start small and get **billed for only what you use**. And, you can use **built-in billing and budgeting tools** to find ways to save money every month as your business grows. So, that's the first advantage.

The next advantage is that you benefit from massive **economies of scale**. AWS is building data centers all around the world. And in order to build all those data centers, AWS is buying a whole lot of hardware. Because AWS buys a good amount of hardware, we can purchase this hardware at a lower price. Then, we pass on those lower costs to the customer.

The next benefit? **Stop guessing capacity**. In a traditional data center, you purchase hardware based on projected usage. Let's say you estimate you'll have a user base of 10 million people over the next three years. That means you purchase enough hardware to support that growth over time. But what happens if, when that third year hits, you only have about 500,000 users? You're still stuck with the hardware you purchased to support 10 million users. Let's flip the script. Imagine you underestimated your capacity. A growing customer base is great for your business, but the scramble to secure more hardware could be a real problem. You would need to rapidly procure more servers to handle that higher capacity. If you don't adjust in time, your customers could have a degraded experience, or you might even lose them altogether.

This is the awesome part of AWS. You don't need to guess capacity. **Scaling** usually takes minutes, instead of the weeks or months it can take in an on-premises data center. You provision needed resources for the present. Then, you use scaling mechanisms to **scale these resources up or down based on the day-to-day demand**.

The next benefit is my personal favorite. Increase speed and agility. With AWS, you have more bandwidth to try new things. You can spin up test environments and run experiments, trying out new ways to solve a problem. And then, if that approach didn't work, you can just delete those resources and stop incurring cost. That way, you can **spend less time provisioning and deprovisioning** and more time innovating and optimizing.

Alright, next up. Stop spending money running and maintaining data centers. We've talked about the upfront cost of data centers, but there's more to it than that. Apart from the fixed cost of what it takes to build out a data center, AWS eases the **cost of running and maintaining servers**. This means instead of spending time and resources focusing on racking, stacking, and powering those servers, you're spending more time focusing on customers.

And last but not least, we have **go global in minutes**. Let's say you're a company based in the US, and you want to expand your operations to India. Traditionally, you would need to run and operate data centers in India to effectively operate there. But with AWS, you don't need to do that on your own. You can instead deploy your applications to an AWS Region in India that is managed by AWS. The idea here is that the time it takes for you to expand into a secondary area of the world traditionally can be months or years. With AWS, it can just take minutes.

That's it. Those are the six main benefits of using the AWS Cloud. As you can tell, with AWS, your business can achieve cost savings, time savings, and the ability to tap into massive global infrastructure. As we continue to dive into the specifics of the infrastructure and services of AWS, it'll become more and more clear how your business can achieve these benefits.

- No fixed cost, No upfront money
- Economic of scale
-

▼ Introduction to the AWS Global Infrastructure

Let's talk about the AWS Global Infrastructure. You've already learned that AWS builds and maintains data centers around the world that you can use to go global in minutes, but there's a huge advantage to global reach that we haven't covered yet—and that's high availability.

To learn a little bit more about high availability, let's head back to our coffee shop. Say you've hired a new employee, and they're learning how to make a latte. They're doing awesome. They've got the right milk-to-espresso ratio, and they are even making some cool designs with their latte pour—until they miss the cup and they pour the latte all over the register. That is not good. The register is now fried, and it seems like it shorted the electricity everywhere in the shop. Yikes! That means we can't ring up the orders or make drinks for our customers. We're gonna have to close up shop until this is sorted out.

So, what does that mean for the business? Does this mean that the entire business isn't making any money until this is fixed? Well, luckily for us, we're prepared. The good news for our business—and for our customers—is that this isn't our coffee shop's only location. The shop is actually a chain, and we have locations all around the city. Customers can still get their coffee by visiting one of our shops just a few blocks away, and the business can continue even if one location is having a bad time.

So, whether an employee had a slight latte mishap, or a shop next door accidentally cut our internet line, or some other disaster keeps the shop from its day-to-day operations...no matter the reason, we know that our product will be highly available for our customers. They still get their coffee, and the business is still generating income. All is well.

AWS has a similar set up with our global infrastructure. It's risky to have one giant data center where all of the resources are housed. If something were to happen to that data center, like a power outage or a natural disaster, everyone's applications would go down all at once. You need high availability and fault tolerance. Let's clarify those terms. High availability is all about making sure your applications stay accessible with minimal downtime. Even if one component fails, another is ready to pick up the slack so your service keeps running.

Fault tolerance takes it a step further by designing a system to continue to operate even if multiple components fail. It's basically building resilience into every layer so that no one single failure brings down the whole system. Designing for high availability and fault tolerance is part of the reason why AWS operates in Regions, which are located in different areas around the world. These Regions are built to be as close to AWS customers as possible. This includes locations like Paris, Tokyo, Sao Paulo, Dublin, or Ohio.

Within each Region, we have what we call Availability Zones, or AZs. There are three or more AZs within a Region, for redundancy. We don't build AZs right next to each other, because if something like a natural disaster were to occur, you could lose connectivity to everything in that AZ. And continuing with the theme of redundancy, within each AZ, there is one or more discrete data centers with redundant power, networking, and connectivity.

So, if a Region is where all the pieces and parts of your application live, some of you might be thinking that we never actually solved the problem that we presented earlier. If my business needs to be disaster proof, then it can't run in just one location. Well, you're absolutely correct. That's why it's common for businesses to operate across multiple Regions. That way, if one Region is experiencing outages for any reason, the operations can failover to another Region...but we'll cover that in more depth in a later lesson.

In the meantime, I'm gonna sit back here and relax, because I know that my business is highly available regardless of any disasters...or spilled lattes.

▼ **The AWS Shared Responsibility Model - AWS Shared Responsibility Model - SRM**

When it comes to securing your business on AWS, it's important to ask the question: Who is ultimately responsible for the security? Is it A: You, the customer? or B: AWS? And the correct answer is: Yes. Both. Both are ultimately responsible for making sure that your workloads are secure.

This is what's known as the AWS Shared Responsibility Model. And it's similar to securing a house. The builder constructed the house with four walls and a door. It's their responsibility to make sure the walls are strong and the doors are solid. It's your responsibility as the homeowner to close and lock those doors. And it really is that straightforward on AWS as well.

One helpful way to think about it is that AWS is responsible for security of the cloud. The customer is responsible for security in the cloud. AWS is responsible for the physical, network, and hypervisor layers. The physical layer is just that: the physical realities of a computer. Access to the hardware has to be secured with locks on doors, access control lists, privilege separation, and a lot more. Similarly, the network layer and the hypervisor virtualization layer have their own protections to ensure isolation of the different AWS customer workloads.

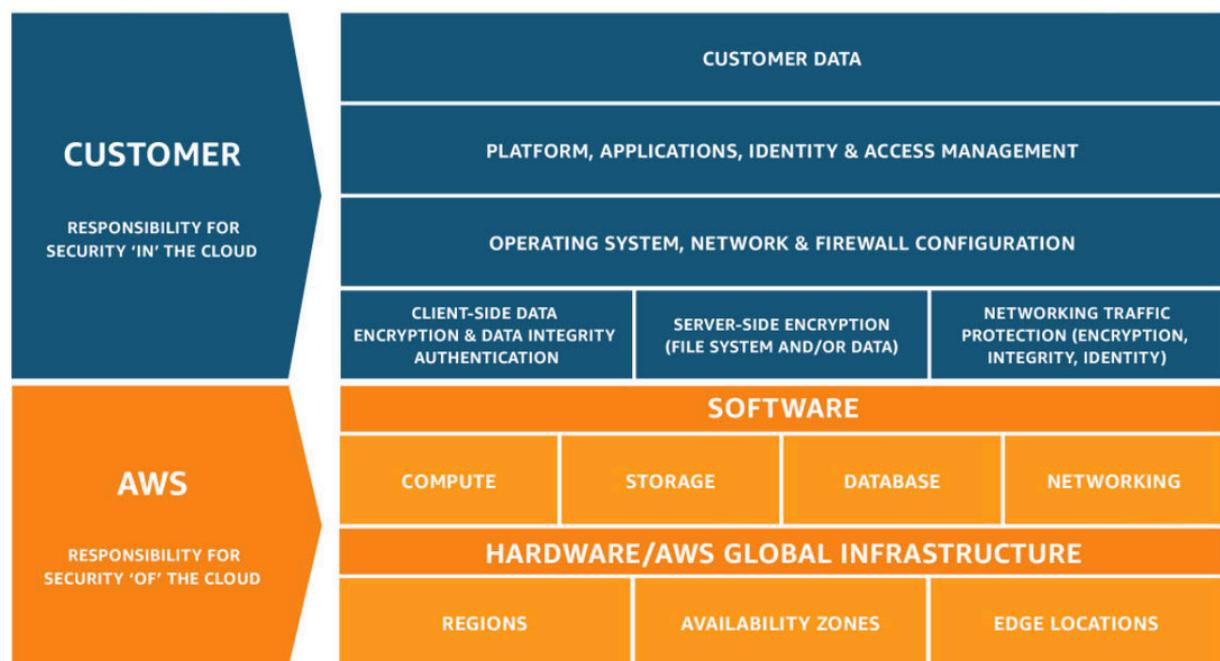
Now let's talk about customer responsibilities. You're 100 percent in charge of your operating system. AWS does not have a back door into your system here. You and you alone have the only encryption key to log in to this OS or to create any user accounts there. I mean, no more than a construction company would keep copies of your front door key, AWS cannot enter your operating system. That means your operations team is responsible for keeping the OS patched. If AWS discovers there are some new vulnerabilities in your version of the operating system, we can certainly notify your account owner, but we cannot deploy a patch. The construction company can install high quality locks to secure your home, but they aren't going to come lock the door for you every time you leave.

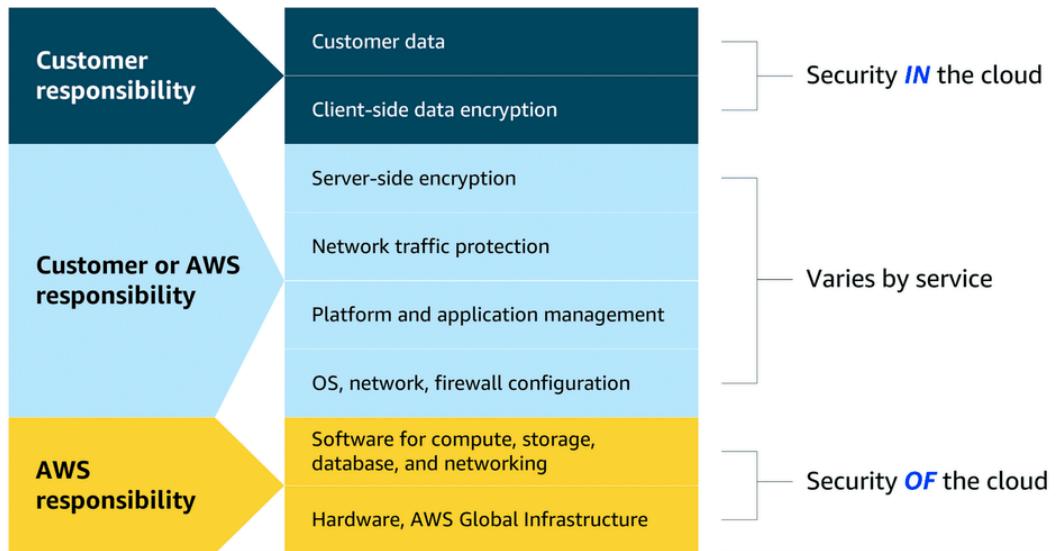
Same goes for the applications you're running: You own them and you secure them. Which takes us to the next part of the stack: data. This is your domain to control. And sometimes you might want to have your data open for everyone to see, like pictures on a retail website. Other times, like banking or healthcare, yeah, not so much.

AWS provides everyone with the controls to open it up to some authorized individuals, to everyone, to just a single person under specific conditions, or even to lock it down so no one can access it. Plus, you have the ability to encrypt your data. That way even if you accidentally left your front door open, all anyone would see is unreadable encrypted content.

The AWS Shared Responsibility Model is about making sure both sides understand exactly which tasks are ours and which tasks are the customer's. Depending on the service used, responsibility can shift, and we'll talk more about specifics as we come across different services.

Again, at a basic level, it's important to remember that AWS is responsible for the security of the cloud, and you are responsible for the security in the cloud. Together, you have an environment you can trust.





▼ Cloud in Real Life: Applying Cloud Concepts to Real Life Use Cases

Morgan: Hey everyone. Welcome to a segment of this training that we call Cloud in Real Life. Every once in a while, the three of us will come together, and take the concepts you've learned, and explore how they work in real-world, real-life scenarios.

Rudy: Exactly, so instead of just talking about features and definitions, we're going to think through how businesses or individuals might approach real-world problems.

Alan: And this is super important because this course is for anyone trying to learn about AWS, including beginners. So, we've purposefully simplified a lot of the AWS concepts to make them digestible for someone just getting started. With these Cloud in Real Life segments, we'll talk about how these basic concepts turn into real-world solutions for businesses.

Rudy: Oh, wait, shall we kick things off with an example? Yeah, yeah.

Morgan: Yeah, let's do it.

Rudy: OK. Let's take two concepts we've discussed in this section: AWS Global Infrastructure and the AWS Shared Responsibility Model. In the real world, these concepts certainly work together and aren't separate ideas.

Alan: So true. Let's use an example of something like...um...about an e-commerce company. One that wants to expand operations globally.

Morgan: Yeah, I like that example. There are tons of e-commerce companies that use AWS. In general, how would using AWS Global Infrastructure benefit a company like that?

Alan: Well, one thing to keep in mind is that the further your infrastructure is from your customers, the longer the latency will be for those customer requests. So, they could choose AWS Regions to host their application closer to their customer base. For example, if they have a significant customer base in Europe and Asia, they might choose to deploy their applications in Regions like eu-west-1 in Ireland and ap-southeast-1 in Singapore.

Morgan: It's so much easier to reach a global audience when you're leveraging existing resources like AWS Regions. The groundwork has already been laid. And even though all businesses benefit from this, I also can't help but think about startup companies, and how much you can achieve with a small team by leveraging the cloud. You don't need to have a large upfront capital investment to reach a global audience, which can really help to level the playing field a little bit for startups and small companies.

Rudy: Oh, that's so true. And also, it's a lot more straightforward to build resiliency into applications. For the e-commerce company, they could start by using at least two Availability Zones, or AZs. They deploy the same configuration in each, and if one fails, they can failover to the other. This is called designing for high availability and fault tolerance.

Morgan: Now we haven't talked about the importance of multiple Availability Zones too much yet, but we'll get there soon. Now, we also learned about the AWS Shared Responsibility Model. So, how does this come into play here?

Alan: Yeah, the shared responsibility model is super important for understanding security and compliance. Our e-commerce company can focus on the higher order issues of securing their data and managing access to their AWS resources. Plus, they'll be able to give attention to things like making sure their applications are configured securely to comply with regulations about credit card information rather than having to decide what locks to put on the doors at the datacenter.

Morgan: Right, there's the whole "in the cloud" and "of the cloud" situation. AWS is responsible for security of the cloud, and customers are responsible for security in the cloud. AWS services are used together like building blocks to form solutions. So this concept of the shared responsibility model will become clearer as we see how different services are used separately and together.

Rudy: Yeah, and there's so much to learn, and we will have more meaningful discussions as we go along. And as you learn new concepts, think about how they might apply to your own work or projects.

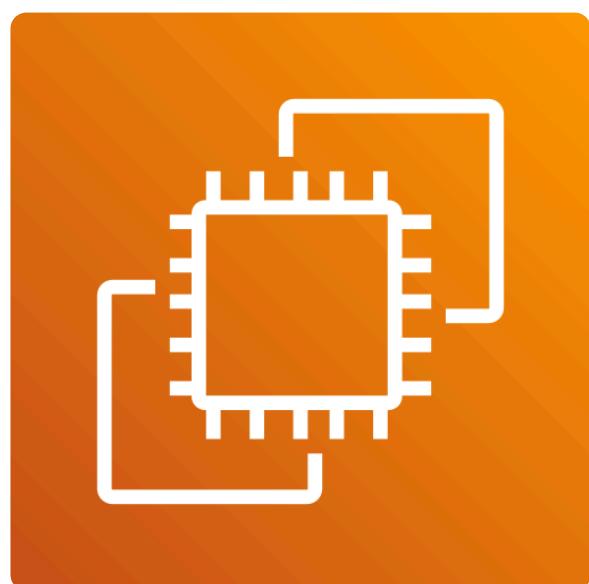
Module 2 - Compute in the Cloud EC2

If you remember from our coffee shop, the employees are a metaphor for the client/server model where a client sends a request to the server, the server does some work, and then sends a response.

▼ Introduction to Amazon EC2

In this video, we are going to talk at a high level about a service called Amazon **Elastic Compute Cloud**, or Amazon **EC2**.

That example is for the coffee shop, but the same idea applies to other businesses. These businesses, whether they are in healthcare, manufacturing, insurance, or delivering video content, are also using this model to deliver products, resources, or data to end users.



You need raw compute capacity to host your applications and provide the compute power that your business needs. When you're working with AWS, those **servers** are called **EC2 instances**. Using EC2 for compute is **highly flexible**, **cost effective**, and **quick** when you compare it to running your own servers on premises in a data center that you own. The **time** and **money** it takes to get up and running with **on-premises** resources is fairly **high**, but with EC2, it's much more convenient to get started.

AWS took care of the hard part for you, and AWS is constantly operating a massive amount of compute capacity ready to be used. And you can use whatever portion of that capacity when you need it. All you have to do is request the EC2 instances you want, and they will launch and boot up, ready to be used within a **few minutes**. After you're done, you can stop or terminate the EC2 instances. You're not locked in or stuck with servers that you don't need or want.

Your usage of EC2 instances can vary greatly over time. And **you only pay for what you use**. Because with EC2, you only pay for running instances, not stopped or terminated ones.

You **STOP PAYING for the compute time (CPU/RAM)**. You **continue to PAY for the attached EBS volumes** (the hard drive).

Stopped Instance Use Case

You stop an instance when you know you will need to use it again soon, or when you need to **change its underlying hardware (instance type)** without losing the **application data** on the attached hard drive.

EC2 instances are **virtual machines**, or VMs. VMs share an underlying physical host machine with multiple other instances, which is a concept called **multi-tenancy**. In a multi-tenant environment, you need to make sure that each VM is isolated from **each other but is still able to share resources provided by the host**.

This job of resource sharing and isolation is being done by a piece of software called a **#hypervisor**, which is running on the host machine. For EC2, AWS manages the underlying host, the hypervisor, and the isolation from instance to instance. So, even though you won't be managing this piece, it's important to have a basic grasp of the concept of multi-tenancy.

When you provision an EC2 instance, you can choose the operating system, or OS, based on either Windows or Linux. You can provision thousands of EC2 instances on demand, with a blend of operating systems and configurations to power your business' different applications.

You also can configure what software you want running on the instance. Whether it's your own internal business applications, simple or complex web apps, databases, or third-party software like enterprise software packages, you have complete control over what happens on that instance. You'll see us launch an EC2 instance in an upcoming demo.

EC2 instances are also **resizable**. You might start with a small instance and realize the application you are running is starting to max out that server. You can then give that instance **more memory and more CPU**. This is what we call **vertically scaling an instance**. In essence, you can **make instances bigger or smaller** whenever you need to.

You also control the **networking** aspect of EC2. So, you decide what type of requests make it to your server and if they are publicly or privately accessible. We'll talk more about this later in the networking section.

Now, it's important to note that the concept of VMs isn't a new thing. AWS has just made it much more convenient and more cost effective for you to acquire servers through this **Compute as a Service model**.

When launching an EC2 instance, you start by selecting an **Amazon Machine Image (AMI)**, which defines the **operating system** and might include **additional software**. You also choose an instance type, which determines the underlying hardware resources, such as CPU, memory, and network performance.

You need to choose EC2 instance type (**how powerful** you want it) and the Amazon Machine Image (AMI), which determines the operating system and software for your instance.

You can connect to an EC2 instance in various ways. Applications can interact with services running on the instance over the network.

Users or administrators can connect using SSH for Linux instances or Remote Desktop Protocol (RDP) for Windows instances. Alternatively, AWS services like **AWS Systems Manager** offer a secure and simplified method for **accessing instances**.

After you are connected to the instance, you can begin using it to run commands, install software, add storage, organize files, and perform other tasks.

▼ A little more about Amazon EC2

Now that we've learned about EC2, let's peel the onion back and discuss types of instances.

Visualize EC2 instances as different types of coffee machines in our coffee shop. Customers like variety. Not all of them drink the same thing, so we need more than just one type of coffee machine.

For example, we'll need a high-powered espresso machine for those who want espresso shots or lattes. We'll also need a classic-drip coffee machine for customers who just want the sauce, and maybe a cold-brew machine for those who want their caffeine chilled. Therefore, if we want our business to operate as efficiently as possible, it's important to use the right coffee machine for each order.

This is similar to deploying EC2 instances. There are **different types of instances** that serve **different purposes** in your AWS environment. **Instance types are grouped under instance families**, which offer **varying combinations** of CPU, memory, storage, and networking capacity. This gives you the flexibility to **optimize** for certain types of tasks by choosing instances specific to your **particular workload**.

The different instance families are **general purpose**, **compute optimized**, **memory optimized**, **accelerated computing**, and **storage optimized**.

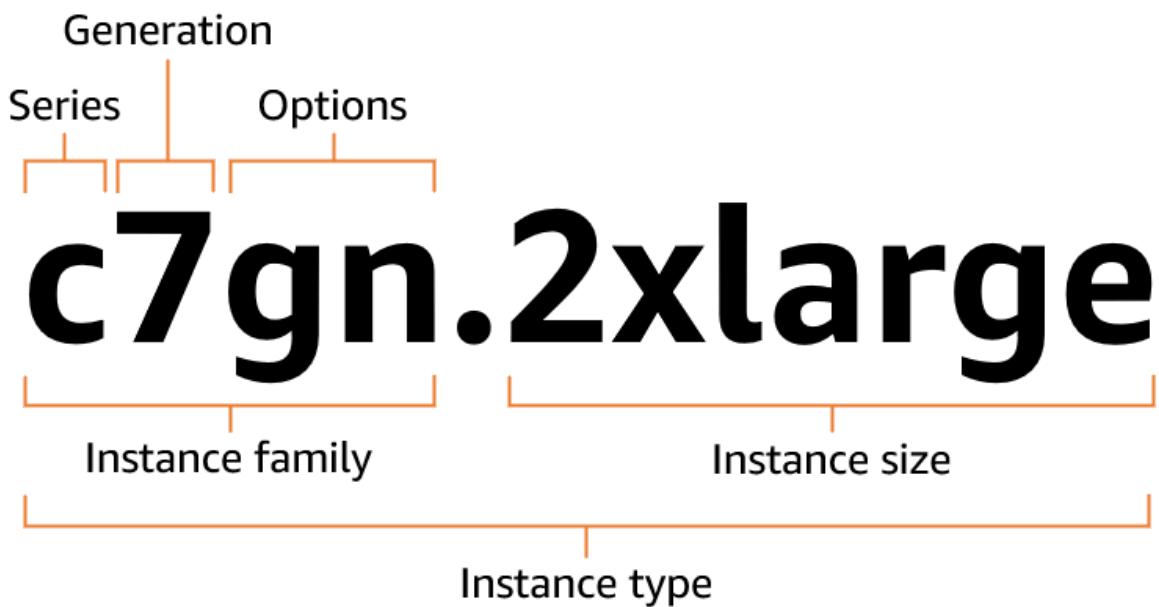
1. **General purpose** instances provide a good **balance** of compute, memory, and networking resources. They can be used for lots of **diverse workloads**, like **web services** or **code repositories**. They're also a good **starting point** if you **don't know how your workload will perform ahead of time**.
2. **Compute-optimized** instances are ideal for **compute-intensive tasks**, like **gaming servers**, **high-performance computing**, **machine learning** tasks--even **scientific modeling**.
3. **Memory-optimized** instances are good for **memory-intensive tasks**. They deliver fast performance for workloads that **process large data sets** in memory. The best choice for real-time analytics.

4. **Accelerated-computing** instances are good for **floating point number calculations, graphics processing, or data pattern matching**. This is because they use **hardware accelerators**. These are **co-processors** that perform functions **more efficiently** than is possible in software running on CPUs.
5. And, finally, **storage-optimized** instances are ideal for workloads that require high performance for **locally stored data**.

After you've decided on an instance type, the next part is to choose the **right instance size**. I urge you to keep not just performance in mind but also **cost**. Sure, bigger instances give you more CPU, memory, and storage, but they also cost more. So, it's important to find a balance that fits your needs and budget. This ensures that you choose the right instance size to get the best performance, but without paying for resources you don't really need.

The best part is that you don't have to use a specific instance type or size forever. You might find that one you initially chose doesn't offer the performance you need. That's perfectly fine. You can change it! The cloud gives you that ability to pivot very quickly, so take advantage of it!

Series	Options
<ul style="list-style-type: none"> * C – Compute optimized D – Dense storage F – FPGA G – Graphics intensive Hpc – High performance computing I – Storage optimized Im – Storage optimized (1 to 4 ratio of vCPU to memory) Is – Storage optimized (1 to 6 ratio of vCPU to memory) Inf – AWS Inferentia M – General purpose Mac – macOS P – GPU accelerated R – Memory optimized T – Burstable performance Trn – AWS Trainium U – High memory VT – Video transcoding X – Memory intensive Z – High memory 	<ul style="list-style-type: none"> * a – AMD processors b200 – Accelerated by NVIDIA Blackwell GPUs g – AWS Graviton processors i – Intel processors m1ultra – Apple M1 Ultra chip m2 – Apple M2 chip m2pro – Apple M2 Pro chip b – Block storage optimization d – Instance store volumes e – Extra storage (for storage optimized instance types), extra memory (for memory optimized instance types), or extra GPU memory (for accelerated computing instance types). flex – Flex instance n – Network and EBS optimized q – Qualcomm inference accelerators * tb – Amount of memory for high-memory instances (3 TiB to 32 TiB) z – High CPU frequency



▼ How to provision AWS Resources

We've been talking about a few different AWS services, as well as the Global Infrastructure. You might be wondering, "How do I actually interact with these services?" And the answer is **APIs**. In AWS, everything is an API call. An API is an **application programming interface** that defines predetermined ways for you to interact with AWS services. And you can invoke or call these APIs to provision, configure, and manage your resources.

There are three main ways you could call AWS APIs: using the **AWS Management Console**, the **AWS Command Line Interface, or CLI**, or the **AWS Software Development Kit, or SDK**. First, let's talk about the console, which is browser based.

Using the **console**, you can manage your resources **visually** and in a way that is **easy** to digest. This is great for getting started and building out your knowledge of the services. It's also helpful for setting up test environments, viewing **AWS bills**, **monitoring resources**, and managing **non-technical tasks**. The console is most likely the first place you will go when you're learning about AWS.

However, after you are up and running in a **production-type environment**, you don't want to rely on the point-and-click style of the console. For example, to create an EC2 instance, you need to navigate through various screens, set all the configurations you want, and then launch your instance. If you wanted to launch another EC2 instance later, you would need to go back into the console and navigate through those screens again to get it up and running.

By having humans do this sort of manual provisioning, you're opening yourself up to potential errors. It's very possible to forget to select a checkbox or misspell something when you are doing everything manually.

Next up, you can use the **AWS CLI** to invoke AWS APIs using a **terminal**. This is different than the visual navigation style of the console, because it allows you to interact with AWS services through text-based input called commands. This makes **automation through scripting** possible.

Let's look at two examples of invoking AWS APIs using the command-line interface. We can run commands using **AWS CloudShell**, which is a cloud-based terminal that has the AWS CLI already installed in a managed environment.

In this example, we're going to create an EC2 instance programmatically with the command, `aws ec2 run-instances`. Let's run the command, and the instance is now initializing. And if you want to do something else, like list all of the AZs in the current Region, you could use the command, `aws ec2 describe-availability-zones`.

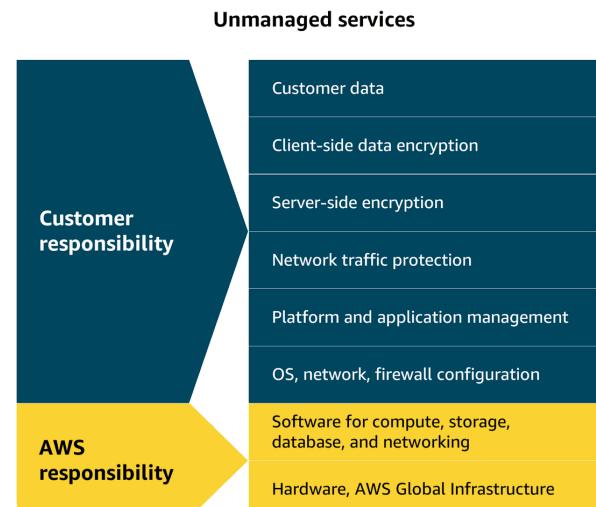
These are two basic examples, but as you explore more AWS services and the resources you can create, you'll discover additional AWS CLI commands to interact with those services as well.

I also want to call out that these commands in this example were run manually, but they could also be included in **scripts and other automation processes**. Automation is an important aspect to having a successful and **predictable cloud deployment over time**. So, with that being said, another way to interact with AWS is through the AWS SDK.

The AWS SDK makes it possible for you to interact with AWS resources through various programming languages, like Python for example. Let's run a Python script using the integrated development environment-IDE, Visual Studio Code, that uses the SDK to list the EC2 instances in the current region.

So, as you begin to see more examples of AWS using AWS resources, remember that AWS hosts APIs that you use to create, interact with, and manage AWS resources. Whether you are using the AWS Management Console, the AWS CLI, or the AWS SDK, these APIs are being called behind the scenes.

An **unmanaged** service like Amazon EC2 requires you to perform all of the necessary security configuration and management tasks. When you deploy an EC2 instance, you are responsible for configuring security, managing the guest operating system (OS), applying updates, and setting up firewalls (security groups). You will learn more about managed and unmanaged services later.



AWS Resources Access Manager - AWS RAM

AWS Resource Access Manager (RAM) is a **free** service that helps you **securely share your AWS resources** with other AWS accounts, organizational units (OUs), or specific IAM roles/users.

In a **multi-account environment**, AWS RAM prevents you from having to **duplicate foundational resources** across every account, which saves time and cost.



Think of AWS RAM as a **digital key-sharing service** for your shared infrastructure. Instead of making a copy of a large, expensive asset (like a whole fence or building) for every team, you just give them a key to the original.

How it Works

1. **Resource Owner Account:** Creates the central resource (e.g., a **VPC Subnet** or a **Transit Gateway**).
2. **Resource Share:** The owner uses RAM to create a "Resource Share," defining:
 - Which resource(s) to share.
 - Which accounts/OUs/IAM principals (the "consumers") to share with.
3. **Resource Consumer Accounts:**
 - If within the same AWS Organization, access is usually accepted automatically.
 - If **outside the Organization**, the recipient must **manually accept the invitation**.
4. **Usage:** The consumer accounts can then use the shared resource **as if it were native** in their own account, subject to the permissions defined by the owner.

Key Benefits

- **Cost Efficiency:** Avoids duplicating expensive resources like NAT Gateways or Transit Gateways.
- **Simplified Management:** You only provision and manage the resource once in the owner account.
- **Consistent Networking:** Essential for creating shared VPC architectures where multiple teams deploy applications into the same, centrally managed network.

▼ Configure and Launch an Amazon EC2 Instance

<https://aws.amazon.com/ec2/instance-types/>

You've been learning a lot about EC2, and by now, you're probably thinking, "How do I create an EC2 instance myself?" In this video, we're going to walk through the process of creating an EC2 instance for a web server using the AWS Management Console. We'll go over some key configurations along the way, so let's get started! First thing we want to do is go to the EC2 console.

Now, I happen to have it here in my "Recently visited" and in my shortcuts bar, but I can just as easily search for EC2 and go to that service. There are tons of options in here, but we're gonna go and zero in, specifically, on the launch instance option.

First thing we have to choose is a **name**. We have to choose our instance name, so we can find it later. Next thing we're gonna do is choose the Amazon Machine Image, or **AMI**. An AMI is a **template of the operating system and the built-in applications** that it's going to come with. We can customize this but for right now, we're going to go with a **default**, the **Amazon Linux AMI**. Now this Amazon Linux AMI is perfect for a **general-use web server**.

An **AMI** is a **pre-configured virtual machine image** that contains the operating system, application server, and applications. This helps to launch EC2 instances quickly with the desired software and settings.

An **AMI** includes the **operating system, storage setup, architecture type, permissions for launching**, and any **extra software** that is already installed. You can use one AMI to launch several EC2 instances that all have the same setup.

AMIs can be used in three ways. First, you can create your own by building a **custom AMI** with specific configurations and software tailored to your needs. Second, you can use **pre-configured AWS AMIs**, which are set up for common operating systems and software. Lastly, you can purchase AMIs from the **AWS Marketplace**, where third-party vendors offer specialized software designed for specific use cases.

AMIs provide **repeatability** through a consistent environment for every new instance. Because configurations are identical and deployments **automated**, development and testing environments are consistent. This helps when scaling, reduces errors, and streamlines managing large-scale environments.

The next thing we have to decide on is the **instance type**. This refers to how much **computing power** our web server is going to have. In this case, we're gonna choose just the basic, **t2.micro**. We'll go over some of the details that makes up this **t2.micro** a little bit later, but for right now, this is going to be fine. One virtual CPU, one gigabyte of memory, and it's in the **Free Tier**. Sounds good to me.

The next thing we choose is the **key pair**. And this is related to how we are **logging into this EC2 instance**. A key pair refers to a pair of keys. Specifically, a **public key**, which is going to be injected into the EC2 instance, and a **private key**, which we will keep. We can create that key pair right here, but I'm gonna go ahead and choose a key that we've already established.

The next section is choosing the network settings. We're going to have a lot of fun going through all these details a little bit later. But for right now, suffice it to say, we're going to allow **HTTP traffic** from the **internet**. It is going to be a **web server**, after all, so that's all we have to do.

The next option we have to choose is the **storage options** for our EC2 instance. In this case, we're going to give it eight gigabytes of disk space using the **gp3 EBS volume**, or elastic block store volume. We'll go over what that means a little bit later, but this means it's going to have plenty of space for web serving.

And speaking of web serving, we do have to make one more change, so it actually is a web server. When we picked the AMI, we picked a very generic AMI. This doesn't have the web server activated at launch. In order to that, we're going to go into the Advanced Details, specifically down to the **User Data** section. What the User Data allows us to do is paste in a **script**, such as this one, which will go ahead and install and activate the **Nginx web server**, which is what we're going to be using to **server content to the internet**.

All this looks fine, so I'm gonna go ahead and hit launch instance, and we'll see what we get. As soon as the instance launches, we can go to its console and see some of the details about that EC2 instance.

Now that our EC2 instance is running, I'm going to copy its public IP address. I'm going to open a new browser, and let's go to it. And there you have it! Your very own EC2 instance running a basic web server.

Congratulations! That's the basic thing for setting up EC2 instances, and we're gonna get into more details about the nuances later in the course. So stay tuned.

▼ Amazon EC2 pricing

We talked about Amazon EC2 instance types, but you're all probably wondering, "How much is this gonna cost me?" Well, don't fret. For EC2, we have multiple billing options available.

The most widely known option is called **On-Demand**. This means you only pay for the **duration** that your **instance runs**. This can be **per hour** or **per second**, depending on the **instance type** and the **OS you choose**. Even better, **no long-term commitments or upfront payments** are needed. Most customers typically use this option when they get started. This makes it possible to spin up servers, play around, test out workloads. It's all **self-service**. This helps you figure out a **baseline** for your **average usage**. Using that average, you can start to explore other pricing options, like the next one, Savings Plans.

Savings Plans offers **lower EC2 prices** for a **commitment to a consistent amount of usage**. This is measured in **dollars per hour** for a **one-year** or **three-year term**. This can provide **savings** of up to **72 percent**, which is pretty significant. In fact, it can lower prices on your EC2 usage, regardless of instance family, size, OS, tenancy, or AWS Region. Additionally, it applies to AWS **Fargate** and AWS **Lambda** usage. Those are **serverless compute** options that we'll cover later. Best for **Dynamic or changing workloads** where instance **types** or generations are likely to **change over time**, or for a broad usage commitment across **multiple compute services**.

Example: You commit to spend **\$5 per hour** on **EC2 M-family instances** in the **US-East** region. If you are running an m5.large, the discount applies. If you **switch** to an m5.xlarge, the **discount still applies** until the **\$5/hour** is **used up**. If you **switch** from Linux to Windows, the **discount still applies**.

Another option is **Reserved Instances**. These are well-suited for **steady-state workloads** or ones with **predictable usage**. They offer you up to a **75 percent discount** compared to On-Demand pricing. You qualify for a discount after you **commit** to a **one-year or three-year term**. There are also three payment options: all **upfront**, where you pay for them in full when you commit; **partial upfront**, where you pay for a portion when you commit; and **no upfront**, where you don't pay anything at the beginning. Best for **Highly predictable, stable workloads** where you know you will run the exact same instance type, size, and OS 24/7. Reserved Instances are older and more rigid. They require you to **commit to specific hardware configurations**. Example (**Standard RI**): You buy a reservation for **one m5.large Linux instance** in **us-east-1**. The discount **only** applies to that **specific configuration**. If you change it to a c5.large (different family), the RI goes **unused**, and the new instance is **charged On-Demand**.

The next option is **Spot Instances**. These make it possible to **request spare EC2 capacity** for up to **90 percent off** of the On-Demand price. The catch here is that **AWS can reclaim the instance at any time**. However, you do receive a **two-minute warning**, so you can **save your progress**. And you can always **resume later** if needed. So, make sure your **workloads can tolerate being interrupted** if you choose Spot Instances.

And, finally, we have **Dedicated Hosts**. These are **actual physical servers** that customers can reserve for **exclusive use**. No other customer's workloads can share the server. This **isolation** use-case is ideal for **security-sensitive or licensing-specific workloads**, like **Windows or SQL Server**. This is because you have control over instance placement and resource allocation. This helps with meeting certain **compliance** and **regulatory** needs.

Dedicated Instances: Pay for instances running on hardware dedicated solely to your account. This option provides isolation from other AWS customers.

- **Dedicated Instances:** You get a guarantee that your instance will run on hardware used **only by your AWS account**. AWS manages *which* physical server it is on.
- **Dedicated Hosts:** You get a guarantee that an **entire physical server** is exclusively yours, and you have **control** over where your instances are placed on that server.

With the variety of EC2 pricing options available, you can choose the most cost-effective solution based on your usage patterns. This helps you to balance flexibility, cost savings, and specific workload needs.

*add a comparison table later - See the Summary note

▼ Scaling Amazon EC2

Part 1

So, we have a good idea now on the basics of Amazon EC2 and how it can help with any compute needs, like making coffee. Well, like, metaphorically making coffee. The coffee represents the...whatever your instance is producing. In reality, the instances could be fielding web requests, processing and analyzing data, or hosting other types of applications.

So now, the next thing we want to talk about is another major benefit of AWS, **scalability** and **elasticity**. This is how capacity can grow and shrink, based on business needs.

If we're running a business and we are planning for growth, we might know on average what's needed capacity-wise. Except the average can include cyclical traffic with busy and quiet seasons. We want happy customers, so we always want enough capacity, but should we plan for the peak usage? What if that peak is only an hour long? Or what if we don't know when the peaks are because it's a new business? Should we buy a lot of extra capacity? That sounds expensive. What's the alternative? Plan for a smaller amount and hope for the best?

What if you could **provision** your workload to the **exact demand**, every hour, every day? Well, now you have happy customers because they can always get the services they want. And you have a happy financial officer because they get the cost savings your company needs.

And here's how it works.

Morgan is behind the counter. She's taking orders, but she isn't doing all the work here, so we need somebody making the drinks. It looks like Rudy's up. Let's ask ourselves what would happen if we lost our order-taking instance. Well, we'd be out of business until we get another person to work the line, or another instance up and running.

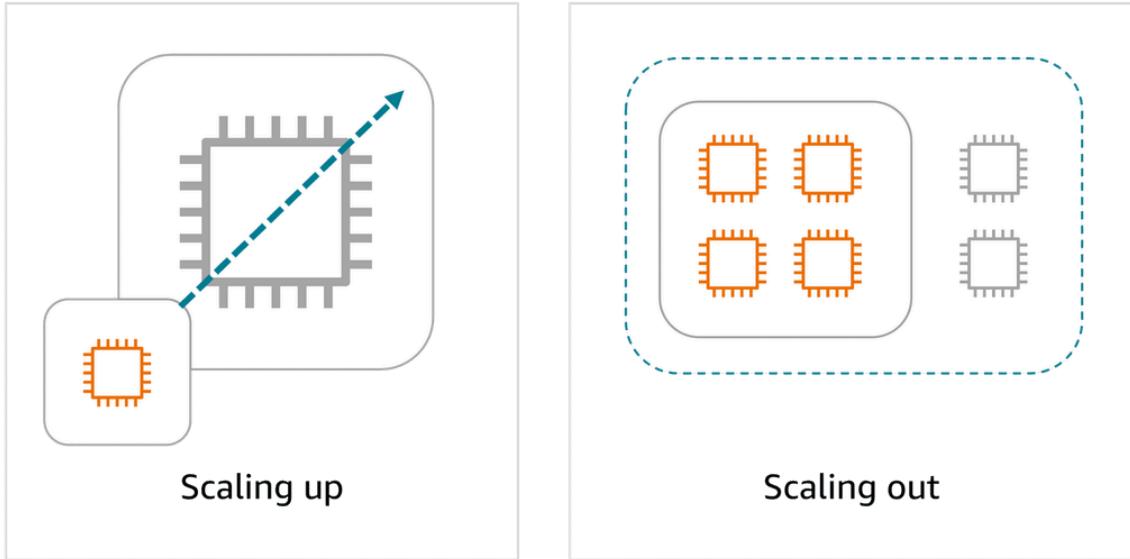
So, here is where AWS makes it really straightforward. Using the same programmatic method that we used to create the original Morgan, we can create a second copy of Morgan.

So, if one fails, we have another one already on the frontline taking orders. The customers never lose service, and in the meantime, we can always spin up another instance of Morgan to replace the one that failed. And let's not forget about the backend. Let's make our processing instances redundant, as well.

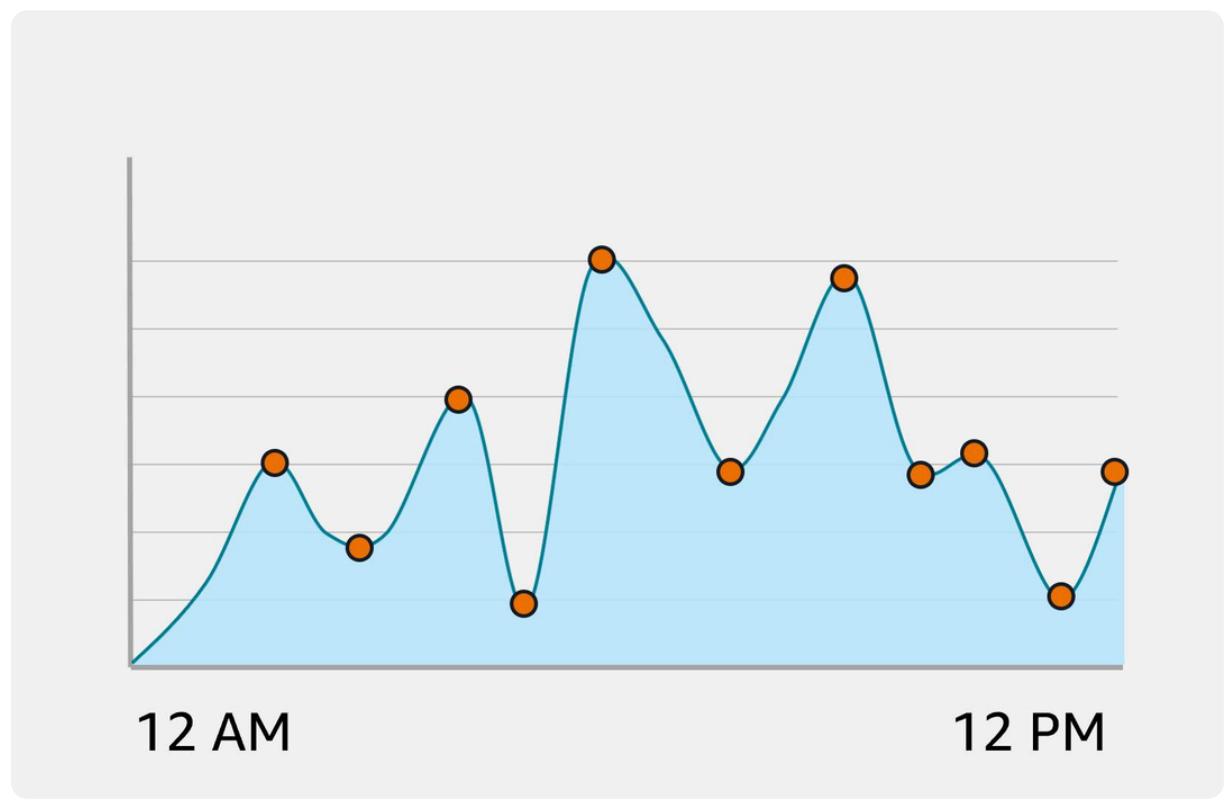
That solves for our regular operating capacity. We now have a **highly available system** with **no single point of failure**. This is why it's a best practice to set up redundant EC2 instances, and to deploy them across multiple AZs in a Region. That way, if there are issues in one place, the instances deployed in the other AZ can pick up the slack.

And as long as the number of customers in line stays the same, we're good. But you know that will change, right? So, let's take a look at what's going to happen when we have a rush of customers—or an increase in demand.

Scalability refers to the ability of a system to **handle an increased load by adding resources**. You can **scale up** by adding **more power to existing machines**, or you can **scale out** by adding **more machines**. Scalability focuses on long-term capacity planning to make sure that the system can grow and accommodate more users or workloads as needed.



Elasticity is the ability to **automatically scale resources up or down** in response to **real-time demand**. A system can then rapidly adjust its resources, scaling out during periods of high demand and scaling in when the demand decreases. **Elasticity** provides **cost efficiency** and **optimal resource usage** at any given moment.



The question of downtime depends on how you set up your redundancy, specifically the load balancing and failover mechanism.

1. Active/Active (Zero Downtime)
 2. Active/Passive (Minimal Downtime)
- **Failover:** If Server A fails, an automated system (like **Auto Scaling** or a custom script) is triggered to **start** Server B and reroute the traffic to it.

- **Downtime:** There will be a brief period of downtime (anywhere from seconds to a few minutes) while Server B boots up and initializes the application.

In summary, for a true high-availability setup, both EC2 instances are kept Active/Active to ensure seamless, instantaneous failover with essentially no downtime for the end-user.

Failover Time Calculation

The failover time is the time it takes the **load balancer** to recognize an instance is unhealthy and stop sending it new traffic. This is controlled by three main parameters you define:

1. **Interval:** The time (in seconds) between health checks.
 - *Default for ALB/NLB: 30 seconds* (can often be set as low as 5 seconds).
2. **Timeout:** The time (in seconds) the load balancer waits for a response from the instance before considering that check a failure.
 - *Default for ALB: 5 seconds.*
3. **Unhealthy Threshold:** The number of **consecutive failed checks** required before the load balancer marks the instance as officially unhealthy and removes it from the rotation.
 - *Default for ALB: 2 checks.*

$\text{Failover Time} \approx \text{Interval} \times \text{Unhealthy Threshold} = 30 \text{ seconds} \times 2 = 60 \text{ seconds}$

What Happens During Failover

The key to **zero service interruption** (for a new request) is that the traffic is only routed to the **healthy** servers (Server B in the example). The delay only affects how quickly the ELB recognizes Server A is dead, but it does not stop Server B from serving traffic.

- **New Connections:** As soon as the ELB marks Server A as unhealthy, all new incoming user connections are routed instantly to the healthy Server B. The user experiences no delay beyond their initial connection request.
- **Existing Connections:** Any user who had an *active, in-progress* connection to Server A when it failed will have that specific request fail. They will need to retry their operation, which will then be directed to Server B.

Part 2

Now there are two ways to handle growing demands. You can scale out or scale up. Scaling out, otherwise known as **horizontal scaling**, is when you **add more resources to the pool**, so you can get more work done in parallel. Scaling up, otherwise known as **vertical scaling**, means adding **more power** to the machines that are running. This is so that the **individual machine** itself has more power to do the work. Scaling up can give you more power per instance, but that isn't always what you need.

Let's take this idea of scaling into our coffee shop.

Here, when we have an increase in customers, a bigger instance of me really can't take a customer's order any faster. That depends on the customer more than me. I'll take an espresso. Oh, wait, is that organic? Mmmm...make it a soy latte. Actually, I don't know. Do you just have tea? What we need are, well, more instances of me to handle more customers in **parallel**.

Let's get more customers in here. Well, now, that's not good. It looks like the processing instances are about to get overloaded because the orders are coming in faster. The orders are processed quicker, but the drinks aren't getting done and customers are waiting. Let's go ahead and scale the processing instances, as well.

Here comes a maybe obvious question. How come there are more order-taking instances than order-making instances?

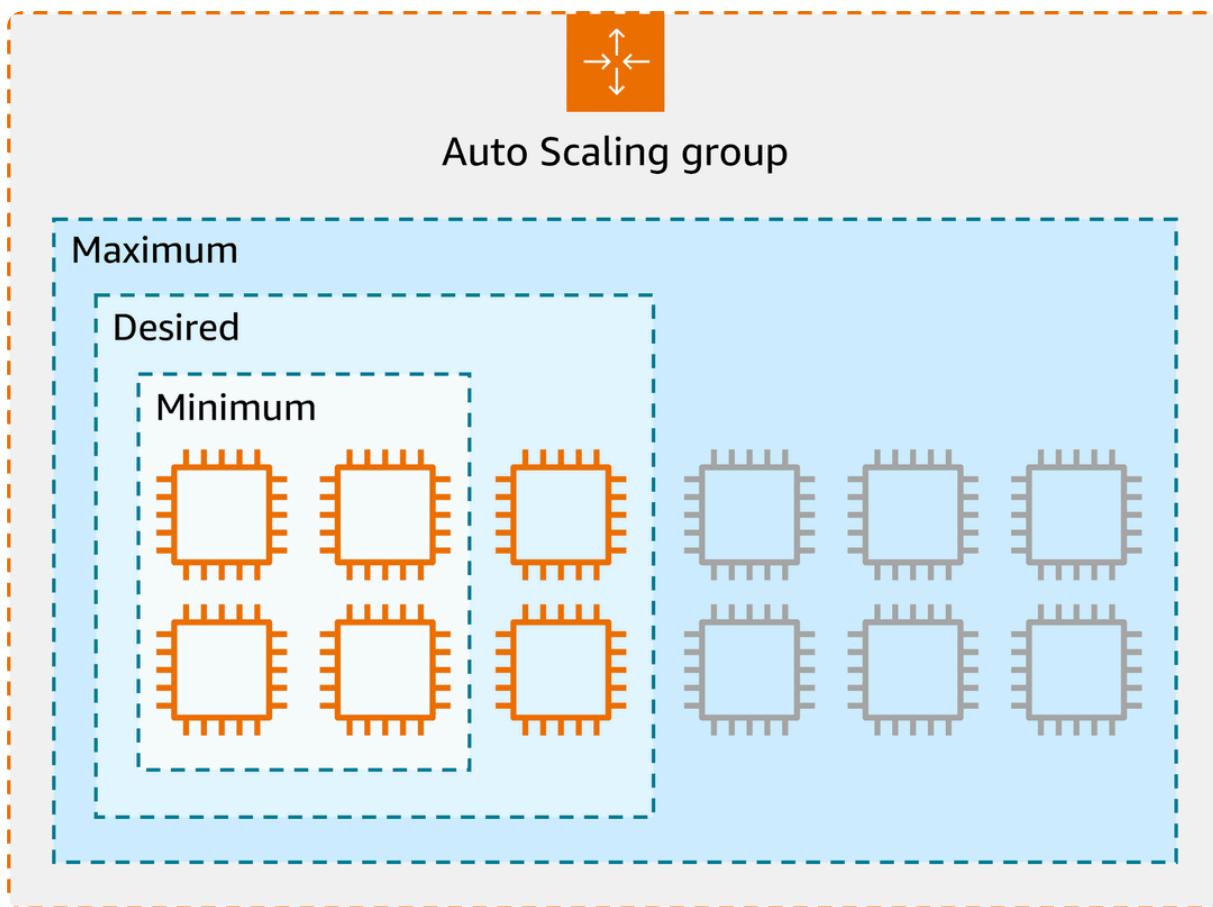
Well, in this case, the amount of work getting done is still more than the order-taking instances can send to the back of the house. There isn't currently any backlog of orders, so there's no reason to add more worker instances. This way, you can end up with exactly the right amount of power for each part of your process, rather than **over provisioning** to solve a **separate problem**. You can scale each piece up and down independently as needed.

Okay, looks like we just cleared that rush. Now here is where AWS really makes a difference to your business. We have extra workers that are sitting around idle. If we don't need them, we can send them home or stop the instances. This is how **Amazon EC2 Auto Scaling** works.

EC2 Auto Scaling adds instances based on demand and **key scaling metrics** and then **decommissions** instances when that **demand goes down**. This means that every minute of the day, you always have the desired number of instances.

The way this works technically involves some other AWS services to make it all happen. You need to be **collecting data** about the **performance of the instances**, or potentially **data** around **latency** and **other application metrics**. You would use the **Amazon CloudWatch** service to **collect** and **monitor** these **metrics**. This data is then used to determine when scaling needs to happen. And, it happens **automatically**, right when you need it.

Amazon EC2 Auto Scaling automatically adjusts the number of EC2 instances based on changes in application demand, providing better availability. It offers two approaches. **Dynamic scaling** adjusts in **real time** to fluctuations in demand. **Predictive scaling** **proactively** schedules the right number of instances based on anticipated demand.



Scaling Out vs. Scaling Up Comparison

Related Project

 Cloud Certified Practitioner Essentials note

 AWS

Scaling is the process of modifying an application's infrastructure capacity to handle increased load. The choice between **Scaling Out (Horizontal)** and **Scaling Up (Vertical)** is a fundamental decision in cloud architecture, primarily driven by performance needs, cost goals, and resilience requirements.

Here is a comprehensive comparison table:

Feature	Scaling Out (Horizontal)	Scaling Up (Vertical)
Action	Adding more servers/nodes	Replacing current server with a larger one
Analogy	Adding more checkout lanes at a grocery store.	Replacing a small truck with a much bigger truck.
AWS Example	Adding more EC2 instances via Auto Scaling Group;	Changing an EC2 instance from

	adding more Read Replicas to a database.	t2.micro to m6i.xlarge.
Limit	Theoretical unlimited scaling (limited by AWS region capacity).	Limited by the largest available instance size in the cloud provider's catalog.
Availability/FT	High ✅. Removes Single Point of Failure (SPOF). If one server fails, others take over.	Lower ⚠️. The single, large server is still a SPOF. Failover to a standby still causes downtime.
Downtime	Zero (rolling deployment). Traffic is seamlessly diverted to new nodes.	Requires Downtime (brief maintenance window or reboot) to swap the instance type.
Cost Model	Linear . Costs increase steadily as you add capacity, but you can scale down automatically when load decreases.	Exponential . Costs jump significantly with each size increase (e.g., doubling the size is often more than double the cost).
Stateless Apps	Excellent	Good
Stateful Apps	Challenging (requires shared storage like EFS or database sharding).	Simple (all state remains on the single server).

🎯 When to Choose Which Strategy

The modern cloud approach favors **Scaling Out** wherever possible due to its superior resilience and cost-efficiency under variable load.

⬆️ Choose Scaling Up (Vertical) When:

- Strict Licensing:** Your application software license is bound to a single physical CPU socket or server (e.g., legacy enterprise software like Oracle or Microsoft SQL Server Standard Edition).
- Immediate Necessity:** You need a quick fix for performance and the load increase is **predictable** and **sustained** (e.g., you need to double memory, but you know you'll always need that memory).
- Monolithic Architecture:** Your application cannot be broken down or clustered (e.g., an old monolith that requires all compute and memory on a single machine).

4. **Database Write Limit:** You need to increase the raw processing power or connection limit of a single Primary Database instance to handle high write concurrency.

→ Choose Scaling Out (Horizontal) When:

1. **Variable/Spiky Load:** Your traffic fluctuates wildly (e.g., an e-commerce site on Black Friday or a ticket sale). Auto Scaling can add 10 servers and then remove them 3 hours later, saving significant cost.
2. **High Resilience:** You need high availability across multiple Availability Zones (AZs) to withstand a data center outage (e.g., any customer-facing web application).
3. **Cost Optimization:** You want to pay only for the resources you are actively using, ensuring you have no idle capacity.
4. **Cloud-Native Design:** You are using modern, serverless, or containerized architectures like **AWS Fargate** or **Lambda**, which are inherently horizontal solutions.

▼ Directing traffic with Elastic Load Balancing

We solved the scaling problem with Amazon EC2 Auto Scaling. But now we've got a bit of a traffic problem, don't we? Let's take a look at the situation.

Customers have three cashier options, but it seems like they are gravitating towards me, cause I'm just so darn adorable. Look, although I am flattered they noticed my new haircut, this is causing an uneven distribution of customers per line.

Look over there. Morgan and Alan have no customers and are just taking selfies. Pfft, they should have gotten new haircuts, too. Anyway, in place of that, it would help a lot if we added a host to our coffee shop.

The host stands at the entrance and directs customers to a specific line when they walk in. They also keep an eye on the cashiers and count the number of people in each line. Using this real-time information, they direct customers to the shortest line. This helps with evenly distributing customers and ensures they are served efficiently and quickly.

The same idea applies to your AWS environment when trying to balance traffic across a group of EC2 instances. We don't want **idle** EC2 instances, nor do we want overloaded ones. This is where we introduce the concept of a load balancer.

A **load balancer** takes in requests and routes them to instances. There are many off-the-shelf load balancers that work great on AWS. So, if you have a favorite flavor that already does exactly what you want, feel free to keep using it. Just remember that you will have to **manage, patch, upgrade, handle failover, and perform maintenance** on it.

However, if you would like AWS to handle all of that, and you just configure it once, check out **Elastic Load Balancing**, or **ELB**. ELB is designed to **distribute network traffic** to improve application **scalability**. The word *elastic* refers to its ability to scale up or down based on traffic, without adding to your hourly costs.

ELB can manage both internal and external traffic to AWS. It offers different routing strategies to ensure efficient traffic management and thusly optimal application performance.

Let's dive into an example. For our coffee shop, we have a website for customers to order drinks. This website has various tiers for functionality, like the ordering tier, production tier, storage tier, and so forth.

Let's look at the ordering tier and how it communicates with the production tier. Right now, every frontend instance is aware of every backend instance. If a new backend instance spins up, then it needs to let every frontend instance know that it can now accept traffic.

This is complicated enough with half a dozen instances. Imagine if we have hundreds or thousands of instances in every tier. Keeping them in sync would be a huge undertaking.

That's why ELB is here to help. We can use it to manage the linking of the backend instances and the frontend instances. And because it's Regional, it's a **single URL** that each frontend instance uses to direct to the backend instances. The ELB will then direct traffic to the backend instance that has the least outstanding requests.

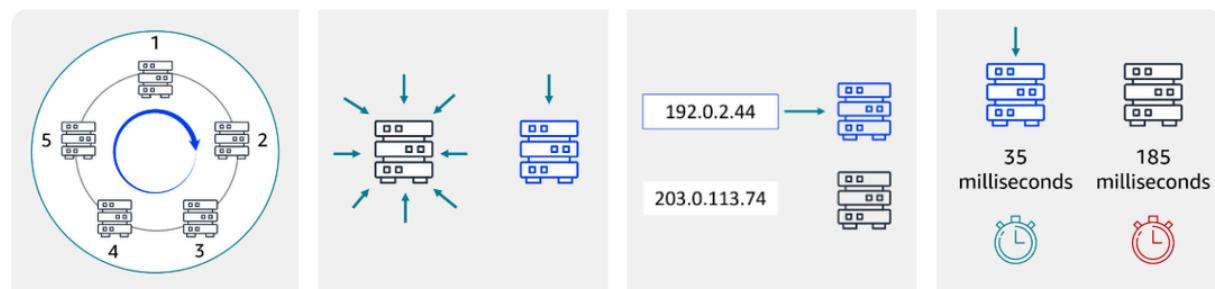
If the back-end needs to scale, it spins up a new instance. After that new instance is ready, it tells the ELB that it's ready for traffic. And just like that, it gets to work.

The frontend doesn't even need to know what's happening. It's all done **automatically**, and it **decouples** the architecture so that each tier is considered its own entity that scales as it sees fit.

Efficient traffic distribution: ELB evenly distributes traffic across EC2 instances, preventing overload on any single instance and optimizing resource utilization.	Automatic scaling: ELB scales with traffic and automatically adjusts to changes in demand for a seamless operation as backend instances are added or removed.	Simplified management: ELB decouples front-end and backend tiers and reduces manual synchronization. It also handles maintenance, updates, and failover to ease operational overhead.
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Routing methods

To optimize traffic distribution, ELB uses several routing methods: **Round Robin**, **Least Connections**, **IP Hash**, and **Least Response Time**. These routing strategies work together for efficient traffic management and optimal application performance.



Round Robin

Distributes traffic evenly across all available servers in a cyclic manner.

Least Connections

Routes traffic to the server with the fewest active connections, maintaining a balanced load.

IP Hash

Uses the client's IP address to consistently route traffic to the same server.

Least Response Time

Directs traffic to the server with the fastest response time, minimizing latency.

See the differences

ELB is responsible for **distributing incoming traffic** evenly across multiple EC2 instances.

Together, ELB and Auto Scaling help maintain application reliability and cost efficiency.

▼ Messaging and Queuing

Alan: Oh, sorry. I was just ordering a cup of coffee. Let's talk about messaging and queuing. In the coffee shop, there are cashiers taking orders from the customers and baristas making the orders.

Currently, the cashier takes the order, writes it down with a pen and paper, and delivers this order to the barista. The barista then takes the paper and makes the order. When the next order comes in, the process repeats. This works great, as long as both the cashier and the barista are in sync.

But what happens if the cashier took the order, turned to pass it to the barista, and the barista was on break or busy with another order? Well, that cashier is stuck until the barista is ready to take the order. And, at a certain point, the order will probably be dropped so the cashier can go serve the next customer.

You can see how this is a flawed process, because as soon as either the cashier or barista is out of sync, the process will degrade. This will cause slowdowns in receiving orders—and failures to complete orders at all. A much better process would be to introduce some sort of **buffer** or **queue** into the system. Instead of handing the order directly to the barista, the cashier would post the order to an order board.

This idea of placing messages into a buffer is called messaging and **queuing**. Just as our cashier sends orders to the barista, applications send messages to each other to communicate. If applications communicate directly, like our cashier and barista previously, that is called being **tightly coupled**. A hallmark trait of a tightly coupled architecture is this: If a **single component fails** or changes, it causes issues for the other components or even the **whole system**.

For example, Application A is sending messages directly to Application B. If Application B has a failure and cannot accept those messages, Application A will begin to see errors, as well. This is an architecture where if one component fails, it is isolated and therefore won't cause cascading failures throughout the whole system. If we designed the application to use a more loosely coupled architecture, it could look like this.

Just like our cashier and barista, we introduced a buffer between the two. In this case, we introduced a **message queue**. Messages are sent into the queue by Application A and are processed by Application B. If Application B fails, Application A doesn't experience any disruption. Messages being sent can still be sent into the queue and will remain there until they are eventually processed.

This is **loosely coupled**. This is what we strive to achieve with architectures on AWS. And this brings me to two AWS services: Amazon **Simple Queue Service**, or Amazon **SQS**, and Amazon **Simple Notification Service**, or Amazon **SNS**.

Simple Queue Service - SQS

SQS makes it possible for you to **send**, **store**, and **receive messages** between software components at any volume. This is done without losing messages or requiring other message consumers to be available.

Think of messages as our coffee orders and the order board as an SQS queue. Messages have the person's name, coffee order, and the time they ordered. The **data within a message** is called a **payload**. SQS queues are where the messages are placed until they are processed. These **scale automatically**, are **reliable**, and are **simple** to configure and use.

Simple Notification Service - SNS

SNS is similar as it also sends messages to services, but it has a big distinction: Sent SNS messages aren't held for pickup until the processing service has a moment to get to them. Instead, SNS messages **need a response right now**. If SQS is the coffee orders board, SNS is the barista yelling out, "One Rudy's Refresher, to go!"

Alan: Additionally, SNS can be used to fan out notifications to end users using mobile push, SMS, and email. Taking this back to our coffee shop, we could send out a notification when a customer's order is ready. This could be a simple text message.

Monolithic applications (coupled) vs Microservices architecture (loosely coupled)

Amazon **EventBridge**, Amazon **SNS**, and Amazon **SQS** are AWS services that help different parts of an application communicate effectively in the cloud.

EventBridge

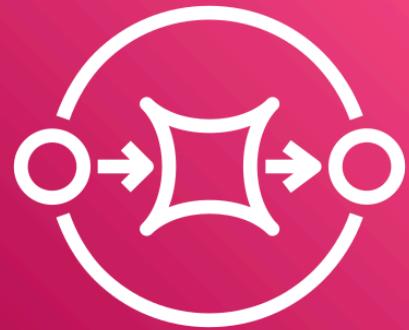
EventBridge is a **serverless service** that helps **connect different parts** of an application using **events**, helping to build scalable, **event-driven systems**. With EventBridge, you route events from **sources** like **custom apps, AWS services, and third-party software** to other applications. EventBridge simplifies the process of **receiving, filtering, transforming, and delivering events**, so you can quickly build reliable applications.



How EventBridge helps: EventBridge can route events, like *order placed* or *payment completed*, to the relevant services (payment, restaurant, inventory, and delivery). It can handle **high volumes** of events during peak times, making sure **each service** works **independently**. Even if one service fails, EventBridge will store the event and process it as soon as the service is available again. EventBridge helps provide a smooth and reliable operation across the entire system.

Amazon SQS

Amazon SQS is a message queuing service that facilitates reliable communication between software components. It can send, store, and receive messages at any scale, making sure messages are not lost and that other services don't need to be available for processing. In Amazon SQS, an application places messages into a queue, and a user or service **retrieves** the message, **processes** it, and then **removes** it from the queue.



Amazon SNS

Amazon SNS is a publish-subscribe service that publishers use to send messages to subscribers through SNS topics. In Amazon SNS, subscribers can include web servers, email addresses, Lambda functions, and various other endpoints. You will learn about Lambda in more detail later.



Feature	Amazon SQS (Simple Queue Service)	Amazon SNS (Simple Notification Service)	Amazon EventBridge (Event Bus)
Communication Model	Queue-Based (1:1/Pull)	Publish/Subscribe (1:Many/Push)	Event Bus (1:Many/Rule-Based Push)
Message Flow	Producer sends a message to a queue. One consumer <i>pulls</i> it out for processing.	Publisher sends a message to a topic. All subscribed endpoints are <i>pushed</i> the message.	Event Source sends an event to a bus. Rules match the event and <i>push</i> it to targets.
Message Persistence	Durable. Messages are stored in the queue for up to 14 days until they are successfully processed.	Not Durable. Messages are instantly pushed; if a subscriber is unavailable, the message is lost.	Not Durable. Events are processed and routed in real-time.

	processed and deleted.	(unless delivered to an SQS queue).	
Primary Use Case	Decoupling and Asynchronous Task Processing. Best for background jobs, rate limiting, and ensuring message durability.	Broadcasting and Real-Time Notifications. Best for sending one message to multiple recipients (microservices, email, SMS, mobile push).	Event-Driven Architectures and Integration. Best for complex routing, advanced filtering, and connecting AWS services with SaaS providers (e.g., Zendesk, Datadog).
Ordering	Yes, with FIFO Queues (First-In, First-Out)	Yes, with FIFO Topics (only when subscribing to FIFO SQS queues).	No (best-effort delivery, order not guaranteed).
Filtering	Basic, based on message attributes.	Yes, subscribers can use Subscription Filter Policies to receive only messages they care about.	Advanced. Rules can match complex patterns and content within the event payload itself.

Comparison Table for Amazon EC2 Instance Types

AWS CCP

Amazon Elastic Compute Cloud (Amazon EC2) instances are virtual servers with diverse configurations of CPU, memory, storage, and networking capacity, designed to match various application requirements. Choosing an instance type allows you to select the appropriate mix of resources to optimize performance and cost-efficiency.

The following table compares the main categories and notable specialized types of Amazon EC2 instances based on their optimization, processor architecture, and typical use cases described in the sources.

Comparison Table for Amazon EC2 Instance Types

Instance Family/Type	Primary Optimization/Use Case	Key Processor/Architecture	Key Features and Use Cases
General Purpose (M)	Provides a balance of compute, memory, and network resources for a broad range of		

	general-purpose workloads.		
M7g	General Purpose	AWS Graviton3 processors (latest generation)	Offers the best price performance for general purpose workloads. Ideal for application servers, microservices, gaming servers, mid-size data stores, and caching fleets.
M7i	General Purpose	4th Generation Intel Xeon Scalable processors	SAP certified; ideal for supporting enterprise applications that need larger instance sizes or high continuous CPU usage. Delivers up to 40,000 Mbps EBS bandwidth and 50 Gbps network bandwidth.
M6g/M6gd	General Purpose	AWS Graviton2 processors	Provides a balanced mix of resources. M6gd features local NVMe-based SSD block-level storage.
Burstable Performance (T)	Designed for low-to-moderate CPU utilization workloads that can temporarily burst to full CPU usage when needed.		The T instance family is the most cost-effective way to run general purpose applications with low-to-moderate CPU usage. Recommended for web servers, small/medium databases, virtual desktops, and dev/test environments.
T4g (Latest Generation)	Burstable Performance	Arm-based AWS Graviton2 processors	Offers the lowest cost EC2 instance type and the best price performance (up to 40% higher price/performance and 20% lower costs than T3).
T3a	Burstable Performance	AMD 1st gen EPYC processors (x86-based)	Lowest cost x86-based instances (10% lower costs than T3 instances).
T2 (Previous Generation)	Burstable Performance	Intel Xeon processors	Provides baseline performance with the ability to burst. The smallest sizes (nano,

			micro, small, medium) are well suited for workloads that need a small amount of memory.
Compute Optimized (C)	Ideal for compute-intensive workloads that require high processing power.		
C7g	Compute Optimized	Latest generation AWS Graviton3 processors	Provides the best price performance in EC2 for compute-intensive workloads. Ideal for HPC, batch processing, gaming, video encoding, distributed analytics, and ad serving.
Memory Optimized (R, X, Z)	Designed for memory-intensive applications requiring high memory throughput and a high memory-to-vCPU ratio.		Ideal for running memory-intensive workloads such as in-memory caches and databases (including open-source databases like MySQL and PostgreSQL).
R7g	Memory Optimized	Latest generation AWS Graviton3 processors	Offers the best price performance for memory-intensive workloads.
R7i	Memory Optimized	4th Generation Intel Xeon Scalable processors	SAP-Certified and ideal for running memory-intensive workloads in open-source databases.
R5b	Memory Optimized	Intel Xeon Platinum processors (AWS Nitro System)	Optimized for throughput-intensive applications ; delivers up to 60 Gbps bandwidth and 260,000 IOPS of EBS performance (fastest block storage performance on EC2).
X2g	Memory Optimized	AWS Graviton2 processors	Optimized for memory-intensive applications, offering a low cost per GiB of memory .
Storage Optimized (I, D)	Focuses on high sequential read and write access		

	to very large datasets on local storage.		
Accelerated Computing (Inf, P, G, F, DL, Trn)	Utilizes hardware accelerators like GPUs or AWS-designed chips to maximize performance for specialized tasks.		
Inf2	Accelerated Computing (Inference)	AWS Inferentia2 (second-generation accelerator)	Purpose-built for deep learning inference . Delivers high performance at the lowest cost in EC2 for generative AI models, including Large Language Models (LLMs).
Optimized Reads (Specific)	Specific configurations used to enhance read latency and performance (mentioned for RDS/Aurora read replicas).		
R6id / R6gd	Optimized Reads (Memory Optimized)	3rd Gen Intel Xeon Scalable / AWS Graviton2	Ideal for memory-intensive workloads, offering local NVMe-based SSD block-level storage for low latency storage needs. R6id offers up to 7.6 TB of direct-attached NVMe-based SSD storage.
Mac Instances	Dedicated to development, build, test, and sign applications for Apple platforms .	Apple Silicon (M1, M2, M1 Ultra) or Intel Core i7.	These instances natively support the macOS operating system. They are billed per dedicated host.

Additional EC2 Instance Information

Amazon EC2 instances are virtual machines (virtual servers) that you can rent in the cloud, paying by the second.

Key Hardware Attributes & Concepts:

- **Instance Types and Sizing:** Instance types are named based on their family, generation, processor family, additional capabilities, and size. Each instance type includes at least one instance size, allowing resources to scale to workload requirements.
- **vCPU and ECU:** The [vCPU](#) (virtual central processing unit) is a unit of capacity used to compare DB instance classes. The Amazon EC2 Compute Unit (ECU) is a relative measure of the integer processing power of an Amazon EC2 instance, making it easier to compare CPU capacity across different instance classes.
- **Resource Sharing:** Amazon EC2 dedicates resources like CPU, memory, and instance storage to a specific instance, while sharing resources like the network and disk subsystem among instances. Instance types with high I/O performance are allocated a larger share of shared resources.

Purchasing Options for EC2 Instances:

There are several ways to pay for Amazon EC2 instances:

Purchasing Option	Description	Recommended Use Cases
On-Demand Instances	Pay per hour or per second (depending on instance) with no upfront payment or long-term commitments required.	Applications with short-term, spiky, or unpredictable workloads that cannot be interrupted; developing or testing applications for the first time.
Spot Instances	Request unused EC2 capacity, available at up to a 90% discount compared to On-Demand prices. The price (Spot Price) fluctuates based on long-term supply and demand.	Applications that have flexible start and end times; applications that are only feasible at very low compute prices; stateless, fault-tolerant, and flexible applications (e.g., big data, containerized workloads).
Reserved Instances (RIs)	Provides a significant discount (up to 72% or 75%) compared to On-Demand pricing. Requires a commitment to a specific instance configuration (type and Region) for a term of 1 or 3 years.	For longer-term savings and reserving capacity.
Savings Plans	Reduces costs by making a commitment to a consistent amount of usage (in USD per hour) for a term of 1 or 3 years.	Workloads with consistent and steady-state usage; customers who want flexibility across different instance types and compute solutions.
Dedicated Hosts	A physical EC2 server fully dedicated for your use. Billing is per host.	Helps reduce costs by allowing you to use existing server-bound software licenses (e.g., Windows Server, SQL Server); helps

		meet compliance requirements.
Dedicated Instances	Instances that run on single-tenant hardware, billed per instance.	Helps address compliance requirements.

Module 3 - Exploring Compute Services

▼ Exploring AWS compute options

Welcome back! At this point, you've learned how to use Amazon EC2 to provision compute resources in the AWS Cloud, and that's great! EC2 is one of the foundational services in AWS and understanding how to use it is key.

To quickly recap, EC2 instances are virtual machines that you can provision on AWS. EC2 is great for all sorts of use cases, from running basic web servers to high performance computing workloads, and everything in between. EC2 offers a high degree of control when it comes to your instances, but it also requires that you manage that fleet of instances over time.

EC2 is an unmanaged service, meaning that you have control over tasks like **patching, scaling**, and **managing the operating system**, while AWS manages the underlying infrastructure. Think back to the Shared Responsibility Model, where AWS is responsible for the security of the cloud, and you are responsible for security in the cloud.

On the other hand, AWS also offers **managed services**. Managed services shift more operational responsibilities to AWS—so you can focus on building your application and less on managing infrastructure. To help illustrate this, let's relate it back to the coffee shop.

Unmanaged services are like those high-end, **fully customizable** espresso machines. You get to choose the beans, grind them up to your desired consistency, tinker with every knob and lever to your liking, so you get that perfect cup of coffee. It's all about control. And it's a coffee enthusiast's dream! But it's also more work because you're on the hook for all the upkeep and maintenance for the machine as well.

Managed services, on the other hand, are more about **convenience**. Think of them like a coffee maker that uses pods. You just pop in a pod, choose your settings, press a button, and within moments, you've got a cup of coffee—no fuss, no hassle. Sure, it might not be as customizable as the espresso machine, but it saves you a lot of time and effort. What you choose really depends on what you're looking for.

AWS services span a range from unmanaged to managed, offering you varying degrees of control and convenience. Some services, like EC2, allow you to fine-tune everything. You're in charge of all configurations and management.

Then, there are **managed services**. Some examples of managed services that you've already learned about are **ELB, SNS, and SQS**. For managed services, you configure the service to meet your requirements, and then AWS makes sure it runs smoothly over time, with no server management required on your part. This idea of not managing any underlying infrastructure led to the rise of **serverless computing**.

Serverless means that you can't actually see or access the underlying infrastructure or instances that are hosting your application. Instead, all the management of the underlying environment from a **provisioning, scaling, high availability, and maintenance perspective** are **handled** for you. All you need to do is focus on your application.

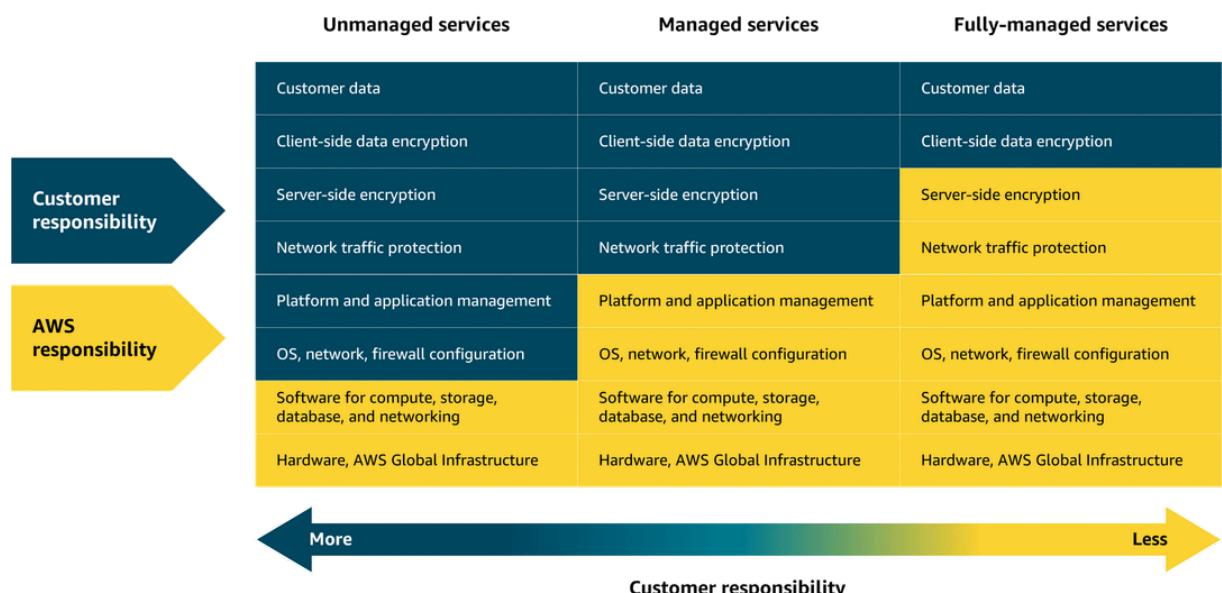
With a new understanding of terms like unmanaged, managed, and serverless, we're going to explore some of the other compute services that AWS has to offer. There are many services available, built for different use cases, and we will be covering some of these in the upcoming lessons.

AWS offers all these different compute services to give you options that cater to various workloads, requirements, and **levels of management**. The key is to recognize what your specific application needs are and choose a service that provides the right balance of customization and ease of use.

Decision Framework

Factor	Serverless (Lambda/Fargate)	Traditional (EC2/ECS)
Traffic Pattern	Sporadic, unpredictable	Steady, predictable
Duration	< 15 min	Any duration
State	Stateless	Stateful OK
Cost at Scale	High volume = expensive	High volume = cheaper
Latency	Cold starts possible	Consistent, low
Management	Zero ops	Requires maintenance
Startup Speed	Instant deploy	Minutes to provision
Scaling	Automatic, instant	Manual or slower

Sometimes you'll want to be the barista, brewing everything from scratch, and other times you just need that quick cup of coffee without all the fuss.



▼ AWS Lambda

Lambda is a **serverless compute** service that **runs code in response to events** without the need to provision or manage servers. It **automatically manages** the underlying infrastructure, scaling resources based on the volume of requests. You are **charged** only for the **compute time consumed**, down to the millisecond. Lambda handles execution, scaling, and resource allocation. You can optimize performance by configuring the appropriate **memory size** for your function.



Ok, time to dive into my favorite managed service—AWS Lambda. Lambda is a serverless compute service. It's also known as a **Function as a Service**.

Let's say you're building a crab classifier app where users upload a picture of a scuttling crustacean and then get notified when a classification has been made. After your app has been coded, you'll then need to deploy it onto infrastructure. This means provisioning servers, scaling up and down, and making sure everything is always available. I see you scratching your head and thinking...this sounds like a lot of work, Rudy. But don't get crabby!

You can use **Lambda to run code!** With Lambda, you don't need to think about servers or crab clusters. You create a Lambda function, put your code in there, configure a **trigger**, and your function runs in response to that trigger.

Going back to the crab classifier app, a simple trigger could be when a user **uploads a new picture** of a crab or when the app has determined the classification of an uploaded image. Triggers can be more complex, though. Like processing data in real-time from a stream or resizing an image into different resolutions. And the claw-some part is that you don't have to manage the environment. AWS handles it all for you!

This managed environment is **automatically scalable** and **highly available**. This means whether you get a single trigger or thousands, Lambda will scale up or down to meet the demand. Even better, AWS takes care of all the patching, updates, and **security**. You shrimpy focus on your code. And with that code, just be aware that the **maximum duration** of a Lambda function is **15 minutes**. Look, we're not being shellfish, but if you cannot **split your code** into 15-minute segments, Lambda might not be the best fit.

Lambda is great for quick, **event-driven processes**. For example, **handling website requests**, **processing batches of data**, and **generating expense reports**. Oh, and did I mention that Lambda **supports any programming language**? 'Cause it does! Lambda supports several languages by default through the use of **runtimes**. Languages like Java, Python, and Node.js. However, you can build your own custom runtime, too. A **runtime** provides a **language-specific environment** that relays **invocation events**, **context information**, and **responses** between Lambda and the function.

One last thing to mention is that Lambda integrates with other AWS services quite easily. This means you can build out your fully-fledged crab classifier app using a plethora of AWS services and still not have to worry about spinning servers up and down. You just have to worry about putting all those crab species into your database.

However, **Lambda** has **250MB** deployment package limit (10GB with container images). Also, AWS shuts down the execution environment after ~15 minutes of inactivity.

Thanks for watching.



AWS Step Functions

AWS Lambda Power Tuning

AWS Lambda Power Tuning is an [open-source tool](#) that uses **AWS Step Functions** to help you optimize your Lambda functions for either cost savings or improved performance.

<https://medium.com/ssense-tech/rightsizing-your-lambdas-lambda-power-tuning-compute-optimizer-f70dab1830ee>

<https://youtu.be/rpL77KDN92Q>

▼ Creating and AWS Lambda Function

Here's our architecture. We've got an **SQS** queue that **automatically triggers** a **Lambda** function whenever a **new message** is added to it. Even for this simple-looking workflow, we must ensure the Lambda function has the right **permissions** to **access** the **SQS** queue.

All right, let's jump into the SQS console. I've already created a queue, so let's add two text messages. OK, we are sending a message. This is a test message. We are going to send message. This is a test message again, send, go back here.

Next, Lambda, Lambda, Lambda. Well, one at least. I'm gonna use a blueprint to save us some time when creating a Lambda function. We'll choose **blueprint** over here. Go down, scroll, and we're gonna find the one that is related to SQS. Here we go.

I think I'll name the function Lambert after that sheepish lion. Here we have the **runtime** and **architecture** for the blueprint already set. The run time is Node.js in this case, but if you create your own function from scratch, you'll get to decide on the runtime. Supported run times include Java and Python, but you can use a **custom runtime** as well.

Now we need to set up an **execution role** to allow Lambert to **read messages** from our SQS queue. We can use the **Amazon SQS poller policy template**, which will allow Lambert to pull messages from our queue. We'll call this role *Demo_Lambert_Role*. So, select it. There we go. Demo. If we examine the blueprint code, you can see Lambert will be able to grab the message, log the ID and text of each message, and it tells us how many messages have been processed.

The last step is to set up the **SQS trigger** for our Lambda function. This trigger is already selected, so we'll just pick our queue from the list. We'll leave other settings as default and clickity-click to create function. Schweet! Lambda has been created, and you can see the code along with the enabled trigger.

Let's head back to the SQS console to check the messages we added earlier. Yep, they've already been processed. There's nothing left in the queue. Not sheepish after all, Lambert.

Back in our Lambda console, we can see **function metrics** by going to the **monitor tab** and then clicking **View CloudWatch logs**. We then click on the **log groups**. We click on the **function name**. We open up the **log stream**. And you can see, for the log stream, we have our two test **messages** that were **processed successfully** by Lambert, the Lambda function.

Time to do a quick manual test. Head back to the SQS console, select the queue, and then click the send and receive messages button. Let's type in test message. We then click send, and this will add the message to the queue. If we go back to the queue, you can see the messages are gone. Nowhere, *nada*. Why? Because Lambert, the Lambda function processed them that quickly.

OK, let's jump back to our Cloudwatch log streams. So, we're in the CloudWatch console here. We click on our Lambda function. We then scroll down to the log streams and just refresh to make sure. And you can see that all our messages were processed. These are the two messages from earlier, and this is the one that we just added.

There you go. All have been successfully processed by Lambert. And that's it, folks. That's a wrap.

▼ Containers and Orchestration in AWS

So far, we've talked about EC2 and Lambda, and there's another compute service we haven't talked about. Let's talk about **containers**!

Imagine you're a developer trying to deploy an application that's worked perfectly on your computer but fails everywhere else. That's frustrating, right? You might even have heard developers trying to debug this issue and saying to each other, "It works on my machine."

Containers solve this **portability problem** by providing a **consistent environment** that can be replicated **anywhere**. That's what containers do. They **package everything** your **application needs to run—code, runtime, dependencies, configuration**—into a **single, portable unit**. This creates a consistent environment, isolating your application from the underlying system and making it convenient to **deploy** and **scale** your **application anywhere**. **Containers** also provide benefits like **faster start times** and **improved resource efficiency**.

Now that you understand what containers are, let's talk about **hosting** them on AWS. You can manage containers on your own, where you place containers on top of a cluster of EC2 instances, but it's a lot of work. You'd have to deal with monitoring the health of the containers, starting and stopping them when needed, updating them, and managing the networking for them, and more. You can manage it, of course, but it would be complicated and easy to mess up. That's where container orchestration services come in.

Container orchestration services manage the **lifecycle of containers**, including starting, stopping, and running them across a cluster. These orchestration services **automatically scale containers** out when traffic increases—and scale back in when things calm down. This way, your application can handle spikes in demand without breaking a sweat. They also handle **recovery from failure, monitoring, and updates**, saving you tons of time and effort.

On AWS, we have two main container orchestration options: Amazon **ECS** and Amazon **EKS**. Let's break them down.

Amazon ECS

Amazon ECS stands for Amazon **Elastic Container Service**. It's ideal if you want something **streamlined and integrated**, but you can still **define your application's container images and resources**, such as **EC2 instance types and load balancers**. ECS **automatically manages** the containers and their infrastructure based on the **parameters** you set.



Amazon EKS

On the other hand, you have Amazon EKS, or **Amazon Elastic Kubernetes Service**. **Kubernetes** is an **open source platform** that **automates containerized application deployment, scaling, and management**. EKS makes it convenient to run **Kubernetes clusters** on AWS. It offers a lot of **control and flexibility**, especially for **large-scale or hybrid deployments**.



Amazon ECR

Amazon Elastic Container Registry (Amazon ECR) is where you can **store, manage, and deploy container images**. It supports container images that follow the Open Container Initiative (OCI) standards. You can push, pull, and manage images in your Amazon ECR repositories using standard container tooling and command line interfaces (CLIs).



Now, orchestration services need somewhere to get their containers from. That's where **Amazon Elastic Container Registry**, or Amazon ECR, comes in. ECR is a **fully managed container registry** that **stores your container images**. You build your containers that have your application and all of its dependencies bundled together. From there, a **container orchestration tool** can **pull the container image and deploy it**.

So, you've got your container image, and you've chosen an orchestration service. Now, let's focus on where your containers will actually run. AWS offers two main options for this: **Amazon EC2** and **AWS Fargate**.

With **EC2**, you **manage the virtual machines** that run your containers. With this option, you have **full control**, but you need to **manage the underlying infrastructure**.

On the other hand, **Fargate** is **serverless** and offers **efficiency** and **convenience**. With Fargate, AWS manages the servers, and you only need to worry about your containers. No need to manage a fleet for these containers to run on.

AWS Fargate

AWS Fargate is a **serverless compute engine** for **containers**. It works with both Amazon ECS and Amazon EKS. Fargate is a **container hosting platform**, unlike Amazon ECS and Amazon EKS, which are both container orchestration services.



When using Fargate, you do not need to provision or manage servers. Fargate manages your server infrastructure for you. You can focus more on innovating and developing your applications, and you pay only for the resources that are required to run your containers.

Lambda vs Fargate comparison table

Feature	AWS Lambda (Function-as-a-Service, FaaS)	AWS Fargate (Container-as-a-Service, CaaS)
Unit of Execution	A Function (a single piece of code).	A Container (an entire application/process).
Execution Model	Event-Driven. Runs only when triggered by an event.	Long-Running/Persistent. Runs continuously or for long batch jobs.
Execution Limit	15 minutes maximum per invocation.	No hard limit. Can run indefinitely (24/7).
Compute Scaling	Instantaneous/Massive. Scales by creating a new execution environment for <i>every single event</i> (up to thousands concurrently).	Measured/Predictable. Scales by launching or stopping whole containers based on CPU/memory metrics.
Resource Control	Limited. You only configure the Memory size , which indirectly controls the CPU.	Granular. You explicitly define the needed vCPU and Memory for the container.
Pricing	Based on Invocations and Execution Duration (rounded to the nearest millisecond).	Based on Allocated vCPU and Memory for the container's running time (rounded to the nearest second, with a one-minute minimum).
Best For	Short-lived, variable workloads like responding to API requests, file uploads, or database changes.	Long-running services like web servers, APIs that need to stay "warm," or complex batch processing.

Alright, let's walk through an example of how all these pieces fit together. First, you'll **upload** your **container images** to **ECR**, which is where your **images** get **stored** securely and are ready to be used. After that, you'll pick an **orchestration service** based on what you need, either **ECS** or **EKS**. Lastly, you'll choose your **compute option**, either **EC2** or **Fargate**.

With AWS, deploying and managing containers is convenient, efficient, and scalable—perfect for keeping your focus on where it matters most: on your application.

AWS has a set of tools for managing containers that fits into three categories: **orchestration**, **registry**, and **compute**.

Amazon ECS launch types:

- **Amazon ECS with Amazon EC2** is ideal for small-to-medium businesses that need full control over infrastructure. Suitable for custom applications requiring specific hardware or networking configurations, with the flexibility of Amazon EC2 and the simplicity of Amazon ECS.
- **Amazon ECS with AWS Fargate** is perfect for startups or small teams building web applications with variable traffic. It's a serverless option—no server management required—so teams can focus on development while Amazon ECS handles scaling and orchestration.

Amazon EKS launch types:

- **Amazon EKS with Amazon EC2:** This is best for enterprises needing full control over infrastructure. It offers deep customization of EC2 instances alongside Kubernetes scalability—ideal for complex, large-scale workloads.
- **Amazon EKS with AWS Fargate:** This is great for teams wanting Kubernetes flexibility without managing servers. It combines Kubernetes power with serverless simplicity, helping to scale applications quickly across various use cases.

Containers and VMs

A container packages your application with everything it needs to run, so it works the same on any computer. This helps to move, update, and manage. Containers are faster and lighter than virtual machines (VMs) because they share the host computer's operating system. VMs use a hypervisor to run full, separate operating systems, which makes them less resource-efficient and have longer startup times.

▼ Additional Compute Services

So far, we've covered compute options like EC2, Lambda, and container services like ECS. But wait...there's more! There are many purpose-built services you can use for specific use cases that can help you achieve your goals. Let's take a quick look at some of them!

Elastic Beanstalk

Elastic Beanstalk is a **fully managed** service that streamlines the **deployment**, **management**, and **scaling** of **web applications**. Developers can upload their code, and Elastic Beanstalk automatically handles the provisioning of infrastructure, scaling, load **balancing**, and application **health monitoring**. It supports various programming languages and frameworks, such as Java, .NET, Python, Node.js, Docker, and more. It provides **full control** over the underlying **AWS resources** while automating many **operational tasks**.



First up is **AWS Elastic Beanstalk**. This is a service that makes it easier to **deploy** and **manage applications** in EC2. Instead of building out the needed infrastructure, like the network, EC2 instances, scaling, and elastic load balancers by yourself, you can provide your application code and desired configurations to the Elastic Beanstalk service. Elastic Beanstalk then takes that information and **builds** out your **environment** for you. Elastic Beanstalk also makes it easy to **save environment configurations**, so they can be deployed again. You won't need to provision and manage all of these pieces separately, and you'll still have visibility and control of the underlying resources.

Good for: Deploying and managing web applications, RESTful APIs, mobile backend services, and microservices architectures, with automated scaling and simplified infrastructure management

AWS Batch

AWS Batch is a **fully managed** service that you can use to run batch computing workloads on AWS. It automatically schedules, manages, and scales compute resources for batch jobs, optimizing resource allocation based on job requirements.



Next up, let's talk about **AWS Batch**, a compute service designed for **heavy-duty tasks** like processing **massive datasets**, running **simulations**, or performing **complex calculations**. AWS Batch takes care of the **infrastructure management** for you. You won't need to worry about provisioning servers, scaling resources, or managing infrastructure. AWS Batch handles all of that, allowing you to focus on the important tasks like building your application or running your analysis. The service also scales **automatically**, distributing tasks across a fleet of compute resources like EC2 instances.

Good for: Processing large-scale, parallel workloads in areas like scientific computing, financial risk analysis, media transcoding, big data processing, machine learning training, and genomics research

Lightsail

Amazon **Lightsail** is a cloud service offering virtual private servers (VPSs), storage, databases, and networking at a **predictable monthly price**. It's ideal for small businesses, basic workloads, and developers seeking a straightforward AWS experience without the complexity of the full AWS Management Console.



Then there's **Amazon Lightsail**, which **simplifies web application hosting** by giving you a relatively easy, **cost-effective** solution for **running** specific types of **applications** and **websites**. It takes care of a lot of the complexity that comes with traditional web application hosting, and it's a great option if you want something **quick** and **easy** to manage!

Good for: Basic web applications, low-traffic websites, development and testing environments, small business websites, blogs, and learning cloud services

Outposts

AWS Outposts is a **fully managed hybrid cloud** solution that extends AWS infrastructure and services to **on-premises data centers**. It provides a consistent experience between on-premises and the AWS Cloud, offering compute, storage, and networking components.



Finally, let's talk about one more unique, purpose-built AWS service called **AWS Outposts**. This one is designed for organizations that need a **hybrid-cloud solution**. If you want to leverage the power of AWS while keeping **some** of your **infrastructure on premises**, Outposts is the answer. Outposts extends AWS services to your on-premises data center, giving you a consistent experience across both environments. You get to run **AWS services locally**, while still benefiting from **cloud computing**. This is perfect for meeting specific needs, like **low latency**, **data residency**, or integration for **hybrid deployments**.

Good for: Low-latency applications, data processing in remote locations, migrating and modernizing legacy applications, and meeting regulatory compliance or data residency requirements

Comparison Table - purpose build compute services

Feature	AWS Elastic Beanstalk	AWS Batch	Amazon Lightsail	AWS Outposts
Primary Purpose	Easier to deploy and manage applications in EC2.	Heavy-duty tasks like processing massive datasets, running simulations, or complex calculations.	Simplifies web application hosting and running specific types of applications and websites.	Extends AWS services to your on-premises data center for a hybrid-cloud solution.

User Input	Provide application code and desired configurations.	Focus on building your application or running your analysis.	A quick and easy solution for running specific applications/websites.	Leverage the power of AWS while keeping some infrastructure on premises .
Infrastructure Management	Builds out the environment for you (network, EC2, scaling, ELBs).	Takes care of infrastructure management (provisioning servers, scaling resources).	Takes care of a lot of the complexity that comes with traditional web application hosting.	Gives you a consistent experience across cloud and on-premises environments.
Scaling	Automatically handles scaling resources.	Scales automatically , distributing tasks across a fleet of compute resources like EC2 instances.	Not explicitly mentioned as a key feature, focus is on simplicity and being cost-effective .	Benefits from cloud computing while running services locally .
Key Benefit/Output	Easy to save and redeploy environment configurations .	Handles all infrastructure, allowing you to focus on important tasks .	Relatively easy and cost-effective solution; quick and easy to manage.	Perfect for meeting specific needs: low latency, data residency, or hybrid deployments .
Control/Visibility	You still have visibility and control of the underlying resources .	Handles all infrastructure management, allowing you to focus on the analysis .	Focuses on reducing complexity for web hosting.	Run AWS services locally .
Good For	Deploying and managing web applications, RESTful APIs, mobile backend services, and microservices architectures, with automated scaling and simplified infrastructure management.	Processing large-scale, parallel workloads in areas like scientific computing, financial risk analysis, media transcoding, big data processing, machine learning training, and genomics research.	Basic web applications, low-traffic websites, development and testing environments, small business websites, blogs, and learning cloud services.	Low-latency applications, data processing in remote locations, migrating and modernizing legacy applications, and meeting regulatory compliance or data residency requirements.

Module 4 - Going Global

▼ Introduction to Going Global

The coffee shop is thriving. In fact, the shop has gained quite a following, and now we're thinking it's time to expand. That's right. Breaking news: the coffee shop is going international.

As the shop plans its expansion, there are a few considerations to keep in mind. First, we need to decide where to open new locations. We want to reach coffee lovers in different parts of the world, but we also need to consider factors like local demand, regulations, and costs. This is like how in AWS, when you are expanding globally, there are a number of factors to consider when selecting AWS Regions. In these upcoming lessons, you'll learn all about what goes into choosing a Region or set of Regions.

The next step of our coffee shop expansion plan is that we want to open some lightweight, smaller footprint versions of our shop called coffee carts. We'll set these up in places like farmers markets, airports, and event venues. They won't serve every drink or item on the menu, but they can provide the most popular items quickly and efficiently. These coffee carts are similar to how **AWS edge locations** work. Edge locations offer fast, localized delivery of the **most frequently accessed content**. They **cache** things like images, videos, and other **assets** and **resources**, allowing users to get the content they need **quickly**, without waiting for it to be retrieved from a central location. You'll learn more about these soon.

Finally, our shop wants to **standardize** and **automate processes** so we can keep our customer satisfaction consistent. No matter which shop or coffee cart the customer visits, we want them to get the same great experience. We'll train staff on the same recipes and use smart coffee machines that can be programmed remotely and replicated to all the different locations, making sure that a cappuccino in Stockholm tastes just like one in Seattle. Similarly, AWS has infrastructure in place to help businesses **scale responsibly and consistently**. In this section, you'll explore how to achieve consistent deployments across environments and global deployments using **infrastructure as code**, or **IaC**, specifically focusing on **AWS CloudFormation**.

By the end of this section, you'll have a thorough understanding of the ins and outs of AWS Global Infrastructure, helping you to achieve robust, scalable, and globally available applications. So, grab your cup of coffee, and let's dive in.

▼ Choosing AWS Regions

One of the best parts about the AWS Global Infrastructure is that you have lots of options when it comes to which AWS Region, or Regions, you deploy your resources in. Before we talk about what factors into this business decision, I want to touch on an important security aspect of AWS Regions. Each Region is isolated from every other Region, in the sense that no data goes in or out of your environment in that **Region** without you explicitly granting **permission** for that **data** to be **moved**.

This is a good thing! Depending on the type of business you are dealing with and where you operate, you might have to adhere to specific compliance **regulations** that require your **data** to remain in one **geographical area**. For example, if you're working with financial information in Frankfurt, local data governance laws state that this financial data cannot leave Germany. The data that is stored in an AWS Region is subject to the **local laws and statutes** of the country where the Region lives. Which is actually our first of four considerations when choosing a region: compliance.

Compliance

Before any of the other factors, you must first look at your **compliance requirements**. Do you have a requirement that your data must live in UK boundaries? Then you should choose the London Region. The choice is pretty straightforward. None of the other options really matter. Or, let's say you must run inside of Chinese borders. Well then, you should choose one of our Regions located in China. Most businesses are not governed by such strict regulations. So, if you don't have a compliance or regulatory control that dictates your Region, then you can look at the other factors.

Proximity

The second factor is **proximity**. How close you are to your **customer base** is a major factor. If most of your customers live in Singapore, consider running out of the Singapore Region. You can certainly run out of Virginia, but the time it takes for the information to be sent, or **latency**, between the US and Singapore is always going to be a factor.

Feature Availability

For number three we have **feature availability**. Sometimes the closest Region might not have all of the AWS features you want. Here's one of the cool things about AWS. We're constantly innovating on behalf of our customers. Every year, AWS releases lots of **new features** and **products** specifically to answer customer requests and needs. These features are **rolled out to regions over time**, so that is also a consideration.

For example, AWS GovCloud Regions are specifically designed to meet the compliance and security requirements of US government agencies and their contractors.

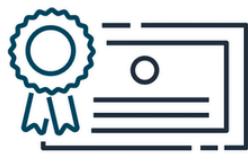
Pricing

Finally factor number four is **pricing**. Even when the services and features are equal from one Region to the next, some locations are **more cost effective** to operate in than others. Things like **local tax** structure and **energy costs** factor into the equation. AWS has a very transparent, granular pricing that we'll continue to discuss in this training. But know that each Region has different numbers for pricing.

Some Regions have lower operational costs than others. These operational costs can impact the overall expenses for hosting applications and services. Some Regions might offer tax incentives or have lower tax rates, which can affect customer pricing.

So, to wrap up, you have a lot of options in terms of where to deploy your resources. Keep these four key factors in mind when choosing a Region: compliance, proximity, feature availability, and pricing.

1: Compliance



2: Proximity



3: Features



4: Pricing



▼ Diving Deeper into AWS Global Infrastructure

At this point, you've seen how user proximity, regulatory compliance, service availability, and even pricing can play a factor in selecting an AWS Region. When it comes to infrastructure, you also want to plan for long-term stability and less or virtually no down time for your users. So, if your infrastructure has an interruption, you can switch to redundant or backup infrastructure seamlessly. This is called building **redundant architectures**.

One method for redundancy is an architecture that uses multiple Availability Zones, or **AZs**. In a multi-AZ architecture, if an AZ has an interruption, no worries. Your application will automatically switch over to the backup AZ you have configured. Even better, if you set it up correctly, your customers won't even notice a difference. 'Cause let's be honest, nobody wants their favorite meme site or coffee shop app to go down! Additionally, **multi-AZ architectures** can assist with **quicker disaster recovery**, improved business **continuity**, **lower latency**, and **compliance**.

But as with anything in the world, there are multiple ways to peel an orange. In fact, you can go one step further and deploy your application in multiple AWS Regions. So, if a whole Region experiences an interruption, you can failover to another one. Orange you glad you chose AWS as your cloud provider?

Some of you might say, "Wait, wait, wait, Rudy, multi-Region and multi-AZ deployments? This sounds like some sort of pinball machine." Well, yeah, it is exactly like a pinball machine but with a multi-ball bonus. It can be difficult to juggle multiple pinballs initially. They are moving in different directions, and you might stress about keeping track of all of them. However, once you come up with a strategy and you get some experience, it gets much easier.

So, don't worry too much about perfecting your AWS global infrastructure right off the bat. Just like pinball, planning and executing multi-AZ and multi-Region deployments gets easier after you've practiced a bit. Who knows, some day you might even get an AWS infrastructure high score! Just kidding, I don't think that's a thing. Is that a thing? Oh, it might be a thing. Oh wait..

Oh, looks like Rudy's Rhubarb Refresher, trademark, is so popular that people are making memes of it! However, images are loading slowly for some people on the app. Let's fix that by utilizing Amazon CloudFront. **CloudFront** is a **content delivery network**, and it's designed to **serve content as close to users** as possible. This content can be images, data, videos, applications, APIs—and, in our case—memes.

CloudFront uses **Edge locations**, which are part of our worldwide **Amazon Global Edge Network**. These edge locations are actually **separate from Regions** and are specifically designed to **accelerate content delivery**. Edge locations host other AWS services, like **AWS Global Accelerator** and **Amazon Route 53**. Route 53 is a Domain Name System, or **DNS**, that **routes end users to internet applications**. Essentially, it converts human-readable URLs to machine-readable IP addresses. 'Cause trust me, you don't want to have to remember IP addresses by heart.

And don't forget about **AWS Outposts**. Say you have a need. And that need is for **speed**. More speed than even a **Region paired with CloudFront** can achieve. This is where Outposts comes in handy. As you might recall, Outposts essentially makes it possible for you to run AWS services on-premises. Look, keep these **multi-deployment concepts** and edge services in your brain as you continue to explore AWS global infrastructure.

AWS Regions are physical locations around the world where AWS has multiple Availability Zones. Edge locations are located outside of AWS Regions and cache frequently accessed content.

▼ Infrastructure and Automation

By now, you know that in order to manage AWS resources, you have to interact with AWS APIs. You've learned how to do this using the Management Console, the CLI, and SDKs. But what happens when you need to create and manage multiple resources, possibly across multiple AWS Regions or multiple accounts, and you want to make sure everything is **consistent** and **repeatable**?

Say you have resources in Region A, and you want to launch them in Region B for high availability. You could set everything up manually by clicking through the console or running commands, remembering all of your configurations as you go along, but that's slow, error-prone, and hard to reproduce. Or you can use **automation** and this is where the concept of **infrastructure as code**, or IaC, comes in.

You can use IaC to define your infrastructure in a file, almost like a **blueprint** for your **AWS architecture**. You can then use tools or services to automatically build and configure your resources based on your blueprint specifications. You can deploy the same setup multiple times without variation, and you can **track changes** to your infrastructure more effectively using **source control**.

AWS CloudFormation is an IaC service that you can use to define a wide variety of AWS resources in a **declarative way** by creating **text-based documents** called **CloudFormation templates**. You can define what resources you want to build without specifying the details of exactly how to build it. CloudFormation parses the template and then provisions all of the resources you defined, calling the needed AWS APIs in the background to make it all happen.

When you deploy the same template in multiple accounts or multiple Regions, identical environments are created across them. There's less room for human error, because it's a totally automated process. So now, instead of manually setting up resources in Region B to match Region A, you can create a CloudFormation template that **defines everything your infrastructure needs**. With a **single command**, AWS provisions those resources exactly as defined.

And, hey, look at that, you've saved time, reduced the margin for error, and made your architecture more resilient. Nice work. You're really starting to catch on to all of this cloud practitioner stuff.

Module 5 - Networking

▼ introduction to Networking

Looks like things are really moving forward in our coffee shop. Though, we had a few eager customers go full steam ahead and yell their orders at the baristas! Tsk tsk.

They should be politely asking the cashiers instead. This does bring up an interesting point, however. It just doesn't make sense to allow every customer to be able to interact with the baristas in the back. After all, our baristas need to stay focused on crafting caffeinated beverages. So, what do we do?

Well, we need to limit access to the baristas, and let customers interact only with our cashiers. And wouldn't you know it, we can use AWS networking to accomplish that. Specifically, a networking concept called **Amazon Virtual Private Cloud**, or **VPC**. VPCs help you provision a **logically isolated section** of the AWS Cloud.

In this virtual network, you can launch whatever resources you decide on. More importantly, these resources can be **public** or **private**. Public-facing resources have access to the internet, whereas private resources do not have **internet access**.

This is perfect for our coffee shop. We can make our cashiers publicly-accessible so they can interact with our customers to take their order and process payments. We can then prevent our customers from interacting directly with the baristas by making them private resources. This means, hey, I can focus on making drinks! Ahhhh, refreshing.

Okay, time to dive into more networking concepts. Good luck!

Amazon Virtual Private Cloud (Amazon VPC)

An Amazon VPC lets you provision a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you define.

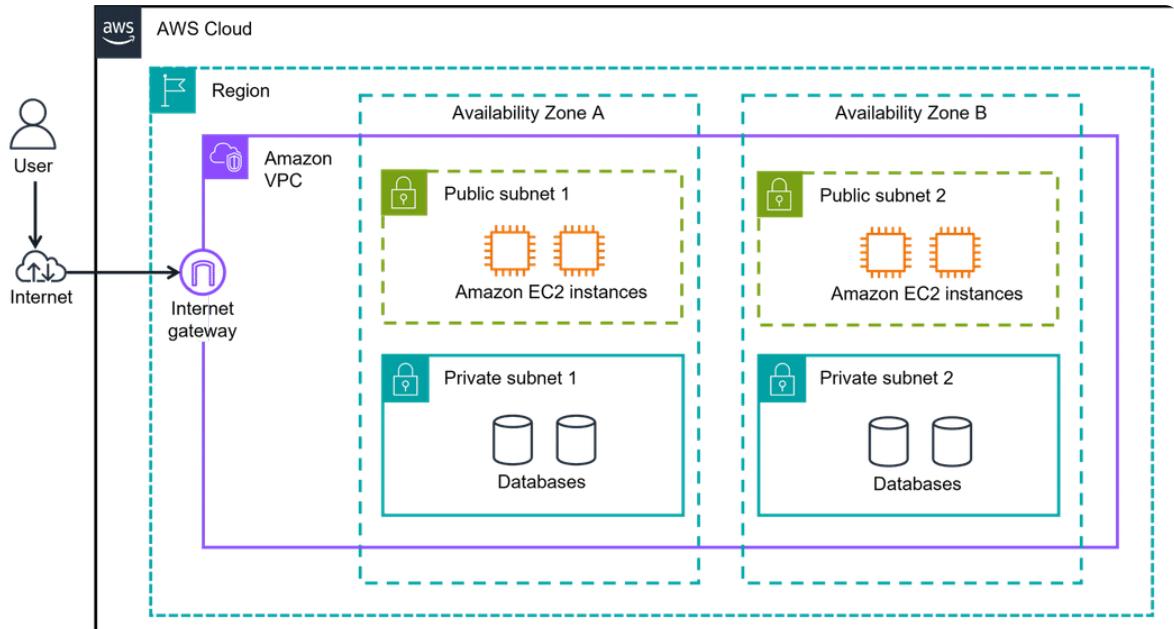
Subnet

Subnets are used to organize your resources and can be made publicly or privately accessible. A private subnet is commonly used to contain resources like a database storing customer or transactional information. A public subnet is commonly used for resources like a customer-facing website.

Feature	Virtual Private Cloud (VPC)	Subnet
Scope/Size	The entire virtual network in a Region.	A segment of the VPC's IP address range.
IP Addresses	Defines the total range of all private IP addresses that can be used (e.g., 10.0.0.0/16).	Defines a smaller range of usable IP addresses taken from the VPC's range (e.g., 10.0.1.0/24).
Isolation Level	Logically isolates your network from all other customers in the public cloud.	Isolates resources within the VPC for organization and security (e.g., separating web servers from databases).
Geographic Span	Spans an entire AWS Region.	Restricted to a single Availability Zone (AZ) within a Region. This is key for high availability.
Key Components	Contains Subnets, Route Tables, Internet Gateways, NAT Gateways, and Security Groups.	Contains your actual resources (EC2 instances, RDS databases, etc.) and is linked to a Route Table.

Networking components: Understanding connections through diagrams

If you are new to IT or cloud computing, you might not have worked with architectural diagrams before. A diagram is, simply put, a schematic or map of your network in the AWS Cloud. It can provide a visual of how users or applications access services, resources, or data. With a quick glance, you can see if the network was built for redundancy, security, and even scalability. It can also serve as a blueprint so you don't forget important connections when building your solutions.



Amazon VPC is a solid box, and it represents your isolated, logically segmented network within AWS. A VPC helps you to control your network resources and security.

Subnets are essentially segments of your VPC, allowing you to divide your VPC into smaller, manageable sections. A subnet is a range of IP addresses in your VPC.

Private subnets are designed to isolate resources that shouldn't be directly exposed to the public internet. In diagrams, they are illustrated with solid boxes.

Public subnets are designed to provide direct internet access to resources placed inside them. To allow access, they are connected with an internet gateway.

▼ Organizing AWS Cloud Resources

To reiterate, a VPC, or virtual private cloud, is essentially **your own private network in AWS**.

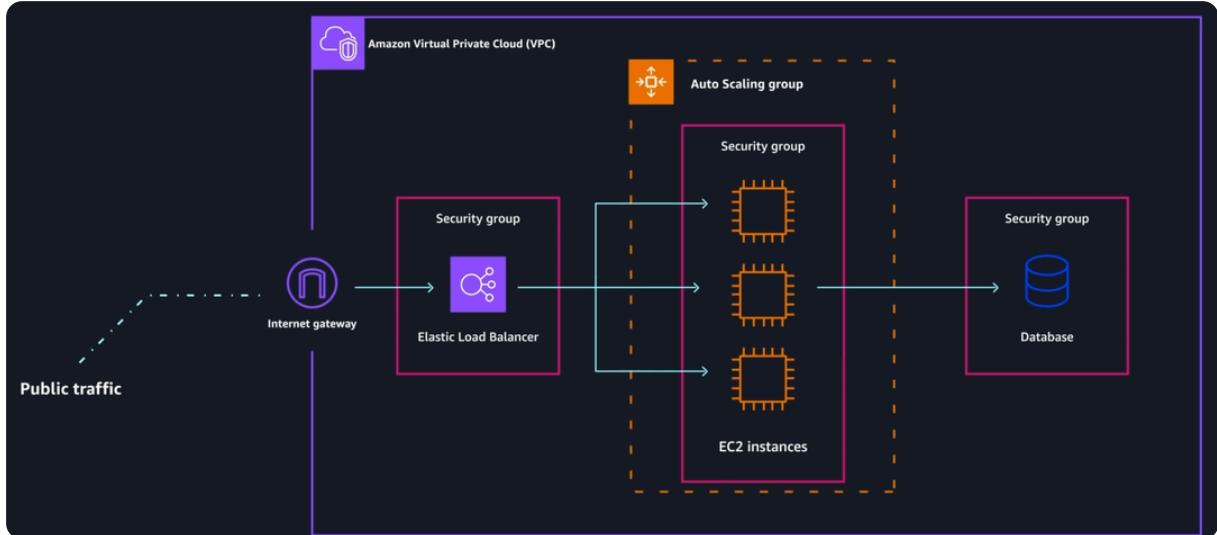
When you use a VPC, you can define your **private IP range** for your AWS resources and place things, like EC2 instances and elastic load balancers, inside of your VPC.

Now, you don't just go throwing your resources into one big VPC network space and then move on. Instead, you place them into different **specific subnets**. Subnets are chunks of IP addresses in your VPC that you can use to **group resources together**. Subnets, along with **networking rules** that we will cover later, control whether resources are either **publicly** or **privately** available.

This idea of public compared to private access to resources is super important. For some VPCs, you might have **internet-facing resources** that the public should be able to reach, like a **public website** or a **load balancer**, for example.

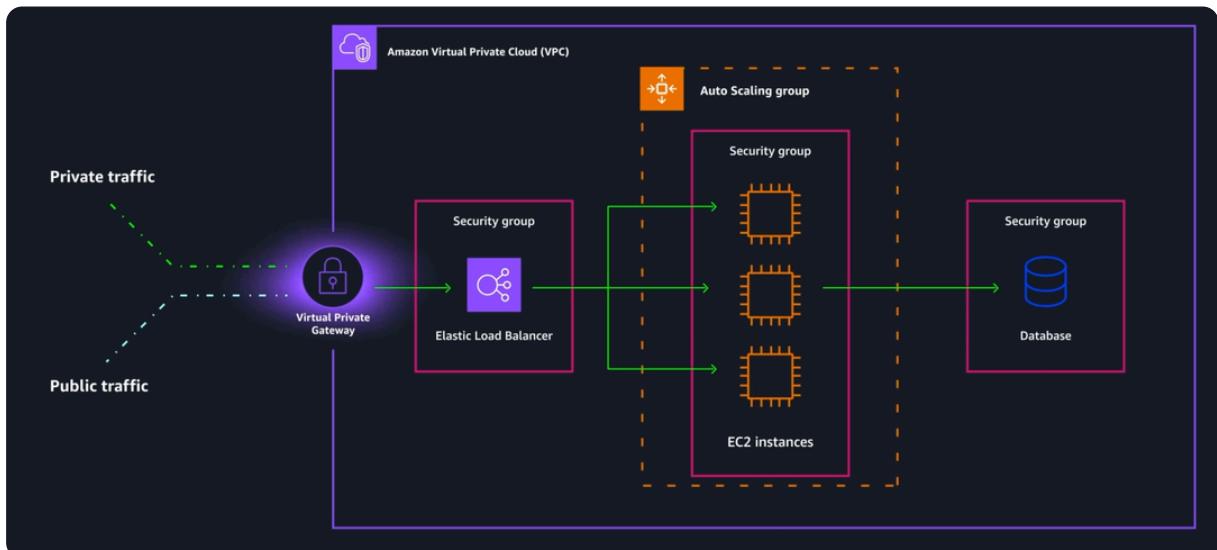
However, in other scenarios, you might have resources that you only want to be reachable if someone is logged into your private network. This might be internal services, like an HR application or a **backend database**.

First let's talk about public-facing resources. To allow traffic from the public internet to flow into and out of your VPC, you must attach what is called an **internet gateway** to your VPC. An internet gateway is like a doorway that is open to the public.



Think of the coffee shop. Without a front door, the customers couldn't get in and order their coffee. So, you install an entrance, and the people can enter and exit when coming and going from the shop. The front door in this example is like an internet gateway. Without it, no one can reach the resources placed inside of your VPC.

Next, let's talk about a VPC with all internal private resources, when you don't want an internet gateway attached to your VPC. Instead, you want a private gateway that only allows people in if they are coming from an approved network, not the public internet. This private doorway is called a **virtual private gateway**, and it allows you to create a **VPN connection** between a private network, like your on-premises data center or internal corporate network to your VPC.



To relate this back to the coffee shop, this would be like if the coffee shop was located inside of a private corporate office building. If I want to go get coffee, I have to badge in to verify my identity. Then I can access the internal coffee shop that only people with access to the building can use. So, if you want to establish an encrypted **VPN connection** to your private internal AWS resources, you need to attach a **virtual private gateway** to your VPC.

Now, something to note about the coffee shop in the private corporate office building is that this office building is shared by multiple companies, and there are a lot of people who work here. Even though I have special access to the coffee shop, I still might have to wait for the elevator, navigate crowded hallways, or stand in line.

This is similar to how a VPN works. While it provides a **secure connection**, it still **routes your traffic** through a **shared network**, which can sometimes lead to **slowdowns**, especially when many people are using it at the same time. It's not that the VPN itself is slow or a bad option, but rather you may need **higher bandwidth** or a **dedicated line** in certain scenarios.

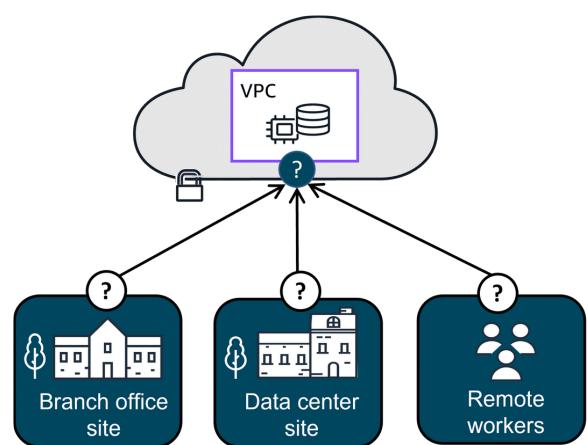
Now, if I had a direct, super-secret magic doorway that led from the studio straight into the coffee shop, I'd bypass any congestion and have a **reliable, high throughput**, coffee **connection** at any time. That sounds pretty nice. This is a similar idea behind wanting **dedicated private connection** to AWS.

▼ More Ways to Connect to the AWS Cloud

Connecting to the AWS Cloud

With so many different types of networks, on-premises datacenters, and remote workers, companies need a wide range of ways to connect to the AWS Cloud. In the following section, you will learn four ways to connect to the AWS Cloud:

- AWS Client VPN
- AWS Site-to-Site VPN
- AWS PrivateLink
- AWS Direct Connect



Securely connect a remote workforce to AWS Cloud resources -**AWS Client VPN**

AWS Client VPN

Imagine a company with a recent acquisition needing to securely connect their new remote workforce to their AWS Cloud resources. Even the largest companies with worldwide remote workers can quickly scale up and connect to the AWS Cloud. That's where **AWS Client VPN** can help.



AWS Client VPN connects your remote workforce to AWS or on-premises with a VPN.

AWS Client VPN is a networking service you can use to connect your remote workers and on-premises networks to the cloud. It is a **fully managed, elastic** VPN service that **automatically scales** up or down based on user demand. Because it is a **cloud VPN solution**, you don't need to install and manage hardware or try to estimate how many remote users to support at one time.

Benefits: AWS Client VPN provides advanced authentication, remote access. It is elastic and fully managed.

Use case: It can be used to quickly scale remote-worker access.

Client VPN, a managed VPN service, provides secure access to AWS resources and on-premises networks from anywhere. It uses an **OpenVPN-based client**, and it works with global Regions by using the AWS global network.

Securely connect sites to other sites - AWS Site-to-Site VPN

AWS Site-to-Site VPN

Some companies might want to establish secure, encrypted connections between their on-premises networks like data centers or branch offices and their resources in their Amazon VPC. That's where Site-to-Site VPN can help.



AWS Site-to-Site VPN is an encrypted network connection to your Amazon VPCs.

Site-to-Site VPN creates a secure connection between your data center or branch offices and your AWS Cloud resources.

Benefits: Site-to-Site VPN provides high availability, secure and private sessions, and accelerates applications. AWS Site-to-Site VPN accelerates applications primarily by routing traffic through the highly optimized **AWS global network backbone by using AWS Global Accelerator** instead of relying solely on the public internet.

Use cases: It can be used for application migration and secure communication between remote locations.

Securely connect resources, even in other VPCs - AWS PrivateLink

AWS PrivateLink

Other companies sometimes need the flexibility to privately connect to resources in other cloud providers as though they were in their own VPC. They need a way to **communicate** with these resources and don't want the hassle of setting up gateways or site-to-site VPNs. That's where AWS PrivateLink can help.



AWS PrivateLink connects your VPC privately to services and resources as though they were in your VPC.

AWS PrivateLink is a highly **available, scalable** technology that you can use to **privately connect your VPC to services and resources** as if they were in your VPC. You do not need to use an internet gateway, NAT device, public IP address, Direct Connect connection, or AWS Site-to-Site VPN connection to allow communication with AWS services or resources from your private subnets. Instead, you **control the specific API endpoints, sites, services, and resources** that are reachable from your VPC. It achieves this by making the external service appear as if it is **hosted directly within your own private VPC. (never leaves the AWS network)**

Benefits: AWS PrivateLink helps you secure your traffic and connect with simplified management rules.

Use case: It is used for connecting your clients in your VPC to resources, other VPCs, and endpoints.

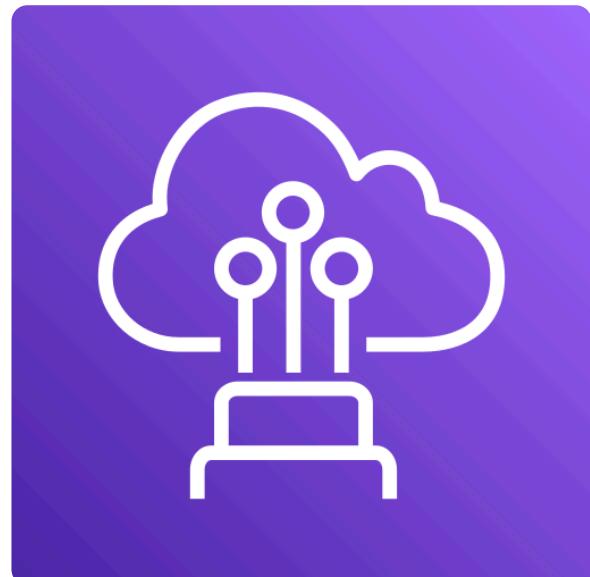
Even though the preceding connections are highly available and scalable, traffic jams are possible because you're using the **same connection** as **other clients**. That's why for some use cases, you might need a dedicated private connection with a lot of bandwidth.

Feature	Simple Explanation	Benefit
No Public Internet	Traffic stays entirely within Amazon's private, high-speed network.	Enhanced Security: Dramatically reduces exposure to threats like DDoS or data interception.
VPC Endpoint	You place a small network interface	Simplified Access: Your application uses a private IP

	(called a VPC Endpoint) in your private Subnet.	address on that endpoint to talk to the external service. You don't need to manage complex routing, NAT, or Internet Gateways.
Service-Specific	The connection is only to a single, specific service (e.g., an API), not the provider's entire network.	Tighter Control: Provides a smaller attack surface and simplified security group management.

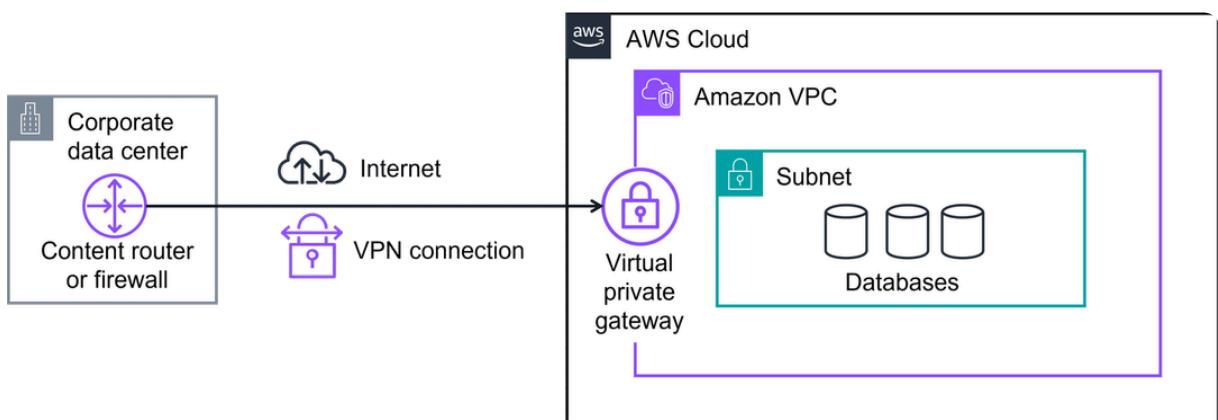
AWS Direct Connect

With AWS, you can achieve that using a service called **AWS Direct Connect**. Direct Connect lets you establish a completely private, dedicated **fiber connection** from **your data center to AWS**. It ensures both **security** and **consistent high performance**. You work with a **Direct Connect partner** in your **area** to establish this connection because, like my magic doorway, Direct Connect provides a **physical line** that connects your network to your Amazon VPC. This can help you meet **regulatory** and **compliance needs**, as well as sidestep any potential **bandwidth issues**.



Thanks for listening. I'm gonna hang out here and keep ordering magic drinks from my magic door. See ya.

With a virtual private gateway, you can establish a VPN connection between your VPC and a private network, such as an on-premises data center or internal corporate network. A virtual private gateway allows traffic into the VPC only if it is coming from an **approved network**.



Latency-sensitive applications

Direct Connect **bypasses** the **internet** and provides a **consistent, low-latency network** experience. This makes it ideal for applications like video streaming and other real-time applications that require high performance.

Large-scale data migration or transfer

Direct Connect helps ensure smooth and **reliable** data transfers at **massive scale** for **real-time analysis**, **rapid data backup**, or **broadcast media** processing.

Hybrid cloud architectures

You can use Direct Connect to link your AWS and on-premises networks to build applications that span environments without compromising performance.

PrivateLink vs Direct Connect

Feature	AWS PrivateLink	AWS Direct Connect (DX)
What it is	A logical connection (a virtual endpoint) to an AWS-hosted service or resource.	A physical connection (a dedicated fiber optic line) from your on-premises data center to an AWS location.
Scope of Connection	VPC-to-Service (or VPC-to-VPC).	On-Premises-to-AWS Network.
Traffic Flow	Traffic never leaves the AWS private network backbone ; it stays entirely in the cloud.	Traffic leaves your physical premises and enters the AWS network over a dedicated, private line.
Goal	Security & Simplification. To make a remote service (like S3, a SaaS API, or another company's VPC) look like it is local to your private VPC.	Performance & Reliability. To get high-bandwidth, stable, and low-latency access to your AWS resources by bypassing the public internet .
Use Case Example	Accessing a public AWS service (like S3) or a vendor's API using a private IP address from your EC2 instance.	Accessing your VPC or Transit Gateway from your corporate data center for hybrid cloud operations.

Site-to-Site VPN vs. PrivateLink vs. Direct Connect

Feature	AWS Site-to-Site VPN	AWS Direct Connect (DX)	AWS PrivateLink
Network Path	Public Internet (Encrypted Tunnel)	Private, Dedicated Fiber	AWS Global Backbone (Internal, Private)
Connection Type	Logical/Virtual (IPsec Tunnel)	Physical (Dedicated Fiber Cable)	Virtual Interface/Endpoint (ENI)
Core Goal			

	Cost-effective Hybrid Cloud	High-throughput, low-latency Hybrid Cloud	Secure, private access to a Specific Service
Security	Encrypted via IPsec	Inherently private, but data is unencrypted by default (often coupled with VPN for encryption).	Inherently private and isolated within AWS network.
Performance	Unpredictable, limited to 1.25 Gbps per tunnel (up to 4 Gbps via ECMP).	Consistent and Predictable ; up to 100 Gbps dedicated bandwidth.	High throughput, very low latency for service access.
Use Case	Backup connection, low-volume hybrid traffic, rapid setup.	Large data migration, mission-critical voice/video, constant high traffic.	Connecting your VPC to a SaaS product, S3, or DynamoDB privately.

Additional gateway services

AWS Transit Gateway

AWS Transit Gateway is used to connect your Amazon **VPCs** and **on-premises networks** through a **central hub**. As your cloud infrastructure expands globally, inter-Region peering connects transit gateways together using the AWS Global Infrastructure. To learn more, refer to [AWS Transit Gateways](#)(opens in a new tab).

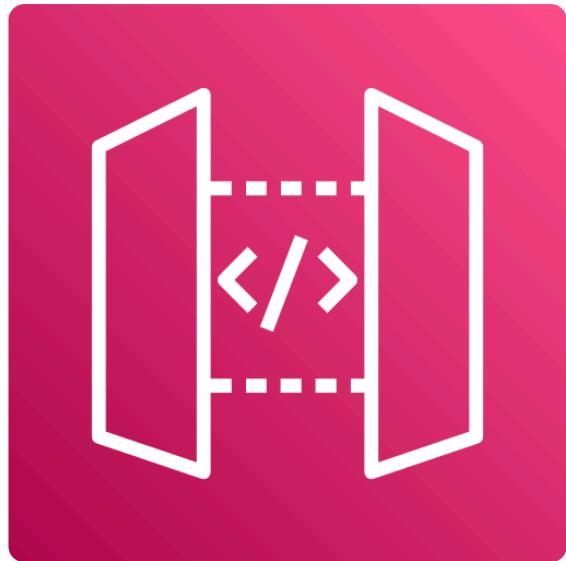


Network Address Translation (NAT) Gateway

A NAT gateway is a NAT service. You can use a NAT gateway so that instances in a private subnet can connect to services outside your VPC but external services can't initiate a connection with those instances. To learn more, refer to [NAT gateway](#)(opens in a new tab).

Amazon API Gateway

You learned about Application Programming Interface (API)s earlier. Quick refresher, an API defines how different software systems can interact and communicate with each other. The Amazon API Gateway is an AWS service for creating, publishing, maintaining, monitoring, and **securing APIs** at any scale. To learn more, refer to [Amazon API Gateway](#) ([opens in a new tab](#)).



▼ Subnet, Security Groups. and Network Access Control Lists

Welcome to your virtual private cloud, or VPC. You can think of it as a hardened fortress, where nothing goes in or out without explicit permission via the appropriate gateways. But that only covers perimeter, and that's only one part of network security that you should be focusing on as part of your IT strategy. AWS has a wide range of tools that cover every layer of security: **Firewalls**, distributed denial-of-service, or **DDoS prevention, encryption**, and much more. We're gonna talk about these a bit later in the security section.

Today, I wanna talk about a few aspects of network hardening looking at what happens inside the VPC. One of the main reasons to use subnets in a VPC is to control access to the gateways. The public subnets have access to the internet gateway, the private subnets do not. But **subnets** can also **control traffic permissions**. Packets are **messages** from the internet, and every packet that crosses the subnet boundaries gets checked against something called a **network access control list**, or **network ACL**. This check is to see if the packet has **permissions** to either leave or enter the subnet, based on **who it was sent from** and **how** it's trying to **communicate**.

You can think of network ACLs as passport control officers. If you're on the approved list, you get through. If you're not on the list, or if you're explicitly on the do-not-enter list, then you get blocked. **Network ACLs check traffic going into and leaving a subnet**. The list gets checked on your way in and on the way out. And just because traffic is let in doesn't necessarily mean they're gonna let responses back out. Only **explicitly approved traffic** can be sent on its way.

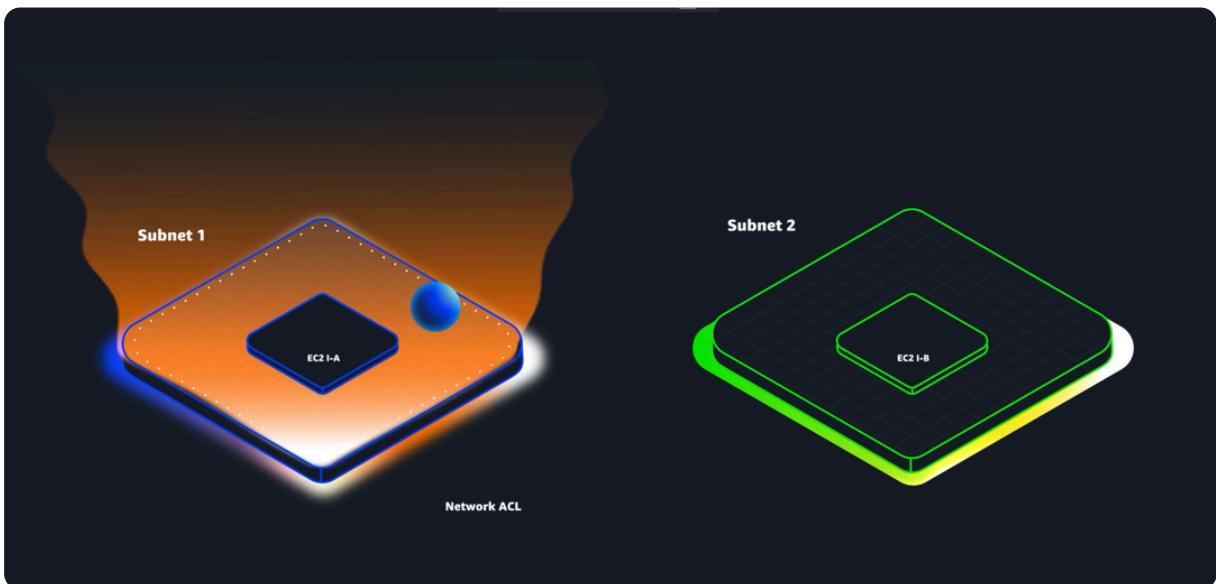
Now, this sounds like great security, but it doesn't answer all of the network control issues. A network ACL only gets to evaluate a packet if it crosses a subnet boundary, in or out. It doesn't evaluate if a packet can reach a specific EC2 instance or not. Sometimes, you will have multiple **EC2 instances** in the same subnet, but they might have different rules around who can send them messages and what **port** those **messages** are **allowed** to be **sent to**. So, you need **instance-level network security**, as well.

To solve instance-level access questions, we introduce **security groups**. Every EC2 instance, when it's launched, **automatically** comes with a **security group**. And by **default**, the **security group does not allow any traffic into the instance** at all. All **ports** are **blocked**. All **IP addresses** sending packets are **blocked**. That's very secure, but perhaps not very useful if you want an instance to actually **accept traffic** from the **outside**, like, say, a message from a frontend instance or a message from the Internet. So, obviously, you can modify the security group to accept a specific type of traffic.

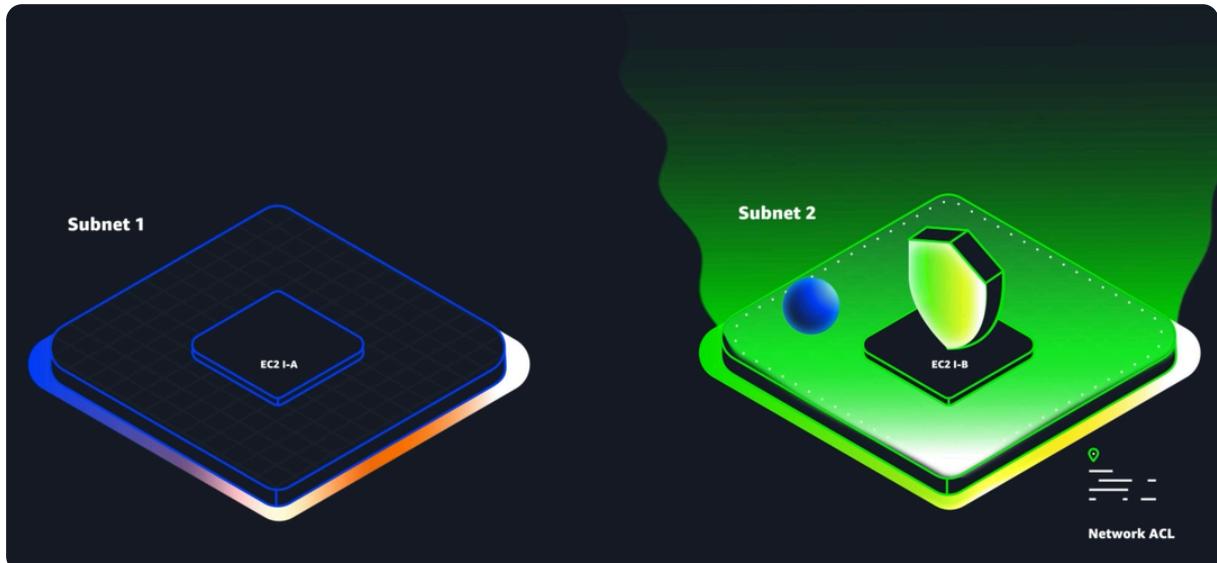
In the case of a **website**, you want web-based traffic like **HTTPS** to be **accepted** but not other types of traffic. If network ACLs are a passport control, a security group is like the doorman at your building, the building being the EC2 instance, in this case. The doorman will check a list to ensure that someone is **allowed** to **enter** the building but **won't** bother to **check** the list on the way **out**. With security groups, you **allow specific traffic in**, and by **default**, **all traffic is allowed out**.

I just described two different things letting good packets in and keeping bad packets out. The key difference between a security group and a network ACL is the **security group is stateful**. That means, as we talked about, it has some kind of a **memory** when it comes to **who to allow in or out**. And the network **ACL is stateless**, which remembers nothing and **checks every single packet** that crosses its border regardless of any circumstances. You know, this metaphor is important to understand. So, I wanna illustrate the round trip of a packet, as it goes from one instance to another instance in a different subnet. Now, this traffic management, it doesn't care about the contents of the packet itself. In fact, it doesn't even open the envelope. It can't. All it can do is check to see if the **sender** is on the **approved list**.

Alright, let's start with a couple instances, and we wanna send a packet from instance A to instance B in a different subnet, same VPC, different subnets. So, instance A sends the packet. Now, the first thing that happens is that packet meets the boundary of the security group of instance A. By default, all outbound traffic is allowed from a security group. So, you can walk right by the doorman and leave. Cool. Right. The **packet made it past the security group of instance A**.



Now, it has to leave the **subnet boundary**. At the boundary, the packet must now make it through passport control, the **network ACL**. The network ACL doesn't care about what the security group allowed. It has its own list of who can pass and who can't. If the **traffic address is allowed**, you can keep going on your journey, which it is. It's left **Instance A** and **Subnet 1** entirely, so now it must go through to **Subnet 2**, where it's **checked by Subnet 2's incoming Network ACL AND** through the **security group** of Instance B. Once through those, Instance B can receive and process the request.



After the transaction's complete, now it's time to come home. It's the **return traffic pattern**. It's the most interesting because this is where the stateful compared to stateless nature of the different engines comes into play—because the packet still has to be **evaluated at each checkpoint**. **Security groups**, by default, allow all return traffic. So, they don't have to check a list to see if they're allowed out. Instead, they **automatically allow the return traffic to pass by**, no matter what.

At the **subnet boundary**, these network ACLs do **not remember state**. Every entrance and exit is checked with the appropriate list. Stateless controls mean it **always checks its list**. Instance B's security group is first to evaluate, and since it's return traffic—and this security group saw the original request packet—the return traffic is allowed. The Network ACL of Subnet 2 goes to evaluate, and being stateless, it has to check the list again.

By the way, this is a **different list than on the way in**. Network ACLs have **rules** for both **incoming and outgoing** and they can be **different**. Now on the Instance A side, our packet is inspected as the stateless Network ACL of Subnet 1 checks the list every time. And the last check is the **security group of Instance A**. Again, because this is **return traffic**, it **doesn't check the list on the way back in**. So, the packet is **allowed**, and on it goes to complete the flow at Instance A.

It might seem like we spent a lot of effort just getting a packet from one instance to another and back. You might be concerned about all the network overhead this might generate. The reality is all of these exchanges happen instantly as part of how AWS networking actually works. That's why it's important to learn all of this.

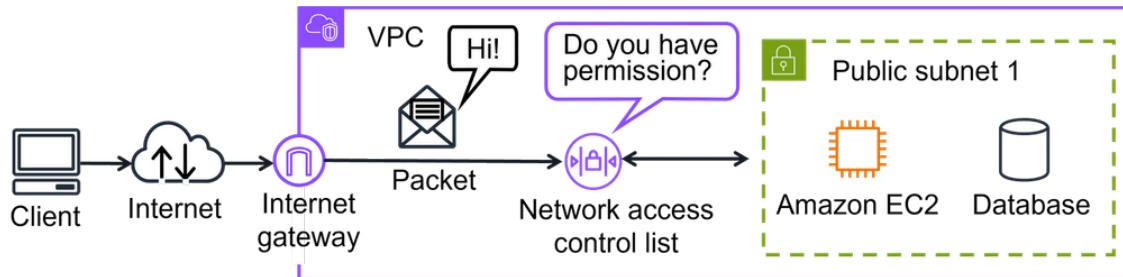
You want to ensure your **IT strategy** and design includes defining the **rules** for your **security groups** and **Network ACLs**. That way, your network design will meet your specific needs while also ensuring good network security. After all, security is a critical consideration in all networking for modern architectures.

Network traffic in a VPC

The movement of data packets traveling across a network

When a customer requests data from an application hosted in the AWS Cloud, this request is sent as a packet. A packet is a unit of data sent over the internet or a network.

It enters into a VPC through an internet gateway. Before a packet can enter into a subnet or exit from a subnet, it will run into several checks for permissions, one being a network ACL associated with the subnet the packet is being routed to. The permissions defined by the network ACLs indicate what is allowed or denied. It is based on who sent the packet and how the packet is trying to communicate with the resources in a subnet.



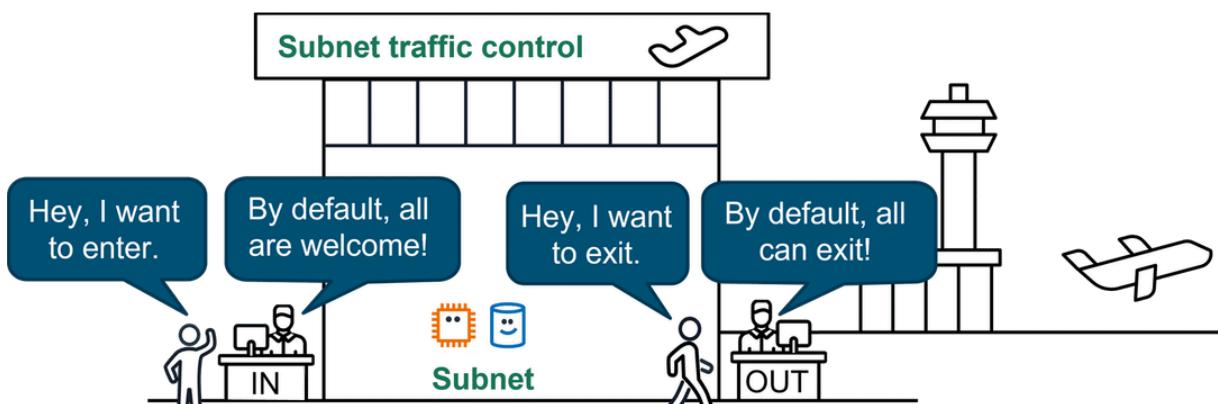
Network ACLs

Virtual firewall controlling traffic

A network ACL is a virtual firewall that controls inbound and outbound traffic at the subnet level.

For example, imagine that you are at the airport. Travelers are trying to enter into a different country. You can think of the travelers as packets and the passport control officer as a network ACL. The passport control officer checks travelers' credentials when they are both entering and exiting the country. This is similar to how a network ACL checks permissions every time a packet travels across a subnet boundary.

Each AWS account includes a default network ACL. When configuring your VPC, you can use your account's default network ACL or create custom network ACLs. By default, your account's default network ACL allows all inbound and outbound traffic, but you can modify it by adding your own rules.



Network ACL by default

For **custom** network ACLs, all **inbound** and **outbound** traffic is **denied** until you **add rules** to specify which traffic to allow. Additionally, all network ACLs have an **explicit deny rule**. This rule makes sure that if a packet doesn't match any of the other rules on the list, the packet is denied.

When it comes to securing the subnets and resources in your VPC with network ACLs and security groups, that is your responsibility.

AWS Security Groups

Security Groups vs Network ACLs

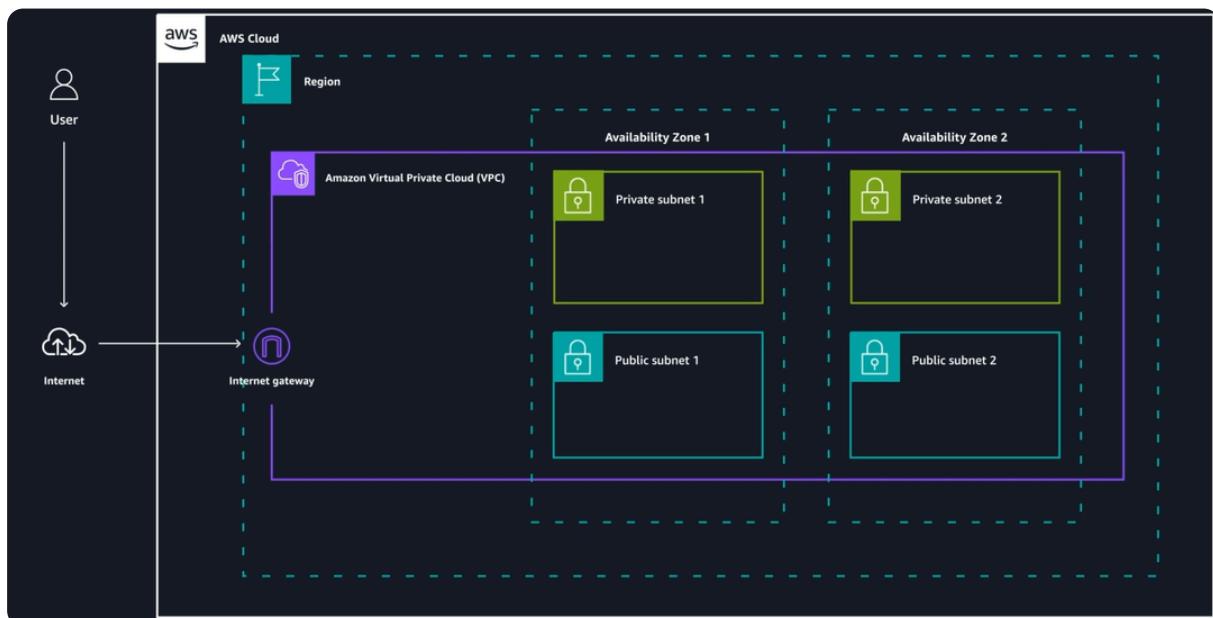
Feature	Security Groups	Network ACLs
Scope	Instance level (attached to EC2 instances)	Subnet level (associated with subnets)
State	Stateful (remembers state)	Stateless (doesn't remember state)
Rule types	Only allow type rules	Both allow and deny type rules
Return traffic	Return traffic is automatically allowed if inbound traffic is allowed	Return traffic must be implicitly allowed in both directions
Uses	Fine-grained control of traffic for individual EC2 instances	Broad control of traffic in and out of subnets

▼ Amazon Networking Demo

It's time to see how all this networking stuff really works in action. In this demo, I'm going to walk you through how to set up the network components you've just learned about.

Although this might not be something you personally would be required to do when working with AWS, I still want to show you how it works with a practical example using the AWS Management Console so you can wrap your head around it.

Let's start with the big picture of what we'll be building. We'll go through how to create a VPC, public and private subnets, an internet gateway, and a route table for our public subnets.



Let's jump into the first task, which is creating the **VPC**. In the search bar, I'll type in VPC, then choose VPC. This brings us to the **VPC dashboard**, and I'll now choose **create VPC**. On the Create VPC page, for Resource to create, I will choose **VPC Only**. For Name tag, I'll give it a name, like My VPC. And now it asks for something called a **classless inter-domain routing address**, or **CIDR** range. This address is used to define the **block of private IP addresses** available to assign to any resources launched into the VPC. We are going to use IPv4 addresses. So we can enter a value like this: four numbers divided by periods like 10.0.0.0/16. Then, scroll down and select Create VPC. With this, every resource in my VPC will get a **private IP address** that starts with 10.0, then the second two numbers will **vary from resource to resource**, each one having a **unique address** in the **VPC**.

The next task is to create the subnets. We are going to create both public and private subnets across two different Availability Zones, or AZs. This is a best practice to achieve high availability for your applications. It allows your instances to remain accessible, even if one AZ experiences an outage, by distributing them across separate physical locations within the same AWS Region.

Let's start by creating the private subnets first. Quick refresher, a **subnet** is a **range of IP addresses** in your VPC. You can launch AWS resources into a specified subnet. Use a public subnet for resources that must be connected to the internet, and a private subnet for resources that won't be connected to the internet.

I'll start with the private subnet. To create your private subnet, in the **left navigation pane**, under Virtual private cloud, **choose Subnets**. Then select **Create subnet**. On the Create subnet page, for VPC ID, I'll choose My VPC from the drop-down and then scroll down. Here under Subnet settings for **subnet name**, I'll type in Private-subnet-1. And then for Availability Zone, I will select us-east-1a. For IPv4 VPC CIDR block, that reflects the VPC CIDR block 10.0.0.0/16, and then we need to find the CIDR block for this subnet. Which will be **10.0.1.0/24**. Then we can select, Create subnet. Now our private subnet has been created, I will select it and then **select Actions**, and then select **Edit subnet settings**. On this page, there is this setting for **Auto-assign IP settings**, I want to make sure this is **not set**, because enabling this setting would give every resource in the subnet a public IP. This is a **private subnet**, so we don't want that. I'll go ahead and select cancel.

I won't demo the steps for time sake, but I would repeat these steps to create the **second private** subnet, placing it in a different Availability Zone and choosing a **different CIDR range**, like **10.0.2.0/24**. Alright, and our second subnet has been created.

Next, I'll create the **public subnets**. To do that I will select Create subnet. On the Create subnet page, for VPC ID, I will select My VPC from the drop-down. For Subnet name, I will enter Public-subnet-1. And then for the Availability Zone, I will us-east-1a. And for IPv4 subnet CIDR block, I will enter **10.0.3.0/24**. And then I will select Create subnet. Now our **public** subnet has been created. In the Actions menu, I will select Edit subnet settings. And then I will **check** the check box for **enabling auto-assign public IPv4 address**. This setting, provides a **public IPv4 address** for all instances or resources launched into this public subnet. Then, I will choose Save.

Now I would need to repeat those steps to create the second public subnet, placing it in a different Availability Zone and choosing a different CIDR range like **10.0.4.0/24**.

Now all four subnets have been created, two public and two private, spread across two different availability zones for this one VPC. It's important to note that even though your subnets are labeled Public 1 and Public 2, they are **not yet public subnets**. First, the **VPC** needs an attached **internet gateway**, which we will attach in the next step. Then, we need to create the **appropriate routes** in the **route tables**. Again, don't worry too much about remembering all of this.

So now let's go ahead and get that **internet gateway** created so that the public subnets can actually **connect** to the **internet**. In the **navigation pane**, under **Virtual private cloud**, I will choose **Internet gateways**. Then I will choose **Create** internet gateway. On the Create internet gateway page, for Name tag, I will enter in **my-ig**, and then select **Create internet gateway**. Then I will select **Actions**, and then **Attach to VPC**. In the search box, I will select My VPC, and then choose **Attach internet gateway**.

So now the internet gateway is attached to your VPC. Even though we now have created an internet gateway and it is attached, we still have to **tell instances within the public subnet how to get to the internet**. That's where the **route table** comes in.

A **route table** contains a **set of rules**, called **routes**, that are used to determine **where network traffic is directed**. Each subnet in your VPC must be **associated with a route table**. The table controls the routing for that subnet. So, to set up a route table in the console, we will go to the **navigation pane**, and then we will select **Route tables**.

We need to create a route table to **route public traffic** to the **internet gateway**. I will choose **Create route table**. And then on this page, for Name I will enter **public-route-table**. And then for the VPC, from the drop-down I will select My VPC and then choose **Create route table**.

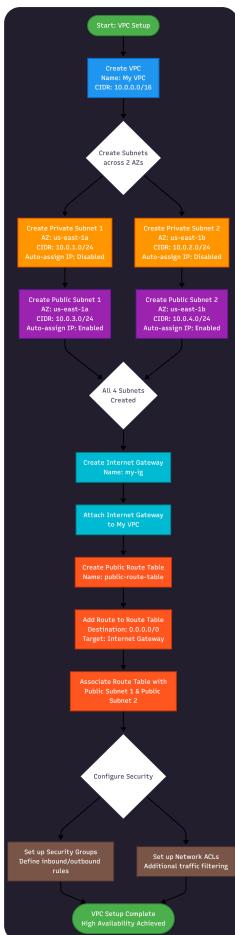
Now that a route table is created, under the **Routes** tab there is one route in your route table that allows traffic within the VPC to flow **within** the network, but it does **not route traffic outside of the network**. So, we need to add a **new route** to enable **public traffic**.

I will choose **Edit routes**. And then I will choose **Add route**. And for this route, under the **Destination**, I will enter **0.0.0.0/0**. And then for **Target**, I will select the **internet gateway** that we just **created and attach to this VPC**, and then I will select **Save changes**.

Now we need to associate this route table with our public subnets.

I will choose the **Subnet associations tab**. And in the **Explicit subnet associations section**, I will select **Edit subnet associations**. And then I will select **Public-subnet-1** and **Public-subnet-2**. Then I will choose **Save associations**. The subnet is now **public** because it is **connected to the internet** through the **internet gateway**. But the private subnets do not have that route so they remain private.

Now that we've created a VPC, four subnets, an internet gateway, and route tables, how could you filter the traffic coming in and out of this VPC? That's right, with **security groups** and **network access control lists**, or network **ACLs**. Remember those **CIDR blocks** you created earlier? Well **security groups** also use **CIDR blocks** to define sources or destinations for **network traffic**. This way, everything is secure and you decide who can access what resources based on your needs.



Flowchart Maker & Online Diagram Software

https://viewer.diagrams.net/?tags=%7B%7D&lightbox=1&highlight=0000ff&edit=_blank&layers=1&nav=1&transparent=1&dark=auto#G11E7lrGEF6Mh3sWYOOQOC7nTuedvKsAUK

▼ Global Networking

Folks, we've been talking a lot about how you interact with your AWS account, the services you deploy, and the applications you build. But how do your customers interact with your apps?

Say you are hosting a website. Your customers would typically type that website's address into their browser, press the Enter key, and before you know it, some magic happens—and the site lights up! If you're wondering where the magic comes in, it's like this coin that I have here right. I take a bite, ptooie! And it's back.

Well, not quite like that, but I'm going to take you through two services, which will help the website example make cents. Get it. The first is **Amazon Route 53**. Route 53 is a **domain name service**, or **DNS**. DNS acts as a translation service. However, instead of translating between languages, DNS translates website names into Internet Protocol, or IP addresses.

Amazon Route 53

Route 53 is a highly available and scalable cloud DNS service.



Humans can read website names and computers can read IP addresses. This acts as the bridge between the two. As an example of how it works, we type a website address into our browser and press the Enter key. The browser contacts Route 53 and asks it for the corresponding IP address of the site. Let's say, 192.2.0.2. The browser then knows where to go and gets routed to the website using that IP address. Additionally, Route 53 can route traffic to **different endpoints** using several different routing policies. These include **latency-based routing**, **geolocation**, **geoproximity**, and **weighted round robin**.

If we take **geolocation**, for example, this policy directs traffic based on where the **customer** is **located**. So, if a customer is located in North America, they are routed to one of the North American Regions. Likewise, if they are located in Ireland, they are routed to the Dublin Region. You can even use Route 53 to **register domain names**. Pick a domain name, check if it's available, and buy it using Route 53.

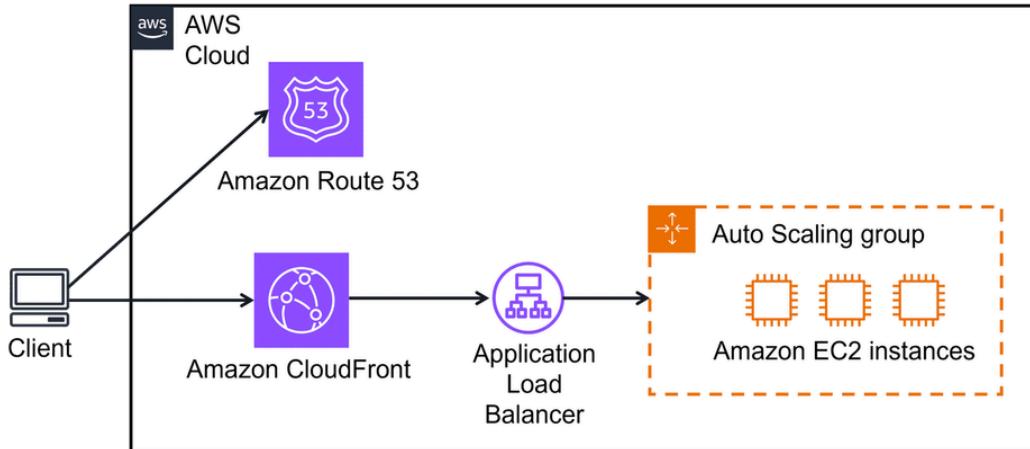
Amazon CloudFront

CloudFront is a CDN service that delivers your content with low latency and high speeds.



The other service that comes to the rescue is **Amazon CloudFront**. If you remember, we talked about **edge locations** earlier in the course. These locations serve content as close to customers as possible, and one part of that is a content delivery network, or **CDN**.

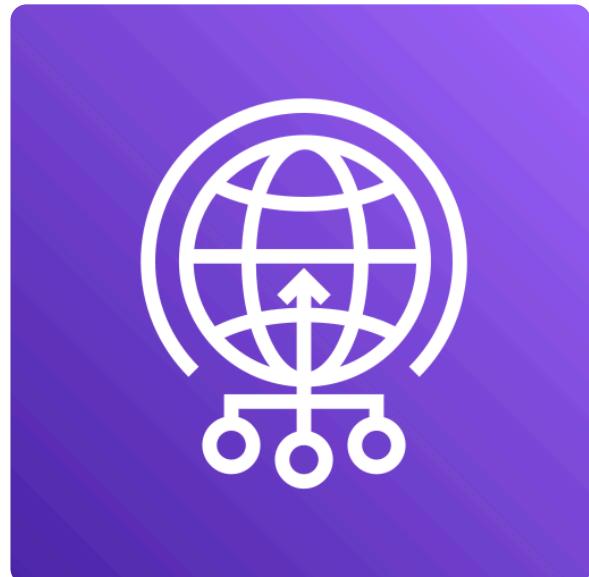
Let's go back to our Regions example. Say we have a user in Seattle, and they want to access a website. To speed this up, we host the site in Oregon and deploy our static web assets, like images and GIFs (or GIFs) in CloudFront's Seattle location. Shorter distance equals quicker delivery times.



I hope you are content after learning about these two services. So, thanks for following along, and I'm going to disappear just like this red cloth. So let's do this. Just pack it in. Abracadabra! Wow. Magic!

AWS Global Accelerator

Global Accelerator is a service that uses the **AWS global network** to improve application availability, performance, and security.



It uses **intelligent traffic routing** and **fast failover** if something goes wrong in one of your application locations.

Global Accelerator is a networking service that helps your applications run faster and more reliably across the globe. Think of it like creating express lanes on the internet highway specifically for your application's traffic. Instead of your users' requests taking the regular, sometimes congested internet routes, Global Accelerator directs traffic through the AWS private global network—getting your users to your application faster and more reliably.

- **IP Addresses:**

- If your application requires a **fixed, static IP address** for whitelisting or consistency (common in enterprise applications), you must use **Global Accelerator**. CloudFront uses dynamic IP addresses.

- **Protocols:**

- If you need to accelerate non-HTTP/HTTPS traffic (like **UDP** for online gaming or **TCP** for secure batch transfers), **Global Accelerator** is the necessary choice as CloudFront is built primarily for web protocols.

Feature	Amazon CloudFront (CDN)	AWS Global Accelerator (Network Service)
Primary Goal	Speed up content delivery by caching.	Improve application availability and performance by optimizing network path.
Optimization Technique	Content Caching at edge locations.	Optimal Routing over the AWS global network backbone.
Layer of Operation	Application Layer (Layer 7) —Understands HTTP semantics.	Network Layer (Layer 4) —Focuses on IP and transport layer.
Protocols	HTTP/HTTPS (Optimized for web traffic).	TCP, UDP, and HTTP/HTTPS (Handles a wider range of application traffic, like gaming or VoIP).
IP Addressing	Uses dynamic, DNS-based names (CNAME) that resolve to changing edge IP addresses.	Provides two static Anycast IP addresses as a fixed entry point to the application.
Failover	Distributes content from a primary origin; less focused on instantaneous multi-Region failover.	Provides instantaneous regional failover (within 60 seconds) by routing traffic away from an unhealthy endpoint.
Best Use Cases	Static websites, video streaming, API acceleration, software distribution.	Gaming, IoT, real-time communication, multi-Region applications requiring consistent performance/static IPs.

▼ Cloud in Real Life: Global Architecture

Morgan: Welcome back to Cloud in Real Life! We're going to go over some relatively complex network diagrams and use cases to reinforce the learning that you've done, but not to worry, you don't need to understand it all perfectly yet.

Rudy: And up until this point, you've been learning about a single VPC in a single region. However, in the real world, companies often need much more complex networks to support global customers. This can mean multiple AWS accounts, multiple AWS Regions, multiple VPCs...even hybrid cloud deployments. But let's start with a very common setup - a VPC with a VPN connection.

A virtual private gateway is the virtual private network (VPN) endpoint on the AWS side. It provides a way for you to establish a secure, encrypted connection between your on-premises network and your virtual private cloud (VPC).

Alan: This setup allows a company to securely **connect** their **on-premises network** to their cloud-based resources on **AWS**. This essentially creates a **private, encrypted tunnel** to **access data and applications** in the cloud **from their physical office location**, while maintaining **data security and privacy** over the **public internet**. Companies often use this for **remote employees** who need to access **sensitive information** that is **stored** in the **AWS cloud**.

Morgan: And while there are definitely benefits to using a VPN to connect to a VPC, and it works great for many customers, it does also have a few potential **limitations** to be aware of. One thing is that VPN connections do share the **bandwidth** of the local internet, so the connection can be prone to **slowdowns** if you have heavy payloads of data being sent over the internet to AWS. Another consideration might be your company requirements to meet certain **compliance or regulatory standards**. If you're worried about those types of things, then you might consider using **Direct Connect**.

Rudy: Exactly! Direct Connect is also awesome when lots of data needs to flow between corporate data centers and AWS. These **huge data transfers** can take a long time over the public Internet so some companies opt for Direct Connect instead. **Traffic** will be **routed** from their **corporate data center** to a **Direct Connect location**. It is then **routed** to a **VPC** through a **virtual private gateway**. All network traffic flows through this **dedicated private connection**. This helps to **speed up** data transfers, address application **performance** and increase a company's data transfer security.

Alan: So, when I have taught AWS networking, I get a lot of questions around when to use VPN or Direct Connect. Rudy, can you expand on that a little bit to clear it up?

Rudy: Dude, of course! Yes, yes, you should use **VPN** when you need a **secure, flexible connection** for **remote access** to your resources. This is especially true for **small-scale data transfers** or when a dedicated connection isn't necessary.

Morgan: And then it's a good bet to use **Direct Connect** when you need much higher bandwidth with a **dedicated line** like with **large data transfers** between your on-premises network and AWS.

Alan: And there are cases when you need to use both VPN and Direct Connect, right?

Rudy: Oh yeah, of course, a common use case for using a VPN alongside AWS Direct Connect is where you use **VPN as a failover for Direct Connect**.

Morgan: Right, because we have to remember that with **Direct Connect** these are **physical hard-wired connections**. So, if there is some situation where a line gets cut accidentally or if something were to happen, you can **VPN as a backup connection** for failover. But there are even cases where you may want to failover to a secondary direct connect line.

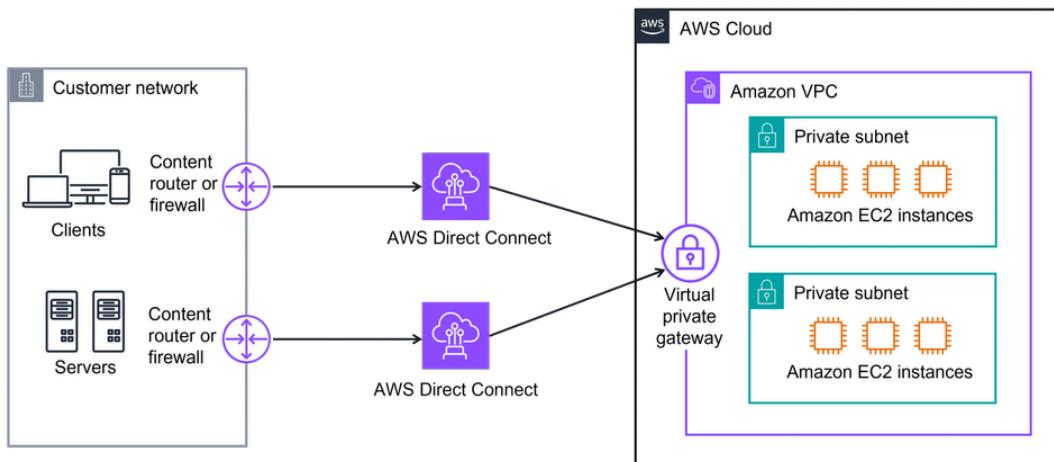
Alan: Sure! In addition to **fault tolerance**, if a customer wants **Increased bandwidth**, they can **combine multiple connections** to achieve **higher aggregate bandwidth**.

Morgan: Okay great, so that makes sense! Now, what about a real-world example of a company that needs to deliver content to several different regions globally? What would that look like?

Rudy: Well, for companies with customers around the globe or even offices in different Regions where they need to deliver content, they could use Amazon CloudFront and Route 53. CloudFront distributes content from edge locations globally, while Route 53 uses its **latency-based routing** capabilities to **direct users to the closest AWS region** (based on their location). This ensures they access the application with the **lowest latency**, which effectively provides a seamless experience across multiple Regions.

Alan: Yeah, let's take a look at how that works. Starting with a user, they access the company's **website** using a **custom domain**, the **request** is then **sent** to a **Route 53 DNS record**. Route 53 uses a **routing policy** to determine **which Region is closest to the user** and then directs them to the appropriate **CloudFront edge location**. The edge location then fetches the content from the designated origin server in the chosen Region. Notice how we're showing an architecture with multiple AWS Regions, and multiple VPCs!

Morgan: Right, this is a much more mature architecture than a single VPC in a single region. Again, you don't need to worry too much about getting all of this right for now, but it's good to have context for how customers are using this stuff in the real world.



Module 6 - Storage - AWS Storage Services

▼ Introduction to Storage

Things are really brewing in our coffee shop. Get it? Hah! And as things brew, we need to store and keep track of lots of records to make sure operations run smoothly. For example, we need a rolling inventory of our different varieties of coffee beans, sales transactions, marketing materials - oh, even our secret recipes.

No, no. These are for my eyes only. As for storage, you wouldn't just toss everything into an unorganized cupboard. You would use appropriate **storage solutions** to **organize** it accordingly. Maybe you would use airtight containers for coffee beans, a filing cabinet for paperwork, and a locked safe for those confidential recipes.

The same idea applies to the **different types of data stored** on AWS. Not everything fits neatly into one type of storage. Take files for example. They need to be stored and retrieved as complete objects. Think of an image or marketing PDF. Nobody wants half an image or three quarters of a PDF. They want the whole thing.

Logs are another type of data. They need **fast reads and writes** in a consistent manner. This requires a **different type of storage**. And sometimes, you have folders that everyone in your shop needs access to. These might include shared training manuals and internal documents. This is why AWS offers different options for **block, object, and file storage**.

Block storage

Block storage provides **persistent, low-latency** block-level storage volumes that **attach** to EC2 instances like physical hard drives. Block storage volumes can be encrypted, backed up via **snapshots**, and modified while in use without disrupting the instance. AWS offers two primary block storage services:

Block storage breaks data into manageable pieces called blocks, which makes it fast and more efficient to access. With **block storage**, data can be **updated, block by block**, meaning the whole file doesn't need to be changed every time you make an update.

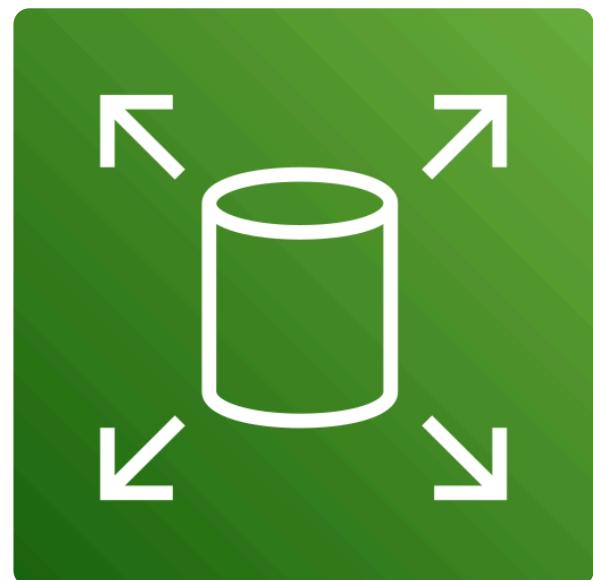
This makes it ideal for developers who work with **applications** or **databases** that need **quick and frequent updates**.

- **Amazon EC2 instance store**

An unmanaged **non-persistent**, high-performance block storage directly attached to EC2 instances for **temporary data**.

- **Amazon Elastic Block Store (EBS)**

A **managed** service that provides **persistent block storage volumes** for EC2 instances, offering various types for different workloads



Object storage

Object storage is a data storage architecture that manages data as objects in a **flat address space**. It offers **unlimited scalability** so you can store vast amounts of **unstructured data** without worrying about capacity constraints. Object storage provides enhanced **metadata** capabilities to provide more efficient data management, search, and analytics across massive datasets.

Object storage saves data in **self-contained units**, as, you guessed it...objects. Each of these **objects** includes the **data**, a **unique ID**, and information about the object called **metadata** that makes it easy to **organize** and **retrieve**. Unlike block storage, where you can update small parts of a file as needed, this type of storage requires **rewriting the entire object** for every change.

Objects are **organized** in **flat structures** called **buckets**, which are different from traditional hierarchical systems like folders. Object storage is best for files that don't change constantly, like **videos, backups, or logs**.

- **Amazon Simple Storage Service (S3)**
A **fully managed scalable** object storage service for storing and retrieving any amount of data from anywhere.



File storage

AWS file storage services provide **shared file systems** accessible over networks, so multiple users and applications can access the same data simultaneously. They offer **scalability** and flexibility so you can expand storage capacity as needs grow without managing physical infrastructure. AWS offers two primary file storage services:

Finally, **file storage** uses a **hierarchical file system** that can be **shared by applications**.

It's **compatible** with most systems, which means little or **no code modification** in most cases. File storage is ideal for applications that require **shared access**, like **content management systems**.

- **Amazon Elastic File System (EFS)**
A fully managed, scalable **NFS** file system for use with AWS Cloud services and on-premises resources.



- **Amazon FSx**

A fully managed file storage services for popular file systems like Windows, Lustre, and NetApp ONTAP.



Beyond these categories, we need to mention **databases**. They allow storage of **organized information** that usually needs to be **queried, updated, and analyzed constantly**. We provide various database services, too. We'll get to those soon, but for now, let's explore the different storage services on AWS in more detail!

These services don't fit cleanly into the categories we've defined so far, but they're important AWS storage offerings that you should be familiar with.

Additional storage services

- **AWS Storage Gateway**

A fully managed, **hybrid-cloud storage** service that provides **on-premises** access to virtually unlimited cloud storage.



- **AWS Elastic Disaster Recovery**

A fully managed service that streamlines the recovery of your physical, virtual, and cloud-based servers into AWS.



File Storage



Elastic File System (EFS)



- Shared file system between servers and applications
- Managed
- Hierarchical storage
- Simplicity & convenience
- Low Cost

Block Storage



Elastic Block Store (EBS)



Instance Store

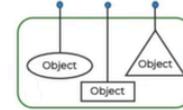
1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

- Each file separated into blocks
- Blocks retrieved in parallel
- Incremental editing of a file
- Better I/O performance
- Can be mounted where needed
- Operating system bootability
- * Expensive *

Object Storage



Simple Storage Service (S3)



- Objects have unique identifiers
- Infinitely scalable
- Flexible access from anywhere
- Contains rich metadata
- Better analytics
- Very Low Cost

File Storage

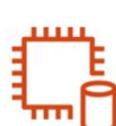


Elastic File System (EFS)

Block Storage



Elastic Block Store (EBS)



Instance Store

Object Storage



Simple Storage Service (S3)

Use Cases:

- File sharing & collaboration
- Structured storing of data
- Archiving files

Use Cases:

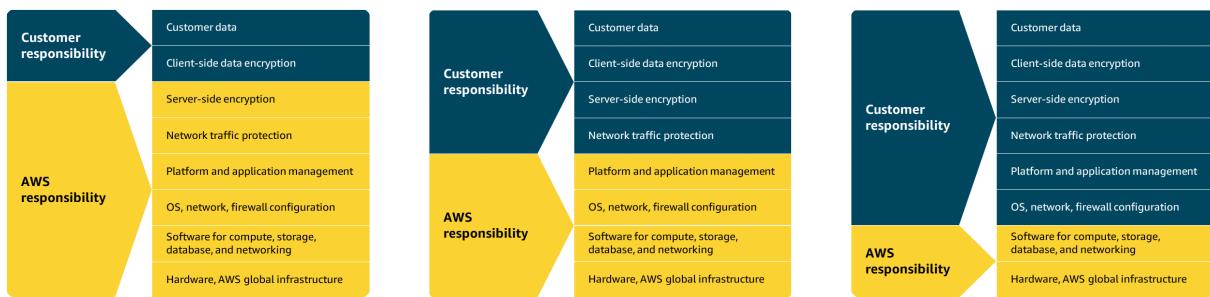
- Ideal for databases (SQL / noSQL)
- Data for applications with server-side processing (Java, PHP, .Net)
- Adding a file system

Use Cases:

- Video, Music, Images
- Backups and log files
- Big data



AWS Shared Responsibility Model - SRM



Management Category	AWS Storage Service	Customer Responsibilities	AWS Responsibilities
Fully Managed	Amazon S3 (Simple Storage Service)	Data management, access controls (IAM/Bucket Policy), service configuration, and security of data <i>in transit</i> .	All infrastructure, hardware, data durability (11 9s), availability, encryption at rest, and replication.
	Amazon EFS (Elastic File System)	Data management, access controls, network configuration (Mount Targets), and security of data <i>in transit</i> .	All infrastructure, hardware, capacity scaling (elastic), durability, and replication across AZs.
	Amazon FSx Family (e.g., FSx for Lustre/Windows File Server/ONTAP)	Data management, access controls, security of data <i>in transit</i> , and connecting the file system to compute instances.	All infrastructure, patching, software updates, durability, replication, and file system administration.
	AWS Backup	Defining backup policies, setting retention and compliance rules, and managing the backup vault access.	The entire backup infrastructure, scheduling, orchestration, and monitoring of the backup process.
Managed Services	Amazon EBS (Elastic Block Store)	Data backup strategies (snapshots), encryption configuration (KMS key), volume performance optimization (IOPS/Throughput), and capacity planning (resizing).	Underlying storage infrastructure, hardware redundancy, volume replication within an Availability Zone, and volume health.
Unmanaged Services	EC2 Instance Store	Full responsibility for data management, backup/recovery , encryption, performance	Maintains the underlying physical hardware and network

	optimization, and data durability (since data is lost on stop/terminate).	infrastructure of the EC2 host.
--	----------------------------------------------------------------------------------	---------------------------------

▼ EC2 Instances Store and Amazon Elastic Block Store - EBS

When you're using Amazon EC2 to run your business applications, those applications need access to CPU, memory, network, and storage. EC2 instances give you access to all of those components, but for right now, let's focus on the storage access.

As **applications run**, they will oftentimes need access to **block-level storage**. You can think of block-level storage as a place to **store files**. A file being a **series of bytes** that are stored in **blocks on disk**. When a file is updated, the whole series of blocks aren't all overwritten.

Instead, it updates just the pieces that need changed.

This makes it an efficient storage type when working with applications like **databases**, **enterprise software**, or **file systems**. When you use your laptop, you are accessing block-level storage. In this case, I'm talking about the hard drive for your laptop. **EC2** instances have **hard drives**, too, and there are a few different types.

Amazon EC2 instance store

Amazon EC2 instance store isn't a stand-alone AWS block storage service. Rather, it refers to the block-level storage that is physically attached to the EC2 instance host computer.

Depending on the type of instance, EC2 instance store might come attached as the default storage. Since its data is lost when an instance is stopped or terminated, EC2 instance store is best for temporary memory-based storage needs like buffers, caches, and scratch data. It is not recommended for applications that require data retention. An Amazon EC2 instance store provides temporary block-level storage for an Amazon EC2 instance. This means that if you stop or terminate an Amazon EC2 instance, all the data written to the attached instance store is deleted.

Depending on the type of the EC2 instance you launch, it can provide you with **physically attached local storage** called **instance store volumes**. The **catch** here is that because this volume is attached to the underlying physical host, if you stop or **terminate** your EC2 instance, all of **data written to the instance store volume will be deleted**.

The reason for this is that if you start your instance from a stopped state, it's likely that that EC2 instance will start up on another host. A host where that volume does not exist.

Because of the **ephemeral** or **temporary** nature of **instance store volumes**, they are useful in situations where you can afford to **lose the data** being written to the drive.

Benefits:

- **Automatically available storage**

Instance store volumes come automatically attached to many EC2 instance types, providing temporary block-level storage at no additional cost. It's physically connected to the host computer, offering high I/O performance for data that disappears when the instance stops.

- **Cost effective**

Because EC2 instance store is included in the EC2 instance price, you don't have to pay any additional fees for storage. It's ideal for temporary storage needs like buffers, caches, or scratch data, potentially reducing expenses for applications that don't require persistent storage.

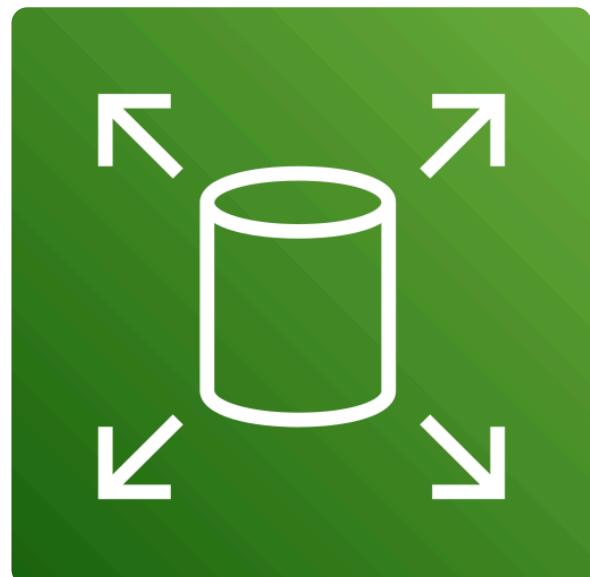
- **High performance**

EC2 instance store offers extremely low-latency storage directly attached to the host server of your EC2 instance. This proximity means exceptionally high I/O performance, making it ideal for temporary storage of data requiring fast processing.

Like if you need scratch space for heavy calculations or data processing. However, there are many use cases where you need **persistent data volumes**. You don't want your entire database getting deleted every time you stop an EC2 instance. Don't worry, this is where a service called **Amazon Elastic Block Store**, or Amazon **EBS**, comes into play.

Amazon Elastic Block Store (EBS)

Amazon EBS provides *persistent block-level storage* volumes for use with Amazon EC2 instances. EBS volumes act like external hard drives, offering consistent and low-latency performance for workloads like databases and file systems.



EBS volumes can be conveniently backed up, resized, and attached to different EC2 instances. To create an EBS volume, you **define** the configuration for things like volume **size** and **type**. After the volume has been created, it can be **attached** to an Amazon **EC2** instance. Because EBS volumes are for data that needs to persist, it's important to back up the data. It's recommended that you take **incremental backups** of EBS volumes by creating Amazon **EBS snapshots**. Amazon EBS provides block-level storage volumes that you can use with Amazon EC2 instances. If you stop or terminate an Amazon EC2 instance, all the data on the attached EBS volume remains available.

With Amazon EBS, you can create **virtual hard drives**, which we call **EBS volumes**, that you can **attach** to your **EC2 instances**. These are **separate drives** from the local **instance store** volumes, and they aren't tied directly to the host that your EC2 instance is running on.

This means, that the data that you write to an EBS volume can **persist** between **stops** and **starts** of an EC2 instance. Amazon EBS ensures data protection through **automatic replication** within the **same Availability Zone**. This provides the **high availability** and **durability** needed for financial applications with critical data.

EBS volumes come in all different **sizes** and **types**. How this works is you define the **size**, **type**, and **configurations** of the volume you need. Provision the volume, and then **attach** it to your **EC2 instance**. From there, you can configure your application to write to the volume, and you're good to go.

Use cases

Some practical use cases of Amazon EBS include database hosting, backup storage for applications, and **rapid deployment of development environments using volume snapshots**.

Benefits

- **Data migration**

EBS volumes can be easily migrated between Availability Zones using snapshots. The snapshots provide a simple way to move data across regions or create copies.

- **Instance type changes**

Since EBS volumes remain independent of EC2 instances, it's not complicated to attach them to different instance types. This flexibility lets you **upgrade or downgrade instances without losing data**.

- **Disaster recovery**

EBS snapshots provide reliable backup solutions that can be restored in different regions during emergencies. Regular automated snapshots ensure your data remains protected and quickly recoverable.

- **Cost optimization**

EBS volumes can be modified to different types and sizes to match actual usage patterns. You can switch between storage types or adjust capacity **without downtime**.

- **Performance tuning**

Amazon EBS offers various volume types to match different workload requirements and IOPS needs. You can **adjust volume performance characteristics on the fly** to meet changing application demands.

Now that we've covered the main features of block storage, it's also good to know that EBS offers different volume types that give you different levels of performance and offer **different pricing**. Performance for EBS volumes is measured in **IOPs**, or **input/output per second**.

Alright, we're not done with EBS just yet. Stay tuned to learn about EBS snapshots.

▼ Amazon Elastic Block Store Data Lifecycle

Let's dive a little deeper into the **data lifecycle** for Amazon **Elastic Block Store**.

If you recall from the shared responsibility model, you are responsible for managing your own data. This means you need to manage your **EBS volumes lifecycle**, which includes provisioning, moving, deprovisioning, and backups.

EBS Snapshot

EBS snapshots are **point-in-time backups** of EBS **volume**. They can be used for disaster **recovery**, **data migration**, **volume resizing**, and for creating consistent backups of production workloads. EBS snapshots are **incremental**, so they only save the blocks on the volume that have **changed** after your most recent snapshot.

EBS snapshots can be used to create multiple **new volumes**, and new volumes created from a snapshot are an **exact copy** of the original volume at the time the snapshot was taken.

Snapshots of EBS volumes are stored **redundantly in multiple Availability Zones** using Amazon S3.

Amazon **EBS snapshots** are **point-in-time copies** of your EBS volumes. Point-in-time means that the backup is done at a specific moment—in time. This can be daily, weekly or even monthly depending on your organization's needs. Even better, this means you can make **incremental backups** and restore them as needed.

But what does incremental backup mean? Well, let's say you take the first snapshot of your data on Monday. All of your data is **copied** in its **entirety**. Then, on Tuesday, you take another snapshot, but this time, it only **copies what has changed** since the **previous snapshot** was taken. That's the incremental part.

This approach makes subsequent snapshots **faster** and more **storage-efficient**, which translates to a significant reduction in **cost** compared to running a full backup every time.

But that only solves half of the problem.

You're taking these snapshots, but how do you manage them? The last thing you want is to log into the AWS Management Console every week and create a snapshot. It's time consuming and prone to human error. Imagine if you needed to do this for thousands of volumes. You'd be spending an hour just selecting all the buttons.

You want more automation? Use a **fully-managed** service like **Amazon Data Lifecycle Manager**. With Amazon Data Lifecycle Manager, you can define policies to help **automate** snapshot lifecycle management. You can schedule snapshot creation, set retention policies, manage lifecycles, and apply consistent backup policies across your organization.

This means nobody needs to log in to the console every Monday and click all of those buttons.

Working with EBS snapshots

In keeping with the AWS shared responsibility model, as the customer, you are responsible for scheduling and managing regular EBS snapshots as part of your backup strategy. This includes monitoring snapshot costs and deleting unnecessary snapshots to avoid excessive charges. You also need to make sure sensitive data within snapshots is encrypted, verify snapshot integrity, and test restoration procedures regularly.

Benefits

- **Data protection and recovery**

Snapshots enable **fast data recovery** from corruption, accidental deletion, or system failures using point-in-time backups.

- **Operational flexibility**

Snapshots enable operations like **cross-Region data migration**, **volume resizing**, **volume cloning**, and **sharing data across AWS accounts**.

- **Cost effective**

Snapshots use incremental backup technology, storing only changed blocks after the initial backup, reducing storage costs and backup time.

Amazon Data Lifecycle Manager

You can **automate the creation, retention, and deletion** of EBS snapshots using Amazon Data Lifecycle Manager. Amazon Data Lifecycle Manager can **schedule** snapshots during **off-peak hours** to minimize performance impact and **automatically delete outdated backups** to **control storage costs**. It's particularly valuable for **large-scale deployments** where manual snapshot management would be time-consuming and error-prone.

▼ Amazon Simple Storage Service - S3

It's time to cover a very popular storage service called Amazon Simple Storage Service, or Amazon S3. From the name, you've probably guessed that it's a storage service and it's, well, simple. Most businesses have data that needs to be stored somewhere.

For the coffee shop, this could be receipts, images, spreadsheets, employee training videos, text files, or other types of files.

Amazon Simple Storage Service (S3)

Amazon S3 is a **fully managed**, highly-available object storage service for storing and retrieving any amount of data as objects. It offers 99.999999999 percent durability, meaning your data is highly protected against loss, and offers features like versioning, lifecycle management, and various storage classes to optimize costs.



Each object typically includes the **data** itself, **metadata**, and a unique identifier, or **key**. Objects can be of any file type, such as images, videos, documents, or application data, and can range in size from a few bytes to several terabytes.

Each Amazon S3 object is uniquely identified within a bucket by its key, which is essentially its file name. Objects also have properties like version ID, access control information, and **user-defined** metadata.

Amazon S3 stores files as objects in containers known as buckets, and each object can range in size from a few bytes to several terabytes. It integrates seamlessly with other AWS services and supports a wide range of use cases, from basic backups to complex data lakes.

Storing these files is where S3 comes in handy because it's a data store that makes it possible to store and retrieve a **virtually unlimited amount of data at any scale**. Here are some of the basic concepts you need to know for object storage with S3.

Single data files, regardless of the type of file, are stored as **objects**. But instead of storing them in a file directory, you store them in what we call **buckets**. Think of a file sitting on your hard drive. That's like an object. And think of a **file directory**. That's like a **bucket**.

Now, the **maximum individual object size** that you can upload is **5 terabytes**, but there is no maximum on the total bucket size. So, you can have many objects in many buckets, and it's virtually unlimited storage. You can also turn on **versioning** for these objects to protect them from accidental deletion. With this feature you can always **restore** the previous versions.

Data stored in S3 for most storage classes is **automatically redundant**. Multiple object copies reside in the cloud simultaneously, and that is all managed by AWS. Now this means that all S3 objects have 11 nines of data durability. So, an object stored in S3 has a 99.999999999 percent probability that it will remain intact after a period of 1 year. That's a lot of nines!

S3 is commonly used for situations like: **hosting websites, storing backups, archiving data**, and managing media files like **videos** or **images**. It's a really versatile service that scales up and down as needed with your business, whether you're a small startup or a large enterprise.

For example, let's say you're running an online store. You might store your **product images** in S3. Then, when a **customer browses your website**, these **images** are quickly **retrieved** from S3 to display. And if your website gets a sudden **spike in traffic**, S3 can handle the increased demand without a problem and without you needing to set up **scaling**.

That is because **S3** is a **managed service**, so all of the underlying mechanisms for scaling and running the service are handled by AWS. S3 is also great for **data backups** because it ensures your data is safely stored and convenient to retrieve when needed. And because it's **durable**, remember those 11 9's?, this means you don't have to worry about data loss, even if something unexpected happens. Speaking of business critical, now let's talk about **securing** your **data** in **S3**.

S3 has multiple security features that make it possible for you to control **who** has **access** to **what**, creating a layer of protection for your data. By **default**, everything is completely **private** and only the person who created the object can access it. Then you have to configure any **additional access** you need through features like **bucket policies**.

Or, in some cases, you want to only grant **temporary access**, and you don't want to set up long-term access policies. For this use case, you can create **time-limited pre-signed URLs** for secure sharing without having to update your bucket policy. There are also features like **Amazon S3 Access Points**, which simplify the process of **creating unique access control policies for shared datasets**, where managing different band their respective access might be complicated.

And, finally, if you want to track who is accessing what, you can use **Amazon S3 Audit Logs** to **track** every **request** made to your resources. This provides complete **visibility** into who is **accessing** your data and **when**. That wraps up our introduction to S3, but there's more to come on this service, so stay tuned.

Benefits

- **Virtually unlimited storage**

Amazon S3 has no fixed storage limit, scaling automatically to accommodate any amount of data you need to store. Since you only pay for the storage you use, it's a cost-effective solution for growing data needs.

- **Object lifecycle management**

Amazon S3 lifecycle policies automatically move objects between storage classes based on your defined rules, optimizing costs over time. You can set up automatic transitions and expirations to manage data throughout its entire **lifecycle**.

- **Broad range of use cases**

Amazon S3 supports a wide range of use cases for both cloud-based applications and traditional on-premises workloads. Amazon S3 is commonly used for *content distribution*, *hosting static websites*, and delivering *media files*. It's also a popular choice for things like *application data storage*, *archiving*, *data lakes*, and *compliance-driven data retention*.

- **Bucket Policies**

Amazon S3 bucket policies are *resource-based policies* that can only be attached to S3 buckets. An S3 bucket policy specifies which actions are allowed or denied on the bucket, in addition to every object in that bucket.

- **Identity based Policies**

Permissions that control what actions users, groups, or roles can perform on S3 resources are configured using *identity-based policies*. These policies attach directly to identities rather than to the S3 resources themselves. You can use these policies to specify which S3 buckets and objects users can access and what actions they can perform.

- **Encryption**

Amazon S3 provides encryption capabilities to protect data both at rest and in transit.

▼ Amazon S3 Storage Classes and S3 Lifecycle

When you start working with Amazon S3, you'll notice that AWS offers different storage tiers, or "storage classes," to store your data. Each one is designed for different needs, and by understanding them, you can optimize for costs while making sure your data is stored in the most appropriate way. Let's start with the basics.

Some storage classes are meant for data that's **accessed frequently**, while others are better for data that's rarely used or needs to be **archived**. And the best part? You don't have to stick to just one storage class for all your objects. Within a single S3 bucket, you can **mix and match classes** based on **data access patterns**.

1. S3 Standard

The first storage class we'll talk about is S3 Standard. This one's great for data that you **access regularly**, like **files for a dynamic website**. It's a **general-purpose storage class** and gives you **fast retrieval speeds** and **affordable storage costs**.



1. S3 Standard Infrequent Access - IA

Now, if you don't need to access your data as often, the storage class S3 Standard-IA, or **Infrequent Access**, is a more **cost-effective** choice. You still get that quick retrieval when you need it, but at a lower storage cost. It's perfect for things like **backups**.

1. S3 Glacier Instant Retrieval

For even **lower-cost** options, we have the **Glacier storage classes**, which are designed for **long-term archiving**. There is S3 Glacier **Instant Retrieval**, which is for data that might require quick access on occasion but you still want to optimize for cost as much as possible. S3 Glacier Instant Retrieval provides the same **quick access speeds** as S3 Standard but has a lower storage cost and can be used for use cases like **medical images** or **media files**.



1. S3 Glacier Flexible Retrieval

However, if you don't need the quick access piece of this, and you are able to tolerate a bit of a wait for the data retrieval, then S3 **Glacier Flexible Retrieval** is a better choice. It can save you even more on cost, though it does take a little bit **longer to retrieve** — ranging from minutes to hours.

1. S3 Glacier Deep Archive

Moving on now to S3 **Glacier Deep Archive**, which is the **most cost-effective** option. This is the best option for storing **data you hardly ever need**, making it ideal for things like **compliance archives** or **digital preservation**. Data in this tier can be restored within **12 hours**.

1. S3 One Zone Infrequent Access - IA

However, we aren't done yet. S3 has a few other storage tiers for specific use cases you that should be aware of as well. Let's start with the **one-zone options**. S3 Express One Zone and S3 **One Zone-IA**, are **lower cost**, but might be susceptible to **data loss** in the unlikely case of the loss or **damage** to all or part of an **AWS Availability Zone**.

1. S3 Express One Zone

S3 Express One Zone stores data in a single Availability Zone. It was purpose-built to deliver consistent single-digit millisecond data access for your most frequently accessed data and latency-sensitive applications. S3 Express One Zone delivers data access speed up to 10x faster and request costs up to 80% lower than S3 Standard.

If you don't need the **extra resilience**, these can help **cut costs** or **improve access speeds**.

1. S3 Intelligent-Tiering

Now, your data might start off as frequently accessed, but then **over time** might become **less frequently accessed**. For example, when you make a new post on social media, it gets a lot of attention for a few days. But a month or a few months out, there are likely very few people looking at and interacting with those posts.

Well, that's where S3 Intelligent-Tiering comes in. This storage class **automatically** moves data between **four different tiers** based on **how often it's accessed**, helping you **optimize for cost** without doing any of the work yourself. It's perfect for things like **data lakes** or **large datasets** that might **change their usage patterns**.

By understanding these options and using tools like Intelligent-Tiering and Lifecycle management, you can store your data more efficiently and optimize for cost.

1. S3 Outposts

Amazon S3 Outposts delivers object storage to your on-premises AWS Outposts environment using **Amazon S3 APIs** and **features**, and serves workloads with **local data residency requirements**. It also helps maintain optimal performance when data must remain in close **proximity to on-premises applications**.



You can **optimize your storage classes** in AWS for these types of situations using **S3 Lifecycle policies**. These are **rules** you set up to help you **automatically move data between classes** or **delete old data** when you no longer need it. For **example**, you can set up a Lifecycle policy to move old data into Glacier after a year, and then automatically delete it when it's no longer necessary for compliance.

To help you figure out the best way to move your data, you can use **S3 Storage Class Analysis**. This feature looks at your **data access patterns** and helps you decide when to move data to a more cost-effective storage class. But what if you want to take advantage of Amazon S3 storage tiers without managing the lifecycle policies yourself?

Amazon S3 storage classes and use cases

Amazon S3 offers various storage classes to suit a variety of **workloads** with specific **performance, access, resiliency**, and **cost** requirements. They're also designed to address **data residency requirements**, **unpredictable access patterns**, **archival storage** needs, and offer the most **cost-effective** options for different access patterns.

S3 Classes Table

Storage Class	Primary Access Frequency	Retrieval Speed	Ideal Use Case	Minimum Storage Duration	Availability/Durability
S3 Standard	Frequent (Default)	Millisecond (Fast)	Dynamic websites, content distribution, frequently accessed data.	None	High (99.99% Avail. / 119s Dur.)
S3 Standard-IA (Infrequent Access)	Infrequent	Millisecond (Fast)	Backups, disaster recovery files, data accessed less than once a month.	30 days	High
S3 Express One Zone	Most Frequent (Latency - Sensitive)	Single-Digit Millisecond (Ultra-Fast)	High-performance computing (HPC), AI/ML training data, interactive data analytics, and real-time streaming.	None	Durability within a single AZ (Data is <i>not</i> replicated across AZs).
S3 One Zone-IA	Infrequent	Millisecond (Fast)	Secondary backups of data that can be easily recreated; cost-optimized when cross-AZ replication isn't needed.	30 days	Lower (single AZ)
S3 Glacier Instant Retrieval	Archival (Accessed rarely)	Millisecond (Fast)	Archives that require immediate, on-demand access (e.g., historical medical records).	90 days	High
S3 Glacier Flexible Retrieval	Archival (Accessed very rarely)	Minutes to Hours (Configurable)	Long-term archives, media assets where retrieval can wait for a few hours.	90 days	High
S3 Glacier Deep Archive	Archival (Extremely rare access)	Hours (12–48 hours)	Long-term regulatory compliance archives; lowest-cost storage option.	180 days	High

S3 Intelligent-Tiering	Varies (Unknown/Changing)	Millisecond (Fast)	Data with unpredictable access patterns; automatically optimizes cost by moving data between tiers.	None (Small monitoring fee applies)	High
S3 Outposts	Frequent (Local Access)	Millisecond (Local Network Speed)	Workloads with local data residency requirements or on-premises applications requiring high-throughput, local object storage.	None	Durability is across devices/servers on your Outpost rack (local redundancy)

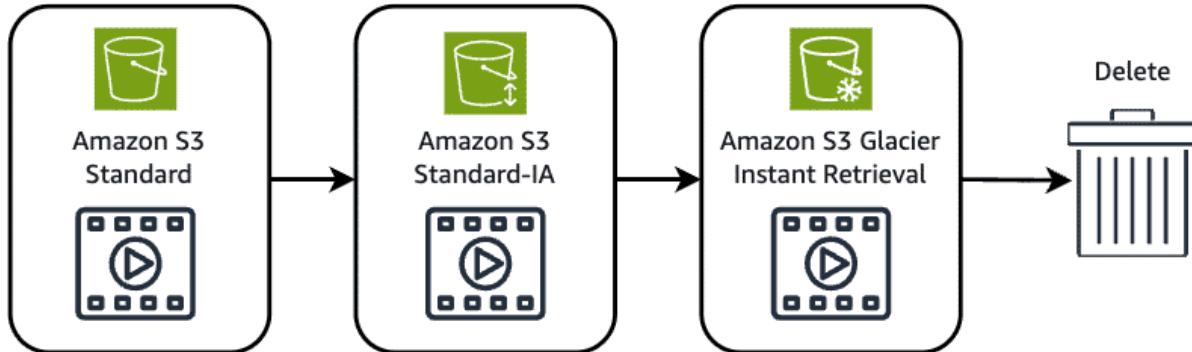
S3 Lifecycle

To avoid manually managing your object storage tier configurations, you can use **S3 Lifecycle configurations** to automate the process. When you define a lifecycle configuration for an object or group of objects, you can choose to automate between two types of actions, as follows:

- *Transition actions*: define when objects should transition to another storage class.
- *Expiration actions*: define when objects expire and should be permanently deleted.

For example, you might transition objects to S3 Standard-IA storage class 30 days after you create them. Or you might archive objects to the S3 Glacier Deep Archive storage class 1 year after creating them.

30 days without access request → 60 days → 365 days → Delete/Expire



▼ Amazon S3 Demonstration

In this demo we're going to walk through some of the basic features and functionalities of Amazon Simple Storage Service, or Amazon S3. We will start by accessing S3 using the AWS management console. In the search bar I will type S3 and then select the Amazon S3 service. Let's create an S3 bucket to get things going.

Choose *create bucket* to start the process. For this you'll need to enter a **globally unique bucket name** meaning it should be unique across all AWS accounts worldwide. I'm gonna enter *Morgan bucket 2025*. Then we need to choose our preferred AWS **region** from the menu. I'm going to go ahead and leave the region to be *North Virginia*.

Scrolling down, notice how the **block all public access settings are enabled** which prevents any external access to this bucket or its contents. For now keep these settings as is and then we can scroll down and choose **create bucket at the bottom of the page**.

To access your newly created bucket choose the link for the bucket from the list of buckets here. You'll notice that this bucket is currently empty which is expected. But where's the fun in an empty bucket. Let's go ahead and create an organized structure for our bucket content by creating a **folder**. Choose *create folder* and then give the folder a name I will call this folder.

Then we can scroll down and notice we're presented with some folder encryption options in case you want to configure any security at this point in the process. We're going to go ahead and keep the default encryption settings and then we will select *Create folder*. Now a new folder has been created within our bucket.

Let's get some objects into our folder by uploading a file from local storage. First, select the newly created folder to open it and then choose upload and then add files. Browse your local system and select a file you would like to add. I'm going to select this demo file and then select open.

Let's now take a moment to explore the object **metadata**. Choose **properties** to expand this section navigate to the *metadata* subsection of this page, and then you can **add metadata**. There are 2 types of object metadata in Amazon S3. **System-defined metadata** includes basic object details like **creation date size and storage class** and **user-defined metadata** can be used to include your own **custom metadata like tags**.

All right, now I'm going to go ahead and remove this and from here we can go ahead and upload the object. This file should now be listed in our bucket inside of our folder. Let me go ahead and check. We can see the file has now been uploaded.

Now let's say you want to **upload a folder and all of its contents** to your bucket. Although it's not possible using the add files method we just covered you can accomplish this using the drag and drop upload method. Identify the local folder on your computer that you want to upload and then choose the upload button. But this time, simply **drag the folder** from your local storage directly to your Amazon S3 folder. This is also something you can do using the **command line interface** or the AWS SDK.

I will scroll down and choose *upload*, and as the progress bar indicates that the upload is complete, we can then choose the name of the folder and we can drill down into the folder that we just uploaded. You'll notice that Amazon S3 has **maintained the original file hierarchy** during the upload process. To return to the bucket we created earlier, locate in the breadcrumb navigation and choose the bucket name.

Now let's explore some **bucket level configurations**, choose the **properties tab**, and here you can **review and modify** important **features** including **bucket versioning, tags, default encryption, intelligent tiering, archive configurations, static website hosting**, and more.

Next we will scroll back up and select the **permissions tab**. On this tab are sections that allow you to configure a variety of **access** settings for the bucket. This includes the **block public access setting, the bucket policy, object ownership, cross-origin resource sharing**, and more.

Scrolling back up let's take a second to talk about bucket policies. An S3 **bucket policy** is a **JSON** statement that provides **access** to objects stored in a bucket. Let's add a bucket policy by choosing **edit** and then you can paste in your bucket policy.

Let's paste in a pre-created bucket policy that will grant cross account permissions to upload objects while ensuring that the bucket owner has full control. This bucket policy has one **statement** that **defines the effect as allow**.

The **principal** is the **AWS account number** that you want to allow to upload objects.

The **action** is the S3 **API** call this effect is taken on and in this case that is the **put object call**.

Then there is the **resource** which **defines** what resources in this **bucket** this policy impacts.

Finally, a **condition** definition states the **upload** is only **allowed cross account** if the **metadata** around the **object ownership** is present and configured **correctly**. Then I can scroll down and select **save changes**.

Now scrolling up let's select the **management tab** where you have the option to set up several advanced features. You can change the **life cycle configuration replication, configuration and inventory configurations** from here.

Now that you have a solid understanding of the available bucket level configurations let's look at what you can configure at the **object level**. Scrolling up and selecting our **objects tab** I can then select one of the objects that we uploaded to view its properties. The **object overview** section shows basic information about the object including the **object URL**. If **block public access** is disabled and the bucket permissions allow anyone can **access the object through the URL**.

However, for our example here, **block public access** is turned on. So you would use the **bucket policy** to define **specifically** who has **access** to this **object**. Further down on the properties tab are sections for the **object management overview, storage class, server side encryption, checksums, tags, metadata, and object lock**.

You **absolutely can** use a bucket policy to grant access to specific users, **provided those users are authenticated** using AWS credentials (IAM Users, IAM Roles, or AWS Accounts).

Block Public Access **does not** prevent authenticated, private access.

Finally, I will scroll back up and select the **versions tab**, and here you could display a list of all previously created versions of the object if they existed. You can also **restore previous versions delete specific versions** and **manage the complete version history of your objects**.

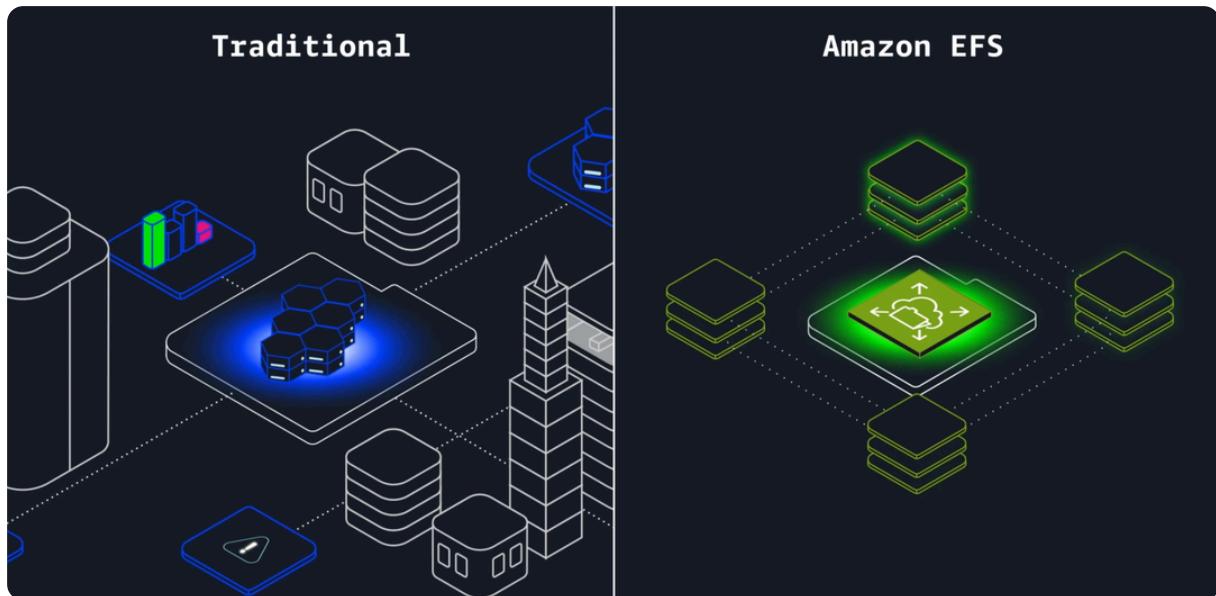
All right, we are just getting started with S3 but for now you have a solid understanding of the fundamentals.

▼ Amazon Elastic File System - EFS

Next up on the list of storage services is a managed file system called Amazon Elastic File System, or Amazon EFS. It's extremely common for businesses to have shared file systems across their applications.

For example, you might have multiple servers running analytics on large amounts of data being stored in a **shared file system**. If this were hosted traditionally, it would require planning for the storage capacity, deciding on data redundancy levels, backup strategies, and maintenance as the dataset grows.

With EFS, you can keep existing file systems in place while AWS does all the heavy lifting of the **scaling** and the **replication**. EFS makes it possible for you to have **multiple instances accessing the data in EFS at the same time**. It scales up and down as needed without you having to do anything to make that scaling happen.



When you create an **EFS filesystem**, it comes with **pre-configured lifecycle policies** that help keep your cloud data **cost-optimized**. EFS **automatically moves data** between **storage classes** based on access patterns, but you have the option to **customize the lifecycle policies** to fit your storage needs. Super nice, right?

EBS and EFS might seem similar, so let's take a moment to distinguish the two. **EBS volumes attach to EC2 instances** and are an **Availability Zone level resource**. In order to attach EC2 to EBS, you need to be in the **same Availability Zone**.

An **EBS volume is effectively a hard drive**. You can save files on it, run a database on top of it, or store applications on it. If you provision a 2 TB EBS volume and fill it up, it **doesn't automatically scale** to give you more storage. So, that's EBS.

Amazon Elastic File System (EFS)

Amazon EFS is a **fully managed, scalable** file storage service for use with AWS cloud services and on-premises resources. It operates using the *Linux Network File System (NFS) protocol*, and **automatically scales** to petabytes as you add or remove files without disrupting applications. EFS is designed to support a wide variety of workloads and can be accessed by *multiple EC2 instances simultaneously*.



EFS can have **multiple instances** reading from and writing to it at the **same time**. But it isn't just a blank hard drive for 1s and 0s — it's a true **file system** for **Linux**. EFS is **not AZ-limited**, you can connect storage to EC2 instances throughout the region or even from other regions. As you write more data to EFS, it **scales automatically** - no need to provision any more volumes.

Benefits

- **Multi-AZ redundancy**

Amazon EFS automatically replicates data across multiple Availability Zones in a region for high availability. This built-in redundancy protects against AZ failures and provides continuous access to your file systems.

- **Shared access**

Amazon EFS supports thousands of concurrent NFS connections, so multiple EC2 instances can access the same file system simultaneously. This shared access model makes EFS ideal for **collaborative workloads** and distributed applications.

- **Elastic storage**

Amazon EFS automatically grows and shrinks as you add and remove files, with no need to provision or manage storage capacity. And since you only **pay for the storage you use**, it's cost-effective for varying workload demands.

Amazon EFS storage classes

With Amazon EFS, you can create and configure file systems quickly without any minimum fee or setup cost. You pay only for the storage used and you can choose from a range of storage classes designed to fit your use case.

1. Standard Classes

The *EFS Standard* and *EFS Standard-Infrequent Access (Standard-IA)* storage classes offer Multi-AZ resilience and the highest levels of durability and availability. They have a higher cost associated with them due to higher availability and durability.

2. One Zone Classes

The *EFS One Zone* and *EFS One Zone-Infrequent Access (EFS One Zone-IA)* provide additional savings by saving your data in a single Availability Zone. By using just one Availability Zone, you can reduce your storage costs when compared to the Standard EFS storage classes.

3. Archive Class

The *EFS Archive* storage class is cost-optimized for data that is accessed only a few times a year or less and that does not need the sub-millisecond latencies of EFS Standard. EFS Archive offers a storage price up to 50% lower compared to EFS Infrequent Access, providing a more cost-optimized experience for cold, rarely-accessed data.

EBS vs EFS comparison table

AWS CCP

Feature	Amazon Elastic Block Store (EBS)	Amazon Elastic File System (EFS)
Storage Type	Block Storage (raw, unformatted volumes)	File Storage (NFS protocol-based file system)
Access Protocol	Accessed via Storage Area Network (SAN) protocols. Appears to the OS as a traditional hard drive.	Accessed via the NFS (Network File System) protocol .
Concurrent Access	No. Can be attached to only one EC2 instance at a time (except for Multi-Attach on specific volume types).	Yes. Can be mounted and accessed concurrently by hundreds or thousands of EC2 instances, Lambda functions, or ECS tasks.
Geographic Scope	Zonal. A volume exists within a single Availability Zone (AZ) and can only be attached to instances in that same AZ.	Regional. The file system spans all Availability Zones within a Region, allowing multi-AZ access.
Capacity & Scaling	Fixed Capacity. You provision a specific size (e.g., 500 GiB). To increase size, you must modify the volume.	Elastic/Automatic Scaling. Grows and shrinks automatically as you add and remove files. Virtually unlimited capacity (petabytes).
Performance Unit	Measured by IOPS (Input/Output Operations Per Second) and Throughput (MB/s).	Measured by Throughput (MB/s) and scales with the size of the file system (or provisioned).
Ideal Use Case	Boot Volumes (OS), Databases (MySQL, PostgreSQL), single-instance applications requiring low latency.	Shared Storage , content repositories, web serving content, development environments, and home directories.
Mounting	Appears as a standard device to the OS (e.g., <code>/dev/sdc</code>). Requires formatting and mounting.	Mounted as a network file share using a Mount Target in each AZ.
Cost Basis	Charged based on the provisioned size (GB/month) and the provisioned performance (IOPS/Throughput).	Charged based on the average usage (GB/month) and the access tier (Standard or Infrequent Access).

▼ Amazon FSx

Amazon FSx makes it convenient and cost effective to launch, run, and scale feature-rich, high-performance file systems in the cloud. Amazon FSx supports multiple filesystem protocols, including Windows File Server and etc. As a fully managed service, it handles hardware provisioning, patching, and backups, and lower total cost of ownership (TCO).



- **File system integration**

Amazon FSx supports industry-standard file system protocols, allowing convenient integration with your existing applications, workflows, and development tools.

- **Managed infrastructure**

Amazon FSx reduces the complexity of managing infrastructure while delivering the features and capabilities of traditional file systems.

- **Scalable Storage**

Amazon FSx adjusts resources dynamically, eliminating the need for complex capacity planning and manual infrastructure management.

- **Cost effective**

Amazon FSx has a pricing model and automated tiering options that optimize costs by charging only for used storage and moving infrequently accessed data to lower-cost tiers.

Amazon FSx for Windows File Server

Amazon FSx for Windows File Server provides fully managed shared storage built on Windows Server. It delivers a wide range of data access, data management, and administrative capabilities.

Use cases include the following:

- Migrate Windows file servers to AWS.
- Accelerate hybrid workloads.
- Reduce SQL Server deployment cost.
- Streamline virtual desktops and streaming.

Amazon FSx for NetApp ONTAP

Amazon FSx for NetApp ONTAP provides fully managed shared storage in the AWS Cloud with the popular data access and management capabilities of ONTAP.

Use cases include the following:

- Migrate workloads to AWS seamlessly.
- Build modern applications.
- Modernize your data management.

- Streamline business continuity.

Amazon FSx for OpenZFS

Amazon FSx for OpenZFS provides fully managed shared file storage built on the OpenZFS file system and accessible through the NFS protocol (v3, v4, v4.1, and v4.2).

Use cases include the following:

- Migrate workloads to AWS seamlessly.
- Deliver insights faster for data analytics workloads.
- Accelerate content management.
- Increase dev/test velocity.

Amazon FSx for Lustre

Amazon FSx for Lustre provides fully managed shared storage with the scalability and performance of the popular Lustre file system.

Use cases include the following:

- Accelerate machine learning (ML).
- Enable high performance computing (HPC).
- Unlock big data analytics.
- Increase media workload agility.

Feature	Amazon Elastic Block Store (EBS)	Amazon Elastic File System (EFS)	Amazon FSx Family
Storage Type	Block Storage (raw volume)	File Storage (NFS-based)	File Storage (Specialized/Commercial)
Concurrent Access	No (Single Instance)	Yes (Thousands of instances)	Yes (Protocol-dependent, typically thousands)
Geographic Scope	Zonal (Single AZ)	Regional (Spans All AZs)	Regional (Typically Multi-AZ options)
Capacity & Scaling	Fixed Capacity (Must be resized manually).	Elastic/Automatic (Grows/shrinks automatically).	Fixed/Scalable (Varies by type; some scale automatically, others provisioned).
Ideal Use Case	Boot volumes, Databases (single-instance), high-performance transactional data.	Linux Shared Storage, general-purpose file sharing, web serving content.	Windows File Server: Lift-and-shift Windows apps. Lustre: HPC, ML, Big Data. ONTAP: Hybrid cloud, advanced data management (snapshots, dedupe).

▼ AWS Storage Gateway

AWS Storage Gateway bridges the gap between your traditional infrastructure and the cloud. It's a **hybrid cloud storage service** that gives you **on-premises access** to virtually unlimited **cloud storage**. You can use it to **extend your local storage** to the cloud while **maintaining low-latency access** to frequently used data.



Although the cloud offers lots of options, some organizations still run much of their business using an on-premises environment. Look, if it works, don't break it. But, if those organizations still want to take advantage of the cloud, to say, perform backups that's a very valid use case.

This is where AWS Storage Gateway comes in handy. It is a hybrid cloud storage service that is like the bridge, giving you access to virtually unlimited cloud storage. There is minimal or no rework needed, and everything is **seamlessly backed** up to AWS. Storage Gateway comes in three types: **S3 File Gateway**, **Volume Gateway**, and **Tape Gateway**.

Amazon S3 File Gateway

Amazon S3 File Gateway bridges your local environment with Amazon S3. It provides **on-premises applications** with access to virtually unlimited cloud storage through **familiar file protocols**. S3 File Gateway makes it possible to store and retrieve cloud objects using **familiar file operations**.

When you deploy an S3 File Gateway, it appears to your local systems as a **standard file server**. Files written to this server are **automatically uploaded to Amazon S3** while **maintaining local access to recently used data** through **intelligent caching**. This means your applications can continue working with files as they always have while the actual data is securely stored in the AWS Cloud.

S3 File Gateway is ideal for storing **files directly in Amazon S3**. Even better, you can access these files from any AWS application or service.

Volume Gateway

With Volume Gateway, you create **virtual storage volumes** while maintaining local access to your data. It essentially functions as a bridge between your on-premises infrastructure and AWS Cloud storage by **presenting your cloud data as iSCSI volumes** that can be mounted by your existing applications.

Volume Gateway operates in **two main configurations**:

- **Cached volume mode** - stores primary data in the cloud while **frequently accessed** data is **cached locally** for **low-latency** access.

- **Stored volume mode** - locally keeps your complete dataset while asynchronously **backing** it up to the cloud as **EBS snapshots**.

Volume Gateway is a bit different in that it **creates block storage volumes locally**.

On-premises applications can then use these volumes. The key benefit is that files are **automatically backed up to AWS as Amazon EBS snapshots**.

Tape Gateway

Tape Gateway makes it possible to replace physical tape infrastructure with **virtual tape capabilities** while benefitting from the **durability** and **scalability** of **AWS Cloud storage**. Tape Gateway provides an interface that works with existing tape backup software, making the transition from physical tapes to cloud storage **seamless**.

When you deploy a Tape Gateway, it presents itself to your **backup applications as standard tape hardware**. Your backup software writes data to these virtual tapes just as it would to physical tapes and stored in Amazon S3. You can also configure Tape Gateway to **automatically transition less frequently accessed data** to a more **cost-effective storage class** for long-term retention.

The final type, **Tape Gateway**, is super useful for businesses still using **physical tape backups**. It provides that **seamless migration of tape data into the cloud**.

As for some example use cases along with benefits, Storage Gateway is a lifesaver for **disaster recovery**. Why? Because you can keep a **backup** of your critical **data in the cloud without changing your existing backup workflows**. It's like a safety net for your current **safety net**. Like a safety-safety net.

It's also ideal for **data archiving**. You can move your **infrequently accessed data** to the cloud for **long-term storage**. Whether you want to **backup, archive**, or just start your **cloud journey**, AWS Storage Gateway has you covered.

And if you like it, cool. If you don't, that's cool, too. We want the best solution for our customers, even if it means using their existing one. So, don't stress. Just keep learning.

Benefits

- **Seamless integration**

Storage Gateway enables smooth connectivity between on-premises applications and AWS Cloud storage, preserving existing workflows and minimizing disruption.

- **Improved data management**

Storage Gateway provides **centralized management** of hybrid storage environments, enhancing accessibility, security, and compliance.

- **Local caching**

Storage Gateway **locally keeps frequently accessed data** for quick access while managing less-used data in the cloud.

- **Cost optimization**

Storage Gateway reduces on-premises storage costs by using cloud storage for data archiving, backup, and disaster recovery purposes.

Feature	Amazon FSx (Fully Managed Cloud File System)	AWS Storage Gateway (Hybrid Cloud Bridge)
---------	----------------------------------------------	-------------------------------------------

Primary Location	In the AWS Cloud (within a specific Region/VPC).	On-Premises (You deploy a VM appliance in your data center).
Core Purpose	To provide feature-rich, high-performance file systems for applications running in AWS.	To provide on-premises applications with low-latency access to cloud storage (S3, EBS, Glacier).
Data Residency	All primary data resides in the cloud (managed by FSx).	Primary data resides in the cloud (S3, EBS, or FSx), with a locally cached copy on your on-premises hardware.
Storage Type	Provides managed instances of specialized file systems: Windows File Server (SMB) , Lustre, OpenZFS, NetApp ONTAP.	Provides a local interface to different cloud storage types: File (S3/FSx) , Volume (EBS) , and Tape (Glacier) .
Management Focus	Managing the complexity of the file system (patching, backups, redundancy, performance scaling).	Managing the data transfer and caching between your local network and the AWS cloud.
Key Use Case	Running Windows applications that require SMB shares, high-performance computing (HPC) workloads, or seamless integration with ONTAP features.	Backup and disaster recovery (moving local tape backups to Glacier), offloading file shares to S3, or providing local access to cloud data.

Note on Amazon FSx File Gateway

It's important to know that there is a specific type of Storage Gateway called **Amazon FSx File Gateway**. This gateway connects your on-premises machines to an **FSx for Windows File Server** instance running in the cloud. In this case, the Storage Gateway is acting as the **local cache for the FSx file system**, allowing remote users to access the cloud-based FSx share quickly without having to route all their traffic over the full WAN distance.

▼ AWS Elastic Disaster Recovery

Elastic Disaster Recovery replicates critical workloads to AWS with minimal downtime. Your servers' block-level data is continuously replicated to AWS, making it ideal for uses that require robust disaster recovery solutions. It supports both physical and virtual servers to enable rapid recovery during disruptions, which is particularly valuable for industries like healthcare where system availability is crucial.



You can use Elastic Disaster Recovery to reduce downtimes and data loss while eliminating the costs associated with maintaining secondary data centers. It also offers non-disruptive disaster recovery testing, meaning it's capable of quickly launching recovery instances when needed.

Benefits

- **Business resilience**
Maintain business operation with continuous block-level data replication and the ability to **recover workloads within minutes** during disruption .
- **Streamlined disaster recovery**
Automate disaster recovery processes through an intuitive console, reducing complex manual configurations and minimizing the risk of human error.
- **Cost optimization**
Eliminates expensive secondary data centers and pay only for what you use, with minimal upfront investment no standby infrastructure costs.

Use cases

- **Healthcare and data protection**
Hospital systems could use Elastic Disaster Recovery to maintain **compliance** while protecting patient records by **replicating on-premises** servers to AWS. This ensures that critical medical data remains **accessible during system outages**. Regular disaster recovery testing helps validate their recovery procedures and meet strict healthcare regulations.
- **Financial services continuity**
A regional bank could implement Elastic Disaster Recovery to protect their core banking applications by continuously replicating their transaction processing systems. This facilitates quick recovery if their primary data center fails and helps **maintain customer trust** and regulatory compliance.
- **Manufacturing operation recovery**
A global manufacturer could employ Elastic Disaster Recovery to protect their production planning systems. By replicating factory management servers to AWS, they can ensure **minimal disruption to supply chain operations** during **disasters**. **Regular failover testing** validates the recovery strategy.

▼ Cloud in Real Life: Comparing Storage Services

Welcome back everyone to Cloud in Real Life. We've covered a lot of different AWS storage services so this time we're going to explore a few scenarios to help you understand when you should use them. Let's think back to the coffee shop this time. We have a **website** and it's currently being hosted using Amazon S3. Let's talk a little bit more about how that works.

Websites have a bunch of static assets and those are stored in an S3 bucket.

Exactly. We are using the **static website** feature of S3 to simplify the process.

S3 has a lot of different features and static website hosting is a really cool one for our learners to know about. Can you talk to me a little bit more about exactly how it works?

Oh yeah, sure sure. It's really straightforward. All you need to do is upload your website's files like HTML, CSS, JavaScript, and media files to an S3 bucket, enable static web hosting, and - booyah! You've got a website.

Nice, that does sound pretty simple, but as we expand our business we also have to consider **scalability**. As more people visit the website, we need to not only ensure we're using a performance solution, but also a **cost effective** one. And S3 has actually been great for this.

It has, you're right. And one of the best things about S3 is that it **automatically scales** to handle any amount of traffic without you lifting a finger. And you only **pay for the storage you use** and the **data transferred out of the bucket**.

So this is just one example of how the coffee shop can use S3. It's a really versatile service.

Oh yeah. So you know what, let's talk about another scenario. This time, let's say we are working on a solution for a **growing fitness center**. The owners launched a **mobile app** to help members book classes. Unfortunately the app isn't able to handle a recent ramp up in traffic, and customers are complaining. The business suspects the database is the bottleneck. Any thoughts on how to optimize this, folks?

Yeah so, well in this case the app is using a database that's running on Amazon **EC2 instances** and the app itself is pretty performance critical from what I understand.

Yeah, the most popular classes fill up fast so **low latency** is key here.

Yeah, exactly. Look, we need to be consistent with this **high performance** especially for the **database read and write** operations.

Yeah that's definitely the goal, and a **relational database** uses something like **EBS volumes** for data storage when running on **EC2 EBS** is designed exactly for this kind of use case.

Look that makes sense to me, but the app is still experiencing some latency when trying to read and write to that database. Alan, can you tell our learners about how to optimize for performance with EBS?

Oh sure. Basically, **EBS volumes** are optimized for **transactional workloads** but some volume types are more performant than others depending on your use case. So we can look at the type of EBS volumes currently being used and migrate to a more performant volume type.

Yeah exactly, and I would suggest they provisioned **IOPs SSD volume type** for this workload. It offers the highest performance for mission critical applications like this mobile app. Now, I know it's not always clear when to use S3 or EBS. So Rudy, can you explain why S3 isn't being used here?

Well something like **S3** is ideal for **object storage**, but it's not designed for the kind of **rapid continuous rewrite operations** that a database requires.

Exactly, **EBS** gives you that **block level access** which is essential for databases.

Okay, so now let's talk about one more scenario before we wrap up. This time let's think of a chain of automotive repair shops. The owners want to create an **internal tool** to improve the knowledge and efficiency of their mechanics. They've decided to

build a **collaborative platform** to **share** repair techniques and diagnostic procedures across **multiple locations**. It needs to store and provide **real-time access** to high resolution images videos and technical diagrams from **various devices and locations**.

And because we are talking about multiple devices and various locations, accessing the same data we really need a **file system**. So I am definitely thinking **EFS** is the right choice for this use case.

And EFS can **scale** to petabytes without disrupting applications. It **automatically** grows and shrinks as you add or remove files so it is good for working with large media files like these images and videos. **EFS** provides **low latency** performance and can handle **high levels** of aggregate **throughput** and **IOPs**, so it's really perfect for **media processing workflows** like this.

Nice, EFS seems like the perfect fit for this project.

I agree, it's **real-time access shared file systems** and **scalability** are exactly what we need here. All right so to recap, Amazon **S3** excels at **scalable object storage** for **web assets backups** and more. Amazon **EBS** provides **block level storage** needed for **EC2 instances** and **databases**. And Amazon **EFS** offers **managed shared file systems** from workloads that require **rapid simultaneous access to files**. Understanding these key differences will help you architect more efficient cost effective storage solutions in AWS. Until next time!

Module 7 - Databases

▼ Introduction to Databases

Well, we have quite a coffee operation going now.

We've got customers, lots of happy customers.

What we need now is some way to show our appreciation to our repeat customers.

How about a **loyalty program**?

Although we could just hand out punch cards, we can't track those well or use them to get to know our customers better.

We can use digital cards instead, but first, we need to be able to keep track of our customers, what they order, and how much they purchased.

This will help customers get the best rewards they've earned and help us to know our customer base better.

This means we need **databases**. And not just any database.

So, let's learn about the different services AWS offers to help you build the perfect database solution.

▼ Relational Databases

So we're storing data about our coffee shop in various systems. Now we need to store data for our loyalty program along with our transactional data to capture the relationships between the customers and their orders. And by relationships, I mean if a customer orders the same drink multiple times, maybe we send them a discount for their next order.

To do this, we need a way to keep track of this relationship somewhere. **Transactional data** is commonly kept in a **relational database management system**. This type of system stores data in a way such that it relates to other pieces of data.

For example, if we have a customer record, we store it in a customer table. We might also have an order record, which we store in a **corresponding** order table. We then **relate the tables** through a **common attribute**, and we can **query** data across both tables simultaneously.

The most common way to **interact** with a **database** is by **using Structured Query Language** or **SQL**. SQL runs on a variety of databases like MySQL, PostgreSQL, Microsoft SQL Server, and many more. If you have an **on-premises environment**, you're probably already running one of those databases, and they're most likely housed in your data center.

In fact, some companies are so fond of their database deployments that they want to keep them, but they still want to move to the cloud. So, can they do both? Of course they can! These companies can perform a **lift-and-shift**. This means taking their **existing database deployment**, lifting it up from their **on-premises environment**, and shifting it onto an **Amazon EC2 instance**.

They still have **control** over the same variables they did in their on-premises environment, such as operating system, memory, CPU, storage capacity, and so forth. All they've done is moved from one location to another, with the other being the AWS Cloud in this case. Cool stuff, right? Even cooler is that they can use a service called **AWS Database Migration Service** to help make the process smoother.

But as companies - and you - become more comfortable with the cloud, you might think, "Doesn't AWS have an option that makes database management even more seamless?"

The answer again is, of course! This option is called **Amazon Relational Database Service**, or **Amazon RDS**, and it offers **several database engines** you can use.

However, the key part here is that Amazon RDS comes with added benefits. These benefits include **automated patching, backups, redundancy, failover, and disaster recovery**, all of which you normally need to manage yourself. Amazon **RDS** is a popular option to AWS customers because it makes it possible for you to focus on business problems instead of maintaining databases.

After you deploy one RDS instance, you usually get a feel for it, sit down, relax, and your database administrators rejoice! Oh, and by using Amazon RDS, you also benefit from the **AWS shared responsibility model (SRM)**. This means you can take advantage of our **expertise in infrastructure management** while maintaining the **security** posture of your database.

We manage the underlying infrastructure, operating system, database software patching, and backup automation. In tandem, **you're responsible** for implementing **security** measures and configuring **encryption** for your **data at rest** and in **transit**. Teamwork makes the dream work, folks!

But we're not done yet. We can actually go one step further and **migrate** or **deploy** to **Amazon Aurora**. This is a **fully-managed relational database** option. It supports Postgre, MySQL, and distributed SQL databases, or DSQSLs.

And, get this, an **Aurora DB cluster** can include up to **15 Aurora Replicas located across Availability Zones** in the cluster's AWS Region. That's a lot of replicas! Additionally, **AWS Backup** supports **continuous backup** for **Amazon Aurora**. This allows specific **point-in-time restoration** within your retention period of **up to 35 days**.

Isn't it cool how everything at AWS integrates into each other and can remove that heavy lifting from your hands? Look I think it's cool. OK, my producer thinks it's cool too. Thank you, producer. And to you, our learners, that's relational databases in a nutshell.

Relational databases

Relational databases store data in a way that relates it to other pieces of data, and they use structured query language, or SQL, to manage and query data. This approach stores data in an easily understandable, consistent, and scalable way that works great for applications requiring structured data management.

AWS offers fully managed relational database solutions that remove the burden of database administration while maintaining high availability and security. AWS relational databases support popular database engines like MySQL, PostgreSQL, and Oracle, making it easier to migrate existing databases to AWS.

Amazon Relational Database Service (Amazon RDS)

Amazon RDS is a **managed relational database** service that handles routine database tasks such as backups, patching, and hardware provisioning. Amazon RDS **supports multiple database instance class** types that optimize for memory, performance, or input/output (I/O).



To improve data resilience, Amazon RDS offers Multi-AZ deployment and automated backups, but you can also manually create backups using DB snapshots. These are full backups of your entire database instance, which can be useful for specific point-in-time recovery or long-term data archiving purposes. Amazon RDS offers security features including network isolation, encryption in transit, and encryption at rest. You can readily scale database resources vertically or horizontally as needed.

Supported database engines

Amazon RDS supports different database engines, including Amazon Aurora, MySQL, PostgreSQL, Microsoft SQL Server, MariaDB, and Oracle Database.

Use cases

Some examples of practical use cases for Amazon RDS are web applications, enterprise workloads, and product inventories for e-commerce platforms.

- **Cost optimization**

Amazon RDS eliminates the high upfront costs of purchasing and maintaining database hardware infrastructure. You only pay for the compute and storage resources that you consume through a flexible pay-as-you-go model. As a managed service, it also reduces operational expenses by automating time-consuming administrative tasks like backups, patching, and monitoring.

- **Multi-AZ deployment**

Amazon RDS improves database reliability through Multi-AZ deployments. It automatically replicates data to a standby instance in a different Availability Zone. During system failures, maintenance, or zone disruptions, Amazon RDS automatically fails over to the standby instance without manual intervention. This ensures continuous database operations with minimal downtime.

- **Performance optimization**

Amazon RDS enhances database performance through automated management of resource allocation, monitoring, and optimization tasks. It includes features like automated backups and read replicas that can help offload read traffic from the primary instance. Amazon RDS performance insights provide real-time monitoring and analysis of database load, to help you identify and resolve performance bottlenecks quickly.

- **Security controls**

Amazon RDS enhances database security through multiple layers of protection, including VPC isolation as well as encryption at rest and in transit. It leverages automated backups and offers Multi-AZ deployments to provide resiliency against potential system failures.

Amazon Aurora

Aurora is a managed relational database designed to help reduce unnecessary I/O operations. It's compatible with MySQL and PostgreSQL, provides high performance and availability, and automatically scales alongside your workloads.



Aurora replicates data across multiple Availability Zones for enhanced durability and fault tolerance, and features automated backups, encryption at rest, and continuous monitoring. Aurora provides comparable performance to high-end commercial databases but at one-tenth the cost, which makes it ideal for organizations looking to reduce database costs without sacrificing performance.

Use cases

Some examples of practical use cases for Aurora are gaming applications, media and content management, and real-time analytics.

- **High performance and availability**

Aurora delivers up to five times the throughput of standard MySQL and three times the throughput of PostgreSQL. It uses a distributed storage system across multiple nodes to provide high performance and availability.

- **Automated storage and backup management**

Aurora automatically grows storage from 10 GB to 128 TB based on your actual data usage, which eliminates guesswork in capacity planning. It also continuously backs up your database to Amazon Simple Storage Service (Amazon S3) to provide point-in-time recovery.

- **Advanced replication and fault tolerance**

Aurora replicates data across three Availability Zones with six copies of data, and provides 99.99% availability. It automatically detects database failures and redirects traffic to healthy replicas without data loss.

Amazon RDS vs. Amazon Aurora

Feature	Amazon RDS	Amazon Aurora
Primary Use Case	Traditional workloads, "Lift and shift" migrations, or when you need a specific database engine (e.g., SQL Server, Oracle).	Cloud-native, high-performance applications, SaaS platforms, and workloads requiring instant scaling and global reach.
Architecture	Traditional. Similar to running a database on a server (EC2) but managed. Compute and storage are tied together.	Cloud-Native. Built from the ground up for the cloud. Compute and storage are decoupled , allowing them to scale independently.
Database Engines	Supports 6 Engines: 1. MySQL 2. PostgreSQL 3. MariaDB 4. Oracle 5. SQL Server 6. Amazon Aurora	Supports 2 Engines: 1. MySQL-compatible 2. PostgreSQL-compatible
Performance	Good. Faster than a standard EC2 installation but limited by traditional database bottlenecks.	High. Up to 5x faster than standard MySQL and 3x faster than standard PostgreSQL.
Storage	Manual/Semi-Auto. You choose the size (e.g., 500 GB). It can auto-scale up to 64 TB , but typically not down.	Fully Automatic. Scales up and down automatically in 10 GB increments up to 128 TB . You pay only for what you use.

Availability	Multi-AZ (Standby). You must enable "Multi-AZ" to get a standby instance. If the primary fails, the standby takes over (60–120 sec failover).	6 Copies by Default. Automatically replicates data 6 times across 3 Availability Zones. Failover is faster (< 30 sec).
Read Replicas	Up to 15.(some only 5) Replication is asynchronous and relies on "binlogs," which can cause lag (delay) during heavy write traffic.	Up to 15. Replication uses the shared storage layer, resulting in very low lag (typically single-digit milliseconds).
Cost Model	Generally Cheaper for stable, predictable, or smaller workloads.	Generally ~20% Higher base cost but can be more cost-effective for high-throughput workloads due to better performance efficiency.
Serverless Option	No true serverless. (Though RDS Proxy helps with connection management).	Aurora Serverless v2. Instantly scales compute (CPU/RAM) up and down based on live demand without dropping connections.

▼ NoSQL Databases Services

NoSQL databases

NoSQL databases are sometimes referred to as ***non-relational databases*** because their structures are different than relational databases like Amazon RDS. Instead of row and column relationships, NoSQL databases build a structure for the data that they contain using ***key-value pairs*** instead. With key-value pairs, data is organized into **items identified by unique keys**.

Each **key** has one or more associated **attributes**, or **values**, that represent various characteristics of the data. You can think of a key as a word entry in a **dictionary**, and the value as its associated definition. Not every item in the table has to have the same attributes, and you can add or remove attributes at any time.

Aw yeah! It's Amazon DynamoDB time! **DynamoDB is a fully managed serverless NoSQL database.** NoSQL means data is stored in a non-relational fashion instead of a relational database management system. Relational databases, like a standard MySQL database, require you to have a well-defined schema, which is one or more tables that might relate to each other. You then use SQL to query the data.

This works well for a lot of use cases, and has been the standard type of database historically. However, these types of **rigid SQL databases**, can have performance and **scaling issues** under specific circumstances. The rigid schema also means you can't always conveniently change that schema for any of the data you store in a table.

Therefore, it's not usually the best fit for evolving datasets that can have **varying attributes**. But that's fine. Since DynamoDB is non-relational, it gives you a **flexible schema**. You just create a table and start to store and query data.

Data is stored as **items**, and items are a collection of **attributes**. Each **attribute** has a **name** and a **value**. An attribute value can be **simpler** types like a **number** or more **complex** ones like a **set**, or a **document**. You can also add or remove attributes at any time, and **not every item** in the table needs to **have the same attributes**.

As for applications that need speed, DynamoDB offers **single-digit millisecond performance**. I mean that's pretty fast. And, you get **no cold starts, no version upgrades, no maintenance windows, no patching, and no downtime maintenance**. Nada.

You can even use DynamoDB for **globally distributed applications**. That particular feature is called **DynamoDB global tables**. An awesome example of this feature in use is Amazon Prime Day 2024. For those 48 hours, there were tens of trillions of calls to the DynamoDB API. It peaked at 146 million requests per second. There was no underlying management needed, and it scaled seamlessly. Now that's cool. I mean, seriously.

So, consider DynamoDB for your applications if you need the **speed, flexibility** of changing schemas, and **offloading of management**.

Amazon DynamoDB

DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance for both document and key-value data structures. It's a powerful and incredibly fast database option for use cases that require a flexible schema, and is ideal for **applications** that require **high performance and seamless scaling**.



DynamoDB seamlessly scales alongside your data without impacting performance, which means that you only pay for the resources that you use. It also includes built-in security features for enhanced protection, and **automatically spreads your data across multiple servers** to handle your workload.

Use cases

Some examples of practical use cases for DynamoDB are **gaming platforms, financial service applications**, and **mobile applications with global user bases**.

- Scalability with provisioned capacity

DynamoDB automatically scales throughput up or down based on actual usage, which ensures consistent performance without manual intervention. You can specify **target utilization levels**, and DynamoDB automatically provisions capacity to maintain those targets. With no practical limits on table size or the amount of data stored, DynamoDB can seamlessly accommodate growing applications.

- **Consistent high performance**

DynamoDB delivers **single-digit millisecond response** times at any scale, which makes it ideal for high-performance applications. It maintains consistent performance by automatically distributing data across multiple servers and SSDs.

- **High availability and durability**

DynamoDB delivers 99.999% data availability by **replicating** data across **three** distinct facilities within each AWS Region. It also maintains multiple copies in separate AWS Regions, to provide **built-in fault tolerance** and **data durability**. This ensures continuous operation and protection against data loss even if individual facilities fail.

- **Data encryption**

DynamoDB offers comprehensive **encryption** capabilities to protect information both **at rest** and **in transit**. All data is automatically encrypted behind the scenes before being written to the storage layer. DynamoDB includes the flexibility to **choose between different kinds of encryption keys** for customized security control.

▼ AWS Databases Demonstration

In this demo you'll explore a hands-on example of **configuring** two major database services: Amazon **RDS** and Amazon **Dynamo DB**.

Here we are in the AWS management console and to get started go to the search bar and enter *RDS*, and then select the service when it appears in the results. From there you'll find yourself in the Amazon **RDS dashboard**. Now you can choose *Create database* to get started. To **configure** this **database** we will choose **MySQL** for the engine options. Then scrolling down, we will select the free tier template since we're just trying things out.

Then scrolling down some more we will provide a **database instance identifier**.

You can enter in a meaningful name for this one but for this demonstration we will leave it as *database-1*. To access the database after it's created you need to **configure authorization** under the credential settings. For the **username** we will leave it to be admin and then for the **password** I will enter in a password.

Scrolling down some more we need to find the **connectivity** section, and from here you can select what sort of network you want to place this in. We'll leave the default VPC selected, but we do need to enable **public access**. So under *public access* I will choose yes.

This will allow us to easily connect to this database later in the demonstration using a **SQL client**. But in the real world, you likely would have this locked down so only **authorized applications** can connect.

Now we have all of our configurations in place, we will scroll to the bottom accepting the rest of the defaults, and then we will choose **Create database**. You're then redirected to the databases page to wait for the **RDS instance** to finish creating. It's ready to use when you see the status change from creating to available.

Now let's look at how to perform some basic commands in the **RDS MySQL instance** we just created. There are many different ways to connect to your RDS instance including a variety of clients. Setting up clients for connecting to the database is out of scope, but to set up the connection to the instance you would use the **endpoint** and **port values** shown on the **connectivity and security tab** of the **database-1** instance details page.

The screenshot shows the AWS RDS console with the 'Connectivity & security' tab selected for the 'database-1' instance. The summary panel displays the DB identifier (database-1), Status (Available), Role (Instance), Engine (MySQL Community), and Region & AZ (us-east-1d). The connectivity & security tab includes sections for Endpoint, Networking, and Security, providing detailed information about the VPC, subnet groups, and security groups.

Then to **authenticate** use the **username** and **password** that you defined when setting up the RDS instance. So using **that information**, we are **connected** to the **instance** and we can now **run** the following SQL statements to **create a database and 3 tables** in the RDS instance.

The screenshot shows the MySQL Workbench interface with the 'test-db' database selected. The left sidebar contains management, instance, and performance tabs. The main area shows the following SQL code being run:

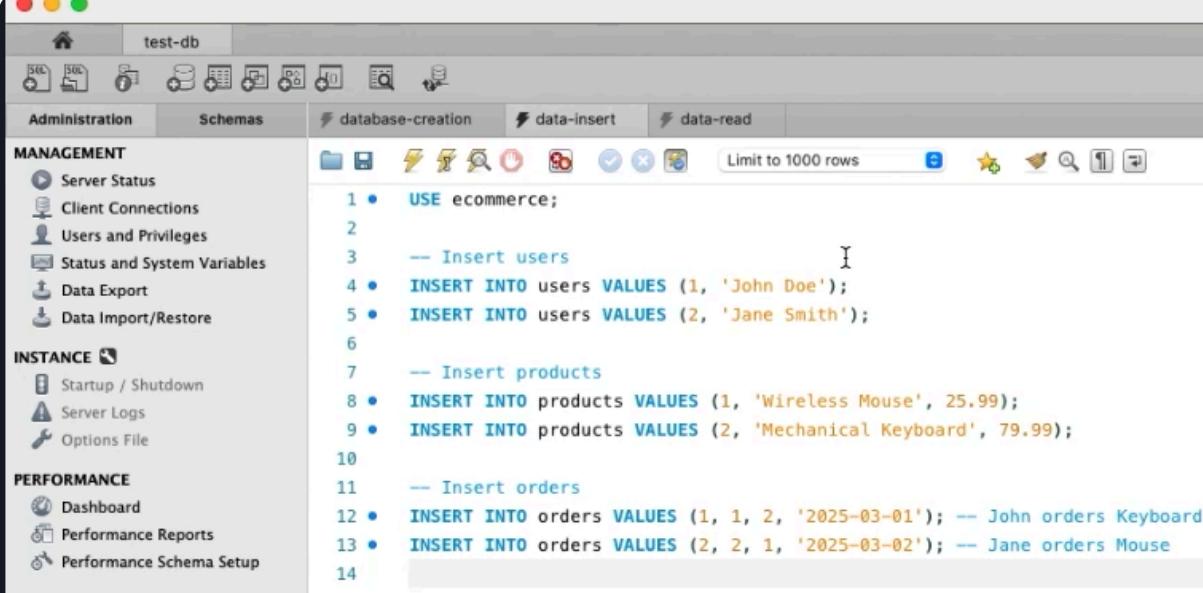
```

CREATE DATABASE ecommerce;
USE ecommerce;
CREATE TABLE users (
    id INT PRIMARY KEY,
    name VARCHAR(50)
);
CREATE TABLE products (
    id INT PRIMARY KEY,
    name VARCHAR(100),
    price DECIMAL(10, 2)
);
CREATE TABLE orders (
    id INT PRIMARY KEY,
    user_id INT,
    product_id INT,
    order_date DATE,
    FOREIGN KEY (user_id) REFERENCES users(id),
    FOREIGN KEY (product_id) REFERENCES products(id)
);

```

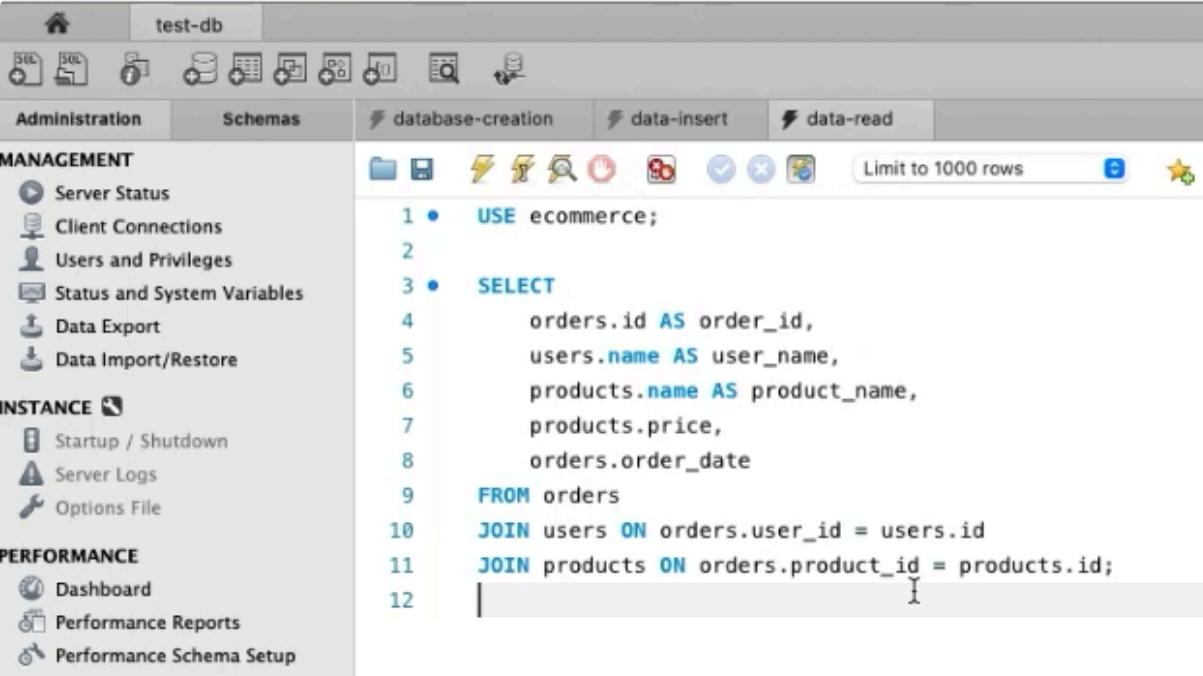
We are creating the **database first** and then in that database we are **creating 3 tables users, products and orders**. These **tables relate to each other through foreign keys**, so if we run this these tables have now been created.

Now let's run some **insert statements** to insert **data** into those tables. Running these statements will populate 3 tables with some data. One thing to note is that **every entry in one table has to have the same columns**, so the **schema** here is rigid.



```
1 • USE ecommerce;
2
3 • -- Insert users
4 • INSERT INTO users VALUES (1, 'John Doe');
5 • INSERT INTO users VALUES (2, 'Jane Smith');
6
7 • -- Insert products
8 • INSERT INTO products VALUES (1, 'Wireless Mouse', 25.99);
9 • INSERT INTO products VALUES (2, 'Mechanical Keyboard', 79.99);
10
11 • -- Insert orders
12 • INSERT INTO orders VALUES (1, 1, 2, '2025-03-01'); -- John orders Keyboard
13 • INSERT INTO orders VALUES (2, 2, 1, '2025-03-02'); -- Jane orders Mouse
14
```

Let's go ahead and run this, and now that we have some data in our tables, we can run this next statement to **retrieve data** from across the tables we just created. Running this statement **returns all orders a user placed joining together the data across the different tables into one result set**. Not too bad, right?



```
1 • USE ecommerce;
2
3 • SELECT
4     orders.id AS order_id,
5     users.name AS user_name,
6     products.name AS product_name,
7     products.price,
8     orders.order_date
9 FROM orders
10 JOIN users ON orders.user_id = users.id
11 JOIN products ON orders.product_id = products.id;
12
```

Now that you know how to create and interact with a table in a basic RDS instance, let's shift focus to another AWS database service **Dynamo DB**. Unlike Amazon RDS, DynamoDB is a **NoSQL database** that doesn't require managing instances or multiple tables within a single database. Instead, it has its **own query language** and uses **stand-alone tables**. There is no concept like a foreign key to relate tables to each other. You don't use SQL to query the data and it has a **flexible schema**. Let's explore this.

Starting here in the AWS management console, enter *DynamoDB* into the search and then select the service when it appears in the results. After you're in the DynamoDB dashboard, choose **Create table**. In the *table details* section enter the **table name "orders"** and then for the **partition key** we will choose **number** from the dropdown and we will give the **partition key a name - "order number"**. Then, for this demonstration, we will keep the rest of the defaults scroll down to the bottom and choose **Create table**.

This is all you need to create a table - just the table name and the partition key. Every **item** in this table has to have a **value** for the **partition key**, but otherwise the schema for each item can vary. It's ready to use once you see the status change from *creating* to *active*.

Now let's add some items to the Dynamo DB table we just created. To do that, I have a **Python script** created that will **load 10 items into this database**. This is the code for this task, and don't worry too much about how this works. Instead, just understand that in order for applications to interact with DynamoDB, they **invoke the DynamoDB APIs using the AWS SDK**. Let's give this a run using the terminal.

Python

```
import boto3
from botocore.exceptions import ClientError
from decimal import Decimal
#Create DynamoDB resource
dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
table = dynamodb.Table('Orders')
orders = [
    {"OrderNumber": 1, "user": "Example Customer 1", "total": Decimal("29.99"),
     "items": ["notebook", "p pen"], "status": "pending", "delivery": "standard",
     "OrderNumber": 2, "user": "Example Customer 2", "total": Decimal("54.10"),
     "items": ["headphones"] }, "status": "shipped", "delivery": "express", "promo": null,
     {"OrderNumber": 3, "user": "Example Customer 3", "total": Decimal("13.75"),
     "items": ["journal"], "st status": "pr processing", "g", "delivery": "standard",
     "prom": null
    {"OrderNumber": 4, "user": "Example Customer 4", "total": Decimal("89.00"),
     "items": ["t-shirt", "jeans"], "status": "delivered", "delivery": "express"
    {"OrderNumber": 5, "user": "Example Customer 5", "total": Decimal("19.99"),
     "items": ["book"], "status": "pending", " ", "delivery": "standard", "promo": False
    {"OrderNumber": 6, "user": "Example Customer 6", "total": Decimal ("47.89"),
     "items": ["water bottle", "mug' ug"], "status": "shipped", "delivery": "express"
    {"OrderNumber": 7, "user": "Example Customer 7", "total": Decimal ("22.5 "items": [
        ["pen", "planner"]
    ], "status": "processing", "delivery": "standard"
    {"OrderNumber": 8, "user": "Example Customer 8", "total": Decimal ("36.0 ,
     "items": ["backpack"], "status": "s pending", "delivery": "standard", "promo": null
    {"OrderNumber": 9, "user": "Example Customer 9", "total": Decimal("63.20"),
     "items": ["keyboard", " ", "mouse"], "status": "delivered", "delivery": "express"
    {"OrderNumber": 10, "user": "Example Customer 10", "total": Decimal("38.00"),
     "items": ["charger"] }, "status": "processing", "delivery": "standard", "promo" }
]

#Insert items
print(" Inserting orders into DynamoDB...")
for order in orders:
    try:
        I
        table.put_item(Item=order)
        print(f" Inserted Order #{order ['OrderNumber']}")
    except ClientError as e:
        print(f"X Failed to insert Order #{order ['OrderNumber']}: {e.response ['Error'] ['Message']}")
```

OK, now that the script is done running we should have 10 items in the table. Let's go check. So here we are back in the AWS management console. Let's select the table that we created and then we will run a **scan** on the table to see if those items loaded.

A **scan** will bring back all of the data in the table essentially performing a **read all type of query**. To do that choose **Explore table items** and then with the **scan selected**, choose **Run**. This will **display** the **results** from a scan that returns all of the **items** with the table entries listed down here under **items returned**. You can scroll over to see all of the information that was entered by that Python script.

Notice how the **notes** attribute is not present for every item in the table. This is an example of showing how **Dynamo DB allows for a flexible schema. Not every item in the table needs to have the same attributes**.

Now, what if your table starts getting larger and you want to bring back **only data for one specific order**? For this demo, we will run a **query** to do this. So selecting **query** and then providing the **order number**, which in this case we can say we want to pull back order number 5. We can then choose **run** and see that **only one item is returned**, which is order number 5.

All right there is of course always more to learn but that wraps things up for this demonstration of AWS database services.

▼ In-Memory Caching Services

As businesses experience **growth** in demand for their applications and services, it's not unusual for them to face **increasing pressure** on their **back-end relational databases**. Performance bottlenecks of this nature are common in databases, including Amazon RDS.

Bottlenecks can arise when dealing with **high read traffic volumes**, or when running **complex queries on large datasets**.

For example, think of an **e-commerce site** that uses Amazon **RDS** to store product information. When **thousands of customers repeatedly view the same product details**, the **database** must run that **same query over and over again**, returning the same results for each customer. This sort of **heavy read traffic for frequently accessed data** can overwhelm the database and lead to **latency and performance issues**.

This scenario isn't unique to e-commerce; many applications experience high volumes of read requests for **relatively static data**. There are many ways to improve performance to avoid these kinds of bottlenecks, and a particularly effective approach involves introducing a **caching layer** into the **architecture**.

Caching reduces the pressure on your **primary database** by **storing frequently accessed data in a high-speed, easily accessible location**. Data in a caching layer is commonly **stored directly in the system's memory instead** of on disk. This approach allows for **near-instantaneous access to data**, which can significantly **reduce the time** it takes to **serve requests to users**.

Additionally, **caching data reduces strain** on **backend databases** by **decreasing query volume**. Common tools for caching include **Redis OSS**, **Valkey**, or **Memcached**. **Amazon ElastiCache** provides a **fully managed cache** that is compatible with these tools to make caching on AWS more convenient.

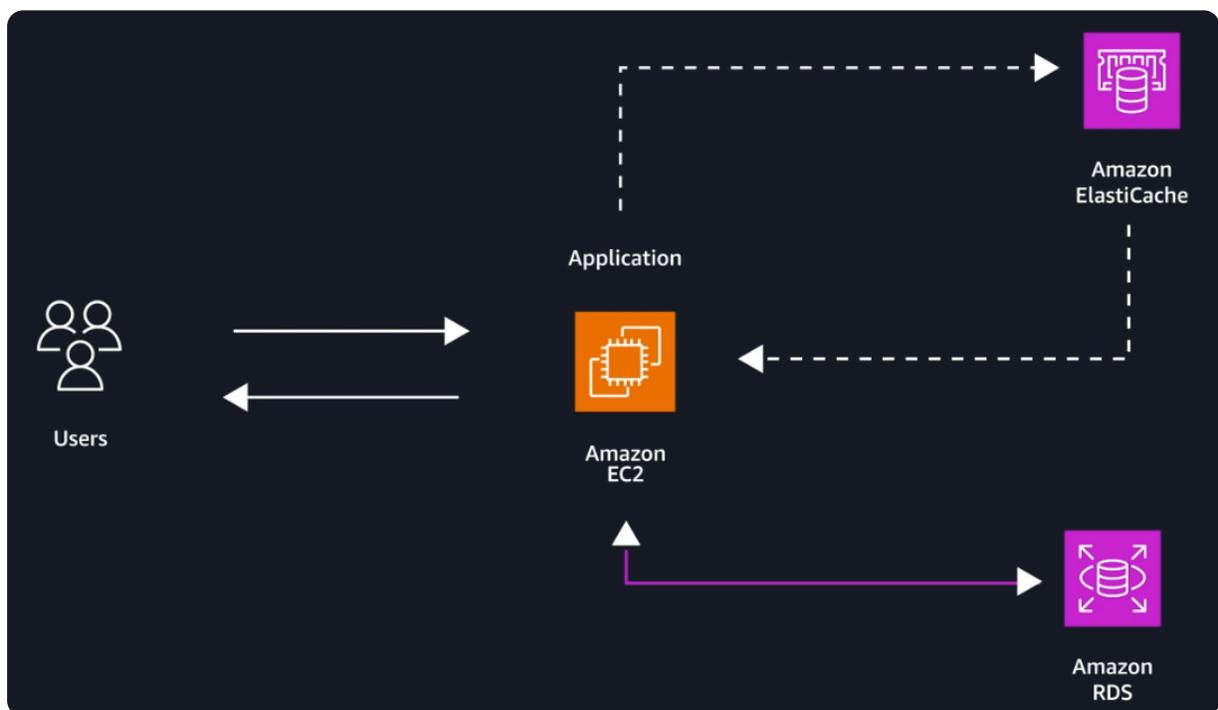
ElastiCache can significantly **improve application performance** by delivering **microsecond latency for data reads and offloading database queries**. ElastiCache is **flexible** and makes it possible for you to **adjust the cache size according to demand**, while AWS manages the complex infrastructure tasks, which reduce operational overhead. There's also a **serverless option** for **ElastiCache that scales with demand**.

And one of my favorite benefits of ElastiCache is that it can be used as a **cost-optimization tool** for your **database deployments**. By directing your applications to use the cache, you can potentially use **smaller, more economical database instances**. Pretty cool!

Now, ElastiCache is useful for a wide range of applications that require quick access to data. For example, if you're running a **gaming platform**, you can use **ElastiCache** to maintain **real-time analytics** and **update leaderboards** efficiently. Or, if you're managing a **content delivery system**, you can use it to **store frequently accessed data** and ensure swift delivery to **end users**.

To better understand how ElastiCache fits into a typical **AWS architecture**, let's review a common setup involving **Amazon RDS**, **Amazon EC2**, and **ElastiCache**. In this **architecture**, an **EC2 instance hosts your application servers** while **Amazon RDS** provides the **backend relational database**. ElastiCache sits alongside the application and the database.

When a **user requests** data from your **EC2 application**, the application first **checks ElastiCache**. If the **data exists** in the cache, it is **immediately returned to the user**. However, if the **data can't be located in cache**, the application will **read the data from RDS**, **store it in cache**, then return it to the **user**. This way the **next time the data is requested**, it's already in the **cache** and can be **retrieved quickly**.



By reducing server load, providing **consistent performance** with **microsecond latency**, and **scaling efficiently to meet** your needs, **ElastiCache** is a great way to level-up your databases.

Amazon ElastiCache

ElastiCache is a **fully managed in-memory caching** service that was built to help reduce the complexity of **administering in-memory caching systems**. This means that you can continue to use the same Redis, Valkey, or Memcached tools and configurations to scale your workloads. It **automatically detects and replaces failed nodes**, which makes it ideal for applications that need consistent high performance.



Use cases

Some examples of practical use cases for ElastiCache are **session data management**, **database query enhancement**, and **gaming leaderboards**.

In-memory caches

An in-memory cache is a high-speed storage layer that *temporarily stores frequently accessed data* in a computer's main memory, or RAM. Retrieving data from RAM provides extremely fast processing and retrieval speeds, often hundreds or thousands of times faster than traditional disk-based storage systems. In-memory caches are ideal for storing session data, API responses, database query results, and other information that applications require repeatedly.

Benefits

- **High performance for Redis, Valkey, or Memcached instances**

ElastiCache streamlines the deployment and maintenance of in-memory caching **environments**, offering high availability for Redis, Valkey, and Memcached by automatically handling hardware provisioning, software patching, and monitoring. ElastiCache offers **seamless scalability** so you can add or remove nodes as demand changes.

- **High availability**

ElastiCache provides high availability by constantly monitoring primary nodes for potential failures. When issues are detected, it maintains application availability while promoting a replica node to become the new primary without manual intervention. The entire recovery process typically finishes within minutes, which minimizes downtime and preserves operations during infrastructure disruptions.

- **Replication across multiple Availability Zones**

ElastiCache enables automatic replication across multiple Availability Zones to protect against infrastructure failures. You can configure primary and replica nodes across different Availability Zones according to their durability requirements. This helps to ensure that data remains accessible even if one zone experiences an outage.

- **Data encryption**

ElastiCache supports data encryption mechanisms to safeguard sensitive information throughout its lifecycle. **At-rest encryption** protects data while stored in disk storage and **automated backups**. **In-transit encryption** secures data traveling between **clients and cache nodes** by employing transport layer security, or **TLS**, for encrypted connections.

ElastiCache

AWS CCP

Related Project



Cloud Certified Practitioner Essentials note

Related Ideas



IT common knowledge

You cannot use just ElastiCache without choosing either Redis (or Valkey) or Memcached.

Amazon ElastiCache is not the database or the hardware itself; it is the **fully managed service layer** that automates the setup, management, and scaling of an in-memory database or caching solution, which must run on a specific engine.

ElastiCache: The Management Tool (The "How")

ElastiCache acts as the "Database Administrator (DBA) as a Service" for popular in-memory data stores.

- **It is the management platform, not the engine.** ElastiCache provides the administrative automation for the caching layer.
- **The Physical Hardware (Nodes):** When you launch an ElastiCache Cluster, AWS automatically provisions **EC2 instances** (called **Cache Nodes**) with network-attached, high-speed RAM. You are billed hourly for these nodes.
- **The Management Tasks it handles:**
 - Hardware provisioning.
 - Operating system and software patching.
 - Configuration and cluster monitoring.
 - Failure detection and automatic recovery/replacement of failed nodes.
 - Scaling (both vertical and horizontal, especially with ElastiCache Serverless).

Redis or Memcached: The Engine (The "What")

You must select a compatible engine when you create an ElastiCache cluster because that engine provides the actual **data structure, protocol, and functionality** that your application interacts with.

- **The Engines:**

- **ElastiCache for Redis (or Valkey):** This is the more feature-rich choice, supporting complex data structures (Lists, Sets, Hashes, Geospatial), persistence, replication, transactions, and Pub/Sub messaging. It can be used as a primary **in-memory database** or a cache.
- **ElastiCache for Memcached:** This is the simpler, highly performant choice, supporting only a basic **key-value store** (strings). It is purely a caching layer.

Can I Use ElastiCache for my DBs?

Yes, you can use ElastiCache as your primary data store, but only if you choose the Redis engine (or Valkey).

While Memcached is strictly for caching (data is volatile and not persisted), **ElastiCache for Redis** offers:

1. **Persistence:** The ability to save data to disk (AOF or RDB snapshots), making it a viable **in-memory database** for certain use cases.
2. **Replication:** Supports primary and replica nodes for high availability and read scaling.
3. **Complex Use Cases:** Ideal for session management, leaderboards, real-time analytics, and message brokering.

However, even when using ElastiCache for Redis as a database, it's typically used as a **fast front-end** alongside a traditional persistent database (like RDS, DynamoDB, or Aurora) that serves as the **source of truth** for all your data.

To learn more about the architecture and benefits of using ElastiCache to offload your primary database, you can watch this video.

[Boost your Primary Database Performance with Amazon ElastiCache](#)

http://googleusercontent.com/youtube_content/9



Boost your Primary Database Performance with Amazon ElastiCache - AWS Databases in

15

https://www.youtube.com/watch?v=6qiU_R-TN0k

Database Comparison: With vs. Without ElastiCache

CCP

Metric	Without ElastiCache (Direct DB Access)	With ElastiCache (Cache-Aside Pattern)
Primary Workload Handled	All reads and writes	Writes and occasional reads (cache misses)
Database Instance	Larger instance required to handle	Smaller, more economical instance (read load absorbed)

Size	peak read/write load.	by cache).
Application Read Latency	Higher (requires disk I/O, often 5-10+ milliseconds).	Ultra-Low (served from RAM, often less than 1 millisecond).
Database Read IOPS/Load	Very High load, as the database processes every single read request.	Low load, as the database only serves read requests when data isn't in the cache (cache misses).
Total Deployment Cost	Higher (driven by the cost of the large database instance).	Lower overall (the cost of the smaller database instance plus the lower cost of the ElastiCache nodes).
Read Scalability Limit	Limited by the maximum scaling capacity of the database instance.	Vastly improved (reads are scaled horizontally by adding more ElastiCache nodes).
Data Freshness (Best Case)	Perfect consistency (always reads the source of truth).	Minor delay possible if data is being updated while old data remains in the cache (TTL issue).
Hardware Used	Persistent storage (SSD/Disk)	In-memory RAM (for speed)

▼ Additional Database Services

Before we wrap up databases let's loop back to the topic that started all this: choosing the right database to fit your business needs, rather than forcing your data to fit your database's requirements. There is no one-size-fits-all database for all purposes. AWS recommends purpose-built databases for specific workloads.

We've covered quite a few database flavors already, but there are even more databases AWS offers for special business requirements. We don't have time to cover them all, but it's worth knowing about them in case you need them.

For example, Amazon **DynamoDB** is great for **key-value pair databases**, but what if instead of basic keys and values, you need your database to support semi-structured data?

That is, data with complex attributes that don't fit neatly into the rows and columns of a relational database.

Amazon DocumentDB

Amazon DocumentDB (with **MongoDB compatibility**) is a fully managed service designed to handle **semistructured data**, which is information that doesn't conform to rigid relational schemas. Amazon DocumentDB is a MongoDB-compatible database, so it manages **JSON-like documents** with dynamic schemas.



Amazon DocumentDB is perfect for applications requiring frequent schema changes and document-oriented data. Unlike relational databases or nonrelational databases, you can **quickly iterate** without relying on predefined schemas. Amazon DocumentDB can **store, query, and index JSON data** effortlessly, all while benefiting from **automatic scaling, continuous backup, and enterprise-grade security features**.

One can use Amazon **DocumentDB**, a specialized database service that helps manage data with complex, varied information that doesn't fit neatly into traditional spreadsheet-like tables. It's great for uses like **content management, catalogs, and user profiles**.

Use cases

Some examples of practical use cases for Amazon DocumentDB are **content management systems, catalog and inventory management, and user profile** and personalization systems.

- **MongoDB compatibility**

Amazon DocumentDB is fully compatible with **MongoDB** workloads and supports MongoDB APIs, drivers/**queries**, and tools. This compatibility means that you can use **existing MongoDB code** and skills without modification. You can also migrate MongoDB applications to Amazon DocumentDB with minimal changes to their application code.

- **Performance and scalability**

Amazon DocumentDB **automatically scales** storage up to **64 TB in 10 GB increments** based on your application needs. It can handle **millions of requests per second** with consistent performance. It also provides the option to **scale compute resources** up or down as needed.

- **Increased read throughput**

Amazon DocumentDB improves **read throughput** for high-volume applications by creating up to **15 replica instances** that share underlying storage.

Amazon Neptune

Neptune is a **fully managed**, purpose-built **graph database** service that manages highly connected data sets, like those used in **social networking** applications. It excels at understanding **complex relationships** that are difficult to identify in traditional relational databases like **user connections, friend networks, and interaction patterns**. Neptune can maintain high performance even as data complexity grows, and offers **high availability with automatic failover and backups**.



What if there's a **social network** that you want to track? Social webs that identify **who is connected to whom** are very clunky to manage in a traditional relational database.

Amazon Neptune was created to solve this problem. It's a **graph database** designed specifically to store and manage **interconnected data**, making it convenient to **query social networking data** to find **relationships and patterns**. It's also a great tool for **fraud detection**.

Use cases

Some examples of practical use cases for Amazon Neptune are social network user connection mapping, fraud detection systems, and **search and recommendation systems**.

- **Purpose-built for complex relationships**

Neptune excels at storing and querying highly connected data using graph models. It supports both **property graph and resource description framework**, or **RDF**, models making it ideal for relationship mapping and **pattern matching applications**.

- **High performance and scalability**

Neptune delivers consistent performance at scale, **processing billions of relationships in milliseconds**. It **automatically grows** storage up to **64 TB** based on your application needs. Its purpose-built **engine optimizes graph queries** to enable **fast traversal of connected data points** at scale.

AWS Managed Blockchain

Perhaps you have a **supply chain** that you have to **track** with **assurances** that nothing is lost. Think of a grocery store **maintaining data** on **shipments** from food suppliers to help ensure food safety. For this type of scenario we offer **Amazon Managed Blockchain**, a service that helps you create and manage **blockchain networks**.



AWS DynamoDB Accelerator

Now databases by themselves are great but what if there was a way to make them faster? Wouldn't that be greater? AWS offers database accelerator options that can be used in a number of unique scenarios. For example, if you're using DynamoDB, try using the **DynamoDB Accelerator**, or **DAX**, a **built-in caching layer** designed to dramatically improve **read times** for your **nonrelational data**.

AWS Backup

AWS Backup **streamlines data protection across various AWS resources** and **on-premises deployments** by providing a **single dashboard for monitoring and managing backups**. It eliminates the complexity of managing multiple backup strategies by supporting multiple storage types, including Amazon Elastic Block Store (Amazon **EBS**) **volumes**, Amazon Elastic File System (Amazon **EFS**) file systems, and various **databases**.



AWS Backup **centralizes** and **automates** data protection processes, improving consistency and **reducing administrative overhead**. It offers flexible **scheduling options**, **encryption capabilities**, and **cross-Region backup support** for enhanced **disaster recovery**.

Use cases

Some examples of practical use cases for AWS Backup are centralized disaster recovery, consistent backup policies for **compliance requirements**, and **consolidating multiple backup processes** through a single interface.

Benefits

- Centralized backup management

AWS Backup provides a single dashboard to *manage backups across multiple AWS services and accounts*. You can **monitor backup jobs**, **restore points**, and **verify compliance status** from one central location to reduce operational complexity and potential configuration errors.

You can create **automated backup schedules** that align with your business requirements and compliance needs. You can set up **backup policies** that automatically protect **new resources as they're created**.

- **Cross-region backup redundancy**

AWS Backup enables **automatic replication** of backup data **across different AWS Regions** for disaster recovery purposes. You can quickly restore data from secondary Regions if the primary Region experiences an outage. Cross-Region redundancy helps you meet **compliance** requirements while guaranteeing data accessibility during Regional failures.

- **Streamlined regulatory compliance**

AWS Backup **maintains detailed audit logs and reports** to demonstrate **compliance** with **regulatory** requirements. You can use it to **enforce backup policies** across your organization and track backup activities for security and compliance purposes.

There's one more important service that's related to databases and storage in general: **AWS Backup**. When planning for data backup, it's often a discussion of backing up the disks -- in this case, EBS volumes -- and backing up specific data in a structured way as it is stored in databases. And with all the types of databases at AWS this can lead to overlapping and confusing backup strategies and methods. AWS Backup simplifies this by supporting EBS volumes, EFS filesystems, RDS Databases, DynamoDB Tables and more-- even encompassing backup of data stored outside of AWS, say in an **on-premises deployment**.

AWS EMR

AWS EMR, or **Elastic MapReduce**, is a **managed** web service that simplifies running **big data frameworks** like **Apache Hadoop** and **Apache Spark** on AWS to process and analyze **large datasets**. It **automates** tasks like **cluster provisioning** and **tuning**, making it easier to **use open-source tools** for tasks like **machine learning**, **log file analysis**, and **web indexing**. EMR **separates** data storage (using Amazon **S3**) from compute (using Amazon **EC2** instances) to allow for **scaling** and **on-demand use**.



How it works

- **Managed service:**

AWS handles the provisioning, management, and maintenance of the infrastructure and software for the big data clusters, including tasks like capacity provisioning and cluster tuning.

- **Open-source frameworks:**

It leverages popular open-source big data tools such as Apache Spark, Apache Hive, Apache HBase, Apache Flink, and Presto.

- **Scalable infrastructure:**

It uses Amazon **EC2** instances to create resizable clusters for **distributed processing**, and data is stored in Amazon **S3**.

- **Job execution:**

You can launch a cluster with specific frameworks and then run jobs on it by connecting to the master node or by sending steps (jobs) through the AWS console.

Key features and use cases

- **Big data processing:** EMR is designed for processing massive datasets, with the ability to run petabyte-scale analyses.
- **Machine learning:** It supports various machine learning tasks and frameworks.
- **Interactive analytics:** You can perform interactive analytics on large datasets.
- **Data pipeline automation:** You can automate data processing pipelines, for example, by creating steps that process data stored in S3.
- **Security and access control:** It integrates with AWS Identity and Access Management (IAM) to control user permissions and uses security configurations to manage encryption and authentication.

When to use AWS EMR

- You need to **process large volumes of data** using frameworks like Hadoop or Spark.
- You want to **automate** big data **environment** setup, operation, and scaling.
- You need to perform tasks like **machine learning**, **log analysis**, or **web indexing** on a large scale.

The key thing to understand is that AWS wants to make sure that you are using the best tool for the job.

A key benefit of **AWS Database Migration Service - DMS** is that the source database remains fully operational during migration, which minimizes downtime to applications.

AWS Databases Table

AWS CCP

Database Service	Database Type/Category	Key Features / Distinction	Primary Use Case
Amazon Aurora	Relational (OLTP)	MySQL/PostgreSQL compatible. 5x faster than standard MySQL.	High-performance, global-scale transactional systems

		Storage is self-healing, fault-tolerant, and autoscaling (up to \$128 \text{ TB}\$\$).	like SaaS and e-commerce.
Amazon RDS	Relational (OLTP)	Fully managed service for six popular database engines (PostgreSQL, MySQL, Oracle, etc.). Automates patching, backups, and monitoring.	Standard, established relational workloads, enterprise resource planning (ERP), and applications requiring complex joins.
Amazon Dynamo DB	Key-Value / Document (NoSQL)	Serverless and highly scalable. Guarantees single-digit millisecond latency at any scale. Supports Global Tables for multi-region replication.	High-traffic web/mobile backends, gaming, ad-tech, and microservices requiring extreme scale and low latency.
Amazon Redshift	Data Warehouse (Analytics)	Uses columnar storage and Massively Parallel Processing (MPP) for high-performance analytics. Stores data in managed clusters.	Large-scale analytical processing (OLAP), business intelligence, and complex reporting on petabytes of structured data.
Amazon Neptune	Graph	Supports Gremlin and SPARQL query languages. Optimized for fast traversal of highly connected data.	Fraud detection, social networking, recommendation engines, and knowledge graphs.
Amazon DocumentDB	Document (NoSQL)	Fully managed, scalable document database that is MongoDB API compatible .	Content management, catalogs, user profiles, and applications migrating from MongoDB.
Amazon ElastiCache	In-Memory (Cache)	Managed Redis and Memcached. Provides microsecond-level latency for fast data retrieval.	Session stores, real-time leaderboards, and general-purpose caching to speed up other databases.
---	---	---	---
Amazon EMR (Elastic MapReduce)	Big Data Processing	Managed cluster platform for running open-source frameworks like Apache Spark, Hadoop, and Hive. Optimized for complex ETL and data transformation.	Large-scale data transformation (ETL), scientific simulation, and processing vast amounts of unstructured or semi-structured data.
Amazon Athena	Interactive Query	Serverless service that lets you analyze data directly in Amazon S3	Ad-hoc querying and exploration of unstructured data in

		using standard SQL. Charged per data scanned.	your data lake without needing to load or manage clusters.
Dynamo DB Accelerator (DAX)	Caching Layer	A fully managed, in-memory cache built specifically for DynamoDB. Reduces read latency from milliseconds to microseconds.	Read-intensive applications using DynamoDB that require microsecond latency (e.g., high-volume product catalogs).
AWS Managed Blockchain	Blockchain Network	Fully managed service for creating and managing decentralized blockchain networks using open-source frameworks (like Hyperledger Fabric).	Applications requiring multiple untrusted parties to securely and transparently share data and transactions (e.g., supply chain tracking).
Amazon QLDB (Quantum Ledger Database)	Ledger Database	Immutable, centralized, and cryptographically verifiable transaction log (journal) of all application data changes.	Financial records, regulatory compliance, auditing, and systems that require indisputable proof of data lineage.
AWS Backup	Data Protection	Centralized, managed backup service for setting up backup policies and consolidating backup and recovery across multiple AWS services (RDS, EBS, EFS, DynamoDB, EC2, etc.).	Unified, automated compliance and data retention management for all supported data stores.

Module 8 - AI/ML and Data Analytics

▼ Introduction to AI and Machine Learning

And we're back in the coffee shop! Each day, we see new and familiar faces while noticing the time when rush hours begin and end. We track which pastries and coffee blends fly off the shelves, and we keep an eye out for which coffee trends are the most popular.

What if we somehow tracked and analyzed all of this information and data? Could we use it to accurately predict what our customers might want next? Could we determine how many coffee beans to purchase for next month? Or even predict future demand for new coffee shop locations around the world? Over time, we could even use all of this data to invent entirely new drinks and menu items based on what we've learned. Of course, to do all this, we need a clean and reliable way to gather and process all of the relevant data points. Otherwise, we're just guessing.

AI

Artificial Intelligence is a broad field focused on the **development** of intelligent computer systems capable of **performing humanlike tasks**.

This is where artificial intelligence and machine learning can be a huge boost for businesses. Artificial Intelligence, or AI, is a broad field focused on the development of intelligent computer systems capable of performing humanlike tasks.

ML

Machine learning is a type of AI for training machines to **perform complex tasks without explicit instructions**. Machine learning training finds the **patterns** hidden in vast amounts of historical data to produce an **ML model**. This ML model can then be applied to new data to make **predictions** or **decisions based on the patterns it's learned**.

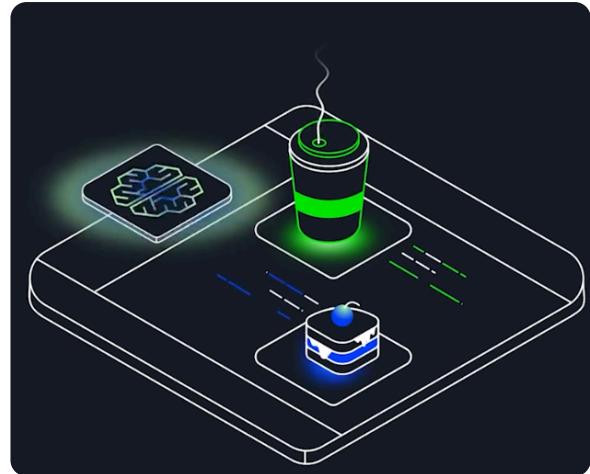
Machine learning, or ML, is a type of AI for training machines to perform complex tasks without explicit instructions. For instance, we're creating an app so customers can order their coffee online. We want our app to **recommend food items** based on each **customers' order history** to go along with their coffee purchase. This is a prime use case for machine learning!



Now, without machine learning, using **classical programming**, you would typically set up **predefined rules** based on **established relationships** between different products. For example, previous customers who purchased a caramel latte often purchased a cheese Danish. You would create a rule that recommends a cheese Danish with any new caramel latte purchase. This rule might prompt some customers to purchase the recommended cheese Danish, but it doesn't really account for the individual customer or product.



By contrast, **machine learning** finds the **patterns** between customers and sales **hidden in vast amounts of historical data**. This process is known as **training**. At the end of the training process, an **ML model** is created using the **patterns** that were found.



This model can then be used for **inference**, which means you give it **new data** and have it **make predictions or decisions based on the patterns** that it's learned.

So, taking our trained ML model, it is then applied to new data like customer purchases for a more relevant and **personalized food recommendation without explicit rules**. Recommendation or personalization engines like this are major use cases for machine learning.

Now, our customers can use our **ML-powered app** to order their coffee ahead of time and receive food **recommendations** that are tailored to their **personal preferences**. Our customers are happy, and our coffee shop sells more food items. Definitely a win-win!

So that is an example of using machine learning to improve business outcomes. But, let's talk about how we can use AI to innovate in other parts of our business, as well.

Imagine if we installed an AI-powered kiosk at the counter. As each customer steps up, the kiosk greets them, understands their spoken order, and enters it directly into the register for the baristas to fulfill. Using **natural language processing**, or **NLP**, the kiosk could even **ask clarifying questions**. For example, "Would you like an extra shot of espresso?" or "Are you interested in trying our new seasonal latte?" This kiosk isn't just rule-based. It's a **conversational AI** that can **change and adapt** to customer responses in **real time**.

Meanwhile, a separate AI system could **analyze and combine local coffee trends on social media** with our **own historical sales data**. It could then **predict** which drinks might be popular next week, or even generate new recipe ideas for our baristas to experiment with.

Those are just two examples related to our coffee shop. The technology behind AI continues to progress and it can help us innovate across different types of businesses.

AWS offers a variety of different services for machine learning and AI. You can **create**, **train**, and **deploy** your **own custom ML models** using tools like **Amazon SageMaker AI**. You can also choose from AI services that use **pre-built models** for **common tasks** like **language translation** or **image recognition**. By using these services on demand, businesses of all sizes can take advantage of AI without having to spend large amounts of time and money building everything from scratch.

But keep in mind that AI models are trained on data that can come from all sorts of places. These might include sales records, social media posts, sensor readings, customer feedback forms, and more. You'll need to gather that data, clean it up, and make sure it's in the **right format**. Without the **right data** and processes, your AI's predictions won't be very accurate.

▼ AI/ML on AWS

Businesses of all types have been using artificial intelligence and machine learning for years. For example, at Amazon, **ML models** power our **ecommerce recommendations** engine and **fulfillment center optimizations**, among many other areas.

But **ML models** can help solve a variety of problems outside of ecommerce. For instance, they can help make **predictions** about **stock market trends** or the **price of goods**. They can help route a caller to the right department based on a voice-prompt request. They can even detect **financial anomalies**, such as **fraud**, in online-banking systems.

Let's discuss the **AWS AI/ML stack**. It's composed of three tiers: **AWS AI services**, **AWS ML services**, and highly **customizable solutions** using **frameworks** and **infrastructure**.

AWS AI services are **managed services**. They provide you with access to **pre-built models** that are **already trained** to perform **specific functions**. These include **Amazon Polly** for **text-to-speech generation**, and **Amazon Comprehend** for **text or sentiment analysis**.

For use cases that require more customization, **AWS ML services**, such as **Amazon SageMaker AI**, can be the way to go. With SageMaker AI, you can **build**, **train**, and **deploy** your own **ML models** using **fully managed infrastructure**, **tools**, and **workflows**. SageMaker AI even helps you **track model training experiments**, **visualize data**, and perform model **debugging** and **monitoring** all within a **single environment**.

Sometimes though, your business requires a highly **specialized ML solution**. In these cases, you can choose a completely custom ML modeling approach by **using AWS frameworks** and **infrastructure**. This includes using **purpose-built chips** that **integrate** with popular **ML frameworks**. You can even build a solution hosted on an **ML-optimized EC2 instance**.

In the following lessons, we'll dive into the world of AWS AI/ML. And we'll even explore the emerging field of generative AI and the AWS services that make it all possible.

▼ AWS AI/ML Solutions

Let's examine the three tiers of the AWS AI/ML stack in more detail. You can think of the tiers as a progression from **pre-built, easily deployable managed solutions** to highly **customized** solutions that require more skill to implement.

Tier 1: Pre-built AWS AI services

The AWS AI services tier is made up of pre-built models that are already trained to perform specific functions. These ready-to-use, managed services can help you quickly solve for a variety of business use cases. In the next section, you will learn about the following three groups of AWS AI services:

- Language services
- Computer vision and search services
- Conversational AI and personalization services

Language services

AWS AI language services are great for when you need to **interpret text or speech** and **transform** it into something **meaningful**. Let's examine how these services solve for some common use cases.



Amazon Comprehend

Amazon **Comprehend** uses natural language processing to extract **key insights** from **documents**. It develops these insights by recognizing **key phrases, language, sentiment**, and other common elements in documents.

Use cases: Content classification, customer sentiment analysis, and compliance monitoring



Amazon Polly

Amazon Polly **converts text into lifelike speech**. It supports multiple languages, different genders, and a variety of accents.

Use cases: Virtual assistants, e-learning applications, and accessibility enhancements for visually impaired users



Amazon Transcribe

Amazon Transcribe **converts speech into text**. It supports multiple languages and offers features such as **speaker identification**, custom vocabulary, and **real-time transcription**.

Use Cases: Customer call transcription, automated subtitling, and metadata generation for media content.



Amazon Translate

Amazon Translate is a text translation service. This service is ideal for global communication because it supports real-time and batch text translation across multiple languages.

Use cases: Document translation and multi-language application integrations



Computer vision and search services

These services are ideal for **answering questions** and **gathering insights** from various types of content sources such as documents, images, videos, and more. Let's look at some examples.

Amazon Kendra

Amazon Kendra uses **natural language processing** to **search for answers** within **large amounts of enterprise content**.

Because it understands the context of a **query**, it can return more **precise** and **relevant answers** than just a list of documents with matching keywords.

Use cases: Intelligent search, chatbots, and application search integration



Amazon Rekognition

Amazon Rekognition is a **video analysis** service. It can **identify objects, people, text, scenes, and activities** within images and **videos** stored in Amazon Simple Storage Service (Amazon S3).

Use cases: Content moderation, identity verification, media analysis, and home automation experiences



Amazon Textract

Amazon Textract detects and **extracts typed and handwritten text** found in **documents** (including scanned documents in image formats), **forms**, and even **tables** within documents. Preserves table structure (rows, columns, cells).

Use cases: Financial, healthcare, and government form text extraction for **quick processing**



Conversational AI and personalization services

With these services, users can **interact** with your **apps** through **text and voice conversations**. You can also present your customers with **product recommendations** personalized just for them. Let's explore these services.

Amazon Personalize

Key Features

- ML service for building personalized recommendations.
- No ML expertise required for deployment.
- Integrates with applications via API.
- Optimized for real-time personalization.

Common Use Cases

- Recommending products to e-commerce customers
- Personalizing content for streaming platforms
- Suggesting relevant articles in news apps



Amazon Lex

With Amazon Lex, you can **add voice and text conversational interfaces to your applications**. This service uses both **natural language understanding (NLU)** and **automatic speech recognition (ASR)** to create **lifelike conversations**.

Use cases: Virtual assistants, natural language search for FAQs, and **automated application bots**



AWS pre-built (managed) AI services

AWS CCP

Service	What It Does / Key Features
---------	-----------------------------

	Category / Use Case	
Amazon Bedrock	Generative AI	Gives access to foundation models (LLMs, FMs) so you can build generative AI apps (chatbots, summarization, etc.) without managing your own model infrastructure. (Amazon Web Services, Inc.)
Amazon Q	Generative AI / Assistant	An AI-powered assistant for work, built on top of Amazon's foundation models, that can answer questions, summarize documents, help with cloud-ops. (Wikipedia)
Amazon Comprehend	Language / NLP	Natural Language Processing: sentiment analysis, entity recognition, topic modeling, language detection. (Medium)
Amazon Comprehend Medical	Language / Healthcare NLP	Extracts medical information from clinical text: medical conditions, medications, relationships, PHI (protected health info). (AWS Documentation)
Amazon Lex	Conversational Interfaces / Chatbots	Build chatbots / conversational UI (text + voice). Uses NLP to understand intent, manage dialog. (AWS Documentation)
Amazon Polly	Speech (Text → Speech)	Converts text into realistic, lifelike speech (supports many languages / voices). (Amazon Web Services, Inc.)
Amazon Transcribe	Speech (Speech → Text)	Automatically converts speech audio into text (real-time & batch). (Amazon Web Services, Inc.)
Amazon Translate	Language Translation	Neural machine translation between many languages. (Amazon Web Services, Inc.)
Amazon Rekognition	Computer Vision	Pre-trained image/video analysis: object recognition, face detection, moderation, text in images. (Wikipedia)
Amazon Textract	Document / Text Extraction	Extracts text, tables, and forms from scanned documents, PDFs, images. (Amazon Web Services, Inc.)
Amazon Kendra	Search / Knowledge Retrieval	Enterprise search powered by ML: you can ask questions in natural language and retrieve relevant content from many sources. (AWS Documentation)
Amazon Forecast	Forecasting / Time Series	Predicts future business metrics (demand, sales, capacity) using ML-driven forecasting models. (Medium)

Amazon Personalize	Recommendation / Personalization	Provides real-time personalized recommendations (product, content, etc.) using ML. (kravigupta.in)
Amazon Fraud Detector	Fraud / Anomaly Detection	Detects fraudulent activity using ML models trained on your data + Amazon's fraud expertise. (AWS Documentation)
Amazon Augmented AI (A2I)	Human-in-the-loop / Labeling	Lets you build workflows where human reviewers validate or correct ML predictions (for higher accuracy or trust). (AWS Documentation)

An AWS Comprehend Medical architecture diagram

<https://claude.ai/public/artifacts/59192e14-8274-4693-af61-e96f0b6eb474>

Tier 2: ML services

The ML services tier provides a **more customized** approach for customers who want a bit more **control** over their **ML solutions** without having to manage infrastructure. **SageMaker AI** is a key offering in this tier.

Amazon SageMaker AI

With this **fully managed service**, you can **build, train, and deploy** your own **ML models** without worrying about infrastructure. The SageMaker AI integrated development environment (**IDE**) provides simplified access control and transparency over your ML projects. You can **track model training experiments, visualize data, and debug and monitor** your workflows all within **one environment**. SageMaker AI even offers **access to hundreds of pre-trained models** that you can deploy in a few quick steps.



Choice of ML tools

Increase innovation with different tool choices. **Data scientists** can use the **IDE**, and **business analysts** can use the **no-code interface**.

Fully managed infrastructure

Focus on ML model development while SageMaker AI provides you with **high-performance, cost-effective infrastructure**.

Repeatable ML workflows

Automate and standardize your **MLOps** practices and governance across your enterprise to support **transparency** and **auditability**.

Tier 3: ML frameworks and infrastructure

Some organizations have **highly specialized** needs that require **complete control** over the **ML training process**. They can use in-house expertise, ML frameworks, and AWS infrastructure to develop their own ML solutions.

ML frameworks

An ML framework is a **software library** or tool that provides **experienced ML practitioners** with **pre-built, optimized components** for building **machine learning models**. AWS supports ML frameworks like **PyTorch**, **Apache MXNet**, and **TensorFlow**.

AWS ML infrastructure

AWS **ML infrastructure**, such as **ML-optimized** Amazon Elastic Compute Cloud (Amazon **EC2**) **instances**, **Amazon EMR**, and Amazon Elastic Container Service (Amazon **ECS**), can support these custom solutions. These services provide high performance and flexibility for **advanced ML workloads**.

Bedrock vs. SageMaker

AWS CCP AI

Aspect	Amazon Bedrock	Amazon SageMaker
Primary Goal	Generative AI and rapid application development using Foundation Models (FMs) .	End-to-End Machine Learning lifecycle (build, train, deploy) for all ML types.
Model Scope	Primarily Foundation Models (FMs) from leading providers (Anthropic, Meta, Cohere, Amazon Titan, etc.).	Any Model: Custom models, open-source models (like Llama, Mistral), and traditional ML models (XGBoost, TensorFlow, PyTorch).
Infrastructure	Serverless (API-Driven). You manage zero infrastructure. You call an API, and AWS handles the serving and scaling.	Serverful/Managed Instance-Based. You provision and manage compute instances (e.g., <code>ml.g5.xlarge</code>) for training and deployment endpoints.
Customization	High-Level. Focuses on Fine-Tuning select models and Retrieval-Augmented Generation (RAG) via Knowledge Bases.	Deep/Granular. Full control over data preprocessing, model architecture, training loops , hyperparameter tuning, and deployment infrastructure.
Target User	Application Developers, Product Managers, Teams	Data Scientists, ML Engineers, MLOps Teams.

	needing quick AI integration.	
Cost Model	Pay-per-use (per token/per image generated) or Provisioned Throughput for predictable high-volume usage.	Pay for Instance Hours (notebooks, training, endpoints). Cost is based on provisioned uptime, not just usage.
Speed to Market	Very Fast. Integrate a state-of-the-art model into an application in hours or days.	Slower. Requires setting up training jobs, endpoints, and managing the MLOps pipeline.
Control & Isolation	Low Control. AWS manages the underlying compute. Excellent security via VPC endpoint access.	High Control. You can deploy endpoints directly within your own VPC for maximum network isolation and security auditing.

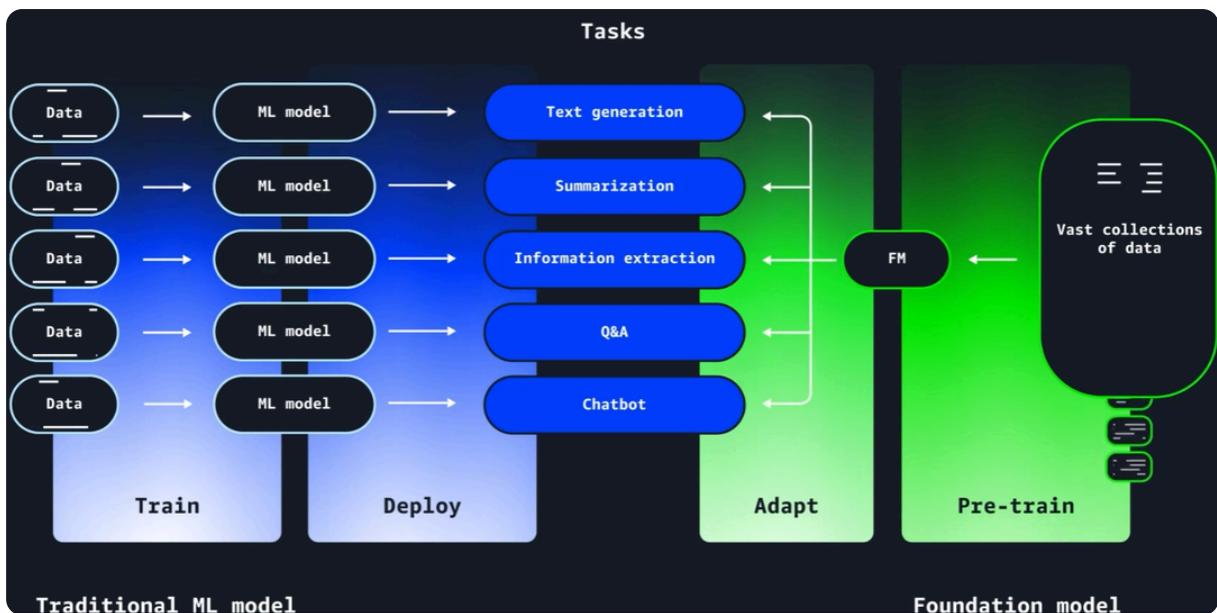
▼ Introduction to Generative AI

We've talked about AI, and we've talked about its subset, machine learning. Let's go even further and talk about deep learning. Deep learning is a subset of machine learning. The theory behind it has been around for decades, but we lacked the hardware to realize it until recently.

In deep learning, models are trained using **artificial neural networks** that mimic the human brain. These networks contain layers of artificial neurons, or **math functions**, that mimic real human neurons. Each of them **summarizes** and **feeds information** to the **next layer of artificial neurons** until a **final model is produced**. This advanced technique helps us tackle more complex problems than ever before, such as computer vision, and **natural language processing**.

All of this makes **generative AI** possible. You might have heard of it. **Generative AI is a type of deep learning** that produces **models** capable of **creating new content and ideas**, like conversations, stories, images, and music. Generative AI is powered by extremely **large machine learning models** that are **pre-trained on vast collections of data**. These are commonly called **foundation models**, or **FMs**.

Large language models, or **LLMs**, are a popular type of **foundation model** that are trained on vast amounts of text so they can learn how human language works. Unlike **traditional ML models**, which are trained to perform **singular tasks**, **pre-trained FMs** can be adapted to perform **multiple tasks**.



AWS provides tools and services to help you **build** and **scale** customized **generative AI solutions** tailored to your business.

The first of these services is Amazon **SageMaker JumpStart**. SageMaker JumpStart is a **machine learning hub** with **foundation models** and **prebuilt ML solutions** that you can deploy with a few clicks. These **pre-trained models** are **fully customizable** for your specific use case, by using **your own data**.

Amazon **Bedrock** is a **fully managed service** that offers a broad choice of high-performing, **pre-trained foundation models** from Amazon and **other leading AI companies**. With Amazon Bedrock, you can privately **adapt these models with your data** and deploy them without managing infrastructure. You can even use a common **API to access multiple FMs**.

Amazon **Q** is an **interactive assistant** that can be tailored to your business. It seamlessly **integrates with your company's information repositories** so it can engage in **contextualized conversations**, provide insightful **solutions**, and **complete actions** relevant to your organization. The **Amazon Q family** of products includes **Amazon Q Business** and **Amazon Q Developer**.

The capabilities of generative AI are exciting to contemplate. It's already helping us reimagine customer experiences, boost productivity, and drive innovation.

Deep learning

Deep learning (DL) is a subset of machine learning where models are trained using layers of artificial neurons that mimic the human brain. Each layer of these neural networks summarizes and feeds information to the next layer until a final model is produced.

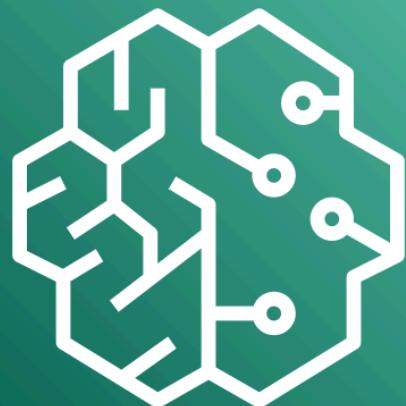
Generative AI

Generative AI is a type of deep learning powered by extremely large ML models known as **foundation models** (FMs). FMs are pre-trained on vast collections of data. While traditional ML models are trained to perform singular tasks, FMs can be adapted to perform multiple tasks.

Large language models (LLMs), are a popular type of FM trained to use human language. Foundation models can also be used to create videos, images, music, and more.

- **Amazon SageMaker JumpStart**

- An ML hub with FMs and pre-built ML solutions deployable with a few clicks



- **Amazon Bedrock**

- A fully managed service for adapting and deploying FMs from Amazon and other leading AI companies



- **Amazon Q**

- An interactive AI assistant that can be integrated with a company's information repositories



How to develop Gen AI with AWS

<https://claude.ai/public/artifacts/43e0d9fe-37a2-4dcc-9272-c67e5c3013dc>



genai_development_paths

Deep Learning & Generative AI Explained

<https://claude.ai/public/artifacts/8e2680de-3031-43be-931c-418c46a1f7a6>



deep_learning_hierarchy

Simple Example:

Scenario: You run an e-commerce company

Use SageMaker ML for:

- Predicting which customers will return a product (you train on YOUR past return data)
- Detecting fraudulent transactions based on YOUR transaction patterns
- Recommending products based on YOUR customer behavior

Use Bedrock FM for:

- Creating a chatbot that answers customer questions in natural language
- Generating product descriptions automatically
- Summarizing customer reviews
- Translating product pages to different languages

Key Differences: SageMaker ML vs Bedrock FM

Aspect	SageMaker MLs	Bedrock FMs
Training	YOU train it from scratch	Already trained (by Amazon/others)
Model Size	Small to medium (MB-GB)	HUGE (100+ GB)
Data Needed	Your specific labeled dataset	Already trained on billions of data points
Tasks	ONE specific task	MANY different tasks
Time to Deploy	Days/weeks (training time)	Minutes (ready to use)
Expertise Needed	ML engineering skills	API integration skills
Cost Model	Pay for training compute	Pay per API call/token

▼ Amazon Generative AI Solution

- Amazon SageMaker JumpStart for accelerating model development and deployment.
- Amazon Bedrock for deploying high-performing FMs through a single API.
- Amazon Q integrates with your existing information repositories to answer questions and helps generate insights and new content.

Amazon SageMaker JumpStart

SageMaker JumpStart is a machine learning hub within SageMaker AI that **accelerates** the process of **building, training, and deploying ML models**. SageMaker JumpStart offers a **library** of **pre-built ML** solutions across various domains such as computer vision, NLP, and tabular data. These pre-trained models can be fine-tuned to suit your specific needs and deployed with just a few clicks.

- **Rapid ML model deployments**

Quickly deploy pre-trained models without extensive ML expertise.

- **Custom fine-tuned solutions**

Fine-tune pre-trained FMs with your domain-specific data.

- **ML experiments and prototypes**

Compare performance for different models before committing to a specific approach.

Amazon Bedrock

Amazon Bedrock is a **fully managed** service that was specifically designed for working with large foundation models and building generative AI applications. It provides access to FMs from Amazon and leading AI startups, such as **Claude** and **Stable Diffusion**, all through a **single unified API**. With Amazon Bedrock, you can quickly experiment with FMs, fine-tune them with your own data, and seamlessly integrate them into your AWS applications.

Uses :

Enterprise-grade generative AI

Build **production-ready** generative AI applications with enterprise-level security, privacy, and scalability.

Multimodal content generation

Create **applications** that can **generate multiple content types**, such as text and images.

Advanced conversational AI

Develop advanced conversational **agents** that connect to **your enterprise data** to provide **accurate responses**.

Amazon Q products

Amazon Q is a generative AI assistant that can help companies streamline processes, get to **decisions faster**, and improve **employee productivity**. It can help every employee gain insights into their data and accelerate their tasks.

Amazon Q Business

Amazon Q Business can **answer pressing questions**, help **solve problems**, and **take actions using the data** and expertise found in **your company's information repositories**. Amazon Q Business provides this tailored assistance with a secure connection to commonly used systems.

Use cases: Information requests, automated workflows, and insight extraction

Amazon Q Developer

Amazon Q Developer provides **code recommendations** to accelerate development for coding languages including C#, Java, JavaScript, Python, and TypeScript applications. It **integrates** with multiple **IDEs** and helps developers write code faster by **generating** entire functions and logical blocks of **code**.

Use cases: Faster code generation, improved reliability and security, and automated code reviews

▼ Introduction to Data Analytics

Let's talk about data. It's what makes advanced technologies like AI and ML possible. In turn, the predictive capabilities of AI/ML are revolutionizing the way we analyze data.

Both AI/ML and traditional data analytics need **clean and accessible data**. Data analytics is when analysts transform **raw historical data** to uncover valuable insights and **trends**. And, though there is a lot of focus on AI/ML in today's tech landscape, traditional data analytics are still incredibly relevant and necessary.

For example, loan companies rely on data analytics to explain lending decisions to customers. Medical researchers still need traditional methods like hypothesis testing to analyze clinical trial data. And insurance companies use analytics to make sure their risk assessment models can be clearly understood and approved by regulators. Additionally, with smaller datasets, data analytics methods can be more efficient and more cost-effective than AI/ML methods.

Regardless of the method, both AI/ML and traditional data analytics have one thing in common: they're hungry for data. **Lots and lots of data!** This data can come from any number of source systems. Every time you make a purchase, log into a website, or even just browse online, you're generating data. All of this data is scattered across different systems and formats. To have any hope of analyzing it or making use of it, it needs to be brought together in **one place**. This is where the concept of **data lakes** comes in. A data lake is like a giant reservoir where businesses can **store all of their data**.

But having all of your data in one place isn't enough. It also needs to be in a **format** that's usable by **analytics tools** and **AI algorithms**. This is where the **extract, transform, and load**, or **ETL processes** are used. With ETL, you **extract** the data from various source systems and **store** it. Then, you **transform** it into a consistent usable **format** for downstream tools to consume. And then, you **load** it into a **destination system**, like a data warehouse or **analytics platform**. Or sometimes, you might follow an **ELT** pattern instead, where you **extract** the **data** and **ingest** it, **load** the data into the **tools**, and then **transform it as needed**. The choice depends on your specific needs and infrastructure.

Sometimes ETL isn't even necessary. A **zero-ETL process** works fine when your **data** is already in a **usable format** and **location** to be consumed by target systems.

When ETL or ELT is needed, **data pipelines** are used to make the process efficient and **repeatable**. These pipelines serve as **assembly lines** that help **automate** the **flow of ingesting data, processing it, and making it consumable**.

AWS has a suite of services for every step of this process. That includes **data ingestion** services like **Amazon Kinesis** and **AWS Glue**, and **storage** solutions like **Amazon S3** or **Amazon Redshift**. For **processing data**, tools like **Amazon EMR** are available, and for **visualization**, you can use services like **Amazon QuickSight**.

AWS services integrate with each other in different ways. So, let's say you're using Amazon S3 as a data lake where you have ingested a massive amount of raw data. You can set up processes to make that same data set available for a wide variety of different business needs. For example, your marketing team can use QuickSight to analyze a dataset for business intelligence. And your data science team can use that exact same dataset in Amazon SageMaker AI to train ML models.

Data pipelines for ETL processes

1. Extract the data from various sources and store it.
2. Transform it into a consistent, usable format for downstream tools to consume.
3. Load it into a destination system, like a data warehouse or analytics platform.

▼ Data Pipelines on AWS

It's time to explore data pipelines on AWS. A data pipeline **automates** the process of **ingesting**, **cataloging**, **transforming**, and **delivering data from source to destination**. This streamlining helps reduce manual effort and minimize errors.

But to understand these concepts better, let's take a step back and review a generic data pipeline.

The **first part** of our pipeline is where we **ingest** and **store** our required **data**. Data can come from many different sources such as applications, databases, live sensors, and streams. To gain insights, we need to consolidate this data in a **single location** and transform it into a useful **format**. Two choices are **data lakes** and **data warehouses**. Data lakes can store vast amounts of **raw data** and are more **flexible**. Data warehouses, however, are more structured because they are **optimized for business intelligence**. Amazon **S3** is a popular choice for **data lakes**, and Amazon **Redshift** is commonly used for **data warehouses**.

Let's move to **ingestion**. This is the process of **moving data** from the **source** systems into our chosen **storage solution**. Some applications might require **real-time ingestion**, whilst others can **tolerate** some **latency**, which makes **batch ingestion** the more appropriate choice.

Amazon Kinesis Data Streams

You can use Kinesis Data Streams for real-time ingestion of terabytes of data from applications, streams, and **sensors**. This **serverless** service even provides **automatic provisioning** and **scaling in on-demand mode**.



Amazon Kinesis Data Streams is ideal for **real-time** data ingestion. This is vital for applications requiring **low-latency processing**. Even better, **multiple apps** can consume data from the **same stream simultaneously**. A financial services company might use Kinesis Data Streams to ingest real-time stock market data so they can analyze it for immediate trading decisions.

Amazon Firehose and Kinesis

CCP IT AWS Database

Related Project

 Cloud Certified Practitioner Essentials note

Related Ideas

 AWS Workflows

Feature	Kinesis Data Streams (KDS)	Data Firehose
Primary Goal	Custom Processing & Real-Time Analytics	Automated Delivery & Loading
Latency	Real-Time (~70 ms with enhanced fan-out)	Near Real-Time (Lowest buffer is \$\approx 60 \text{ seconds}\$)
Management & Scaling	Manual/Semi-Managed. Requires manual Shard Management (provisioning, scaling, and scaling logic).	Fully Managed & Serverless. Scales automatically to meet incoming data volume with zero administration.
Data Storage	Built-in Storage. Data persists for 24 hours (default), extendable up to 365 days.	No Storage. Data is immediately buffered and delivered; no built-in retention.
Data Consumers	Open-Ended. Supports multiple consumers reading the same data stream simultaneously (e.g., one consumer for Lambda, one for EMR).	Closed-Ended. Only delivers data to the pre-configured destination (one delivery stream = one main destination).
Data Transformation	Requires custom code in a consumer application (Lambda, KCL, Apache Flink).	Supports basic serverless transformations (format conversion to Parquet/ORC, compression, and optional Lambda functions).
Pricing		

	Charged per Shard Hour and per PUT Payload Unit (more complex).	Charged per GB of data ingested (simpler, pay-as-you-go).
--	-------------------------------------------------------------------------------	------------------------------------------------------------------

Use Case Examples

Kinesis Data Streams (KDS): The Custom Processor

Use KDS when you need maximum control, low latency, and the ability to process data with custom logic before it lands in storage.

- **Real-Time Dashboarding & Alerting:** Capturing financial transaction data or IoT sensor readings and immediately running custom code to identify anomalies or trigger an alert **within milliseconds**.
- **Complex Event Processing (CEP):** Consolidating clickstream data from a website, processing it with a custom application (using the Kinesis Client Library) to calculate complex user sessions, and then sending the derived data to multiple downstream systems concurrently.
- **Replay Capability:** Storing the raw stream data for up to a week to allow developers to re-run processing jobs or recover from application errors.

Data Firehose: The Data Loader

Use Data Firehose when your main goal is simplicity, serverless scaling, and reliability in dumping streaming data into a permanent data store with minimal processing.

- **Log Ingestion to S3 Data Lake:** Streaming large volumes of application logs, network logs (like AWS WAF), or clickstream data directly into **Amazon S3** for long-term storage and subsequent analysis with **Athena** or **Redshift Spectrum**.
- **BI & Search Analytics:** Loading clean, pre-processed data streams directly into an analytics store like **Amazon Redshift** or **Amazon OpenSearch Service** (Elasticsearch) for immediate business intelligence (BI) reports.
- **Automated Format Conversion:** Automatically converting incoming JSON logs into the more efficient **Apache Parquet** format before saving them to S3, without writing a single line of ETL code.

What Does Kinesis Data Streams Do?

The Missing Piece: DynamoDB Can't Talk Directly to Firehose!

Think of it like this:

- **DynamoDB** speaks "Database language"
- **Firehose** speaks "Streaming data language"
- **Kinesis Data Streams** = The translator between them!

The Real Problem:

DynamoDB Streams (DynamoDB's change log feature) captures every change:

- Customer added item to cart → DynamoDB records it
- Customer updated profile → DynamoDB records it
- But DynamoDB Streams **can't send directly to Firehose**

Kinesis Data Streams bridges this gap:

1. Reads changes from DynamoDB Streams
2. Converts them into streaming format
3. Feeds them to Firehose in the format it expects

Visual Analogy:

```
Text
DynamoDB → "I have changes!"
↓
Kinesis Streams → "Let me repackage those for you"
↓
Firehose → "Perfect! Now I can aggregate and deliver!"
```

What Does "Ingesting" Mean?

Ingesting = The process of bringing/collecting data into your system

Think of it like eating:

- **Ingesting food** = Putting food into your mouth and stomach
- **Ingesting data** = Pulling data from sources into your data pipeline

In This Pipeline:

"Data Ingestion" specifically means:

- **Capturing** data changes from DynamoDB
- **Streaming** them through Kinesis
- **Delivering** them to S3 via Firehose

It's the entry point of your data pipeline - like the mouth is the entry point for food!

Why This 3-Step Process?

Step	Purpose	Why Needed?
DynamoDB	Store operational data	Fast reads/writes for app
Kinesis Streams	Capture & stream changes	Bridge between DynamoDB and Firehose

Firehose	Aggregate & deliver	Batch data efficiently to S3
----------	---------------------	------------------------------

Without Kinesis: DynamoDB → X → Firehose (can't connect directly!)

With Kinesis: DynamoDB → ✓ Kinesis → ✓ Firehose (works perfectly!)

Now let me create a full architecture diagram for you: Perfect! I've created a complete architecture diagram with all components clearly labeled and explained. Here's the summary:

Key Answers to Your Questions:

Why Do We Need Kinesis?

Simple answer: DynamoDB and Firehose can't talk to each other directly! They speak different "languages."

The relay race analogy:

1. **DynamoDB** = Fast runner (handles app data)
2. **Kinesis Streams** = Middle runner (passes the baton)
3. **Firehose** = Final runner (delivers to S3)

You can't skip the middle runner! Without Kinesis, there's no connection.

What is "Ingesting" in This Context?

Ingesting = The process of pulling/collecting data into your pipeline

Think of it like a restaurant:

- **Ingestion** = Food arriving in the kitchen (data flowing in)
- **Processing** = Cooking the food (transforming it)
- **Serving** = Delivering to diners (storing in S3 for use)

In this pipeline:

- Kinesis Streams **ingests** changes from DynamoDB
 - Firehose **ingests** streaming data from Kinesis
 - Then delivers to S3
-

The Complete Flow:

Customer Action → DynamoDB → **Kinesis (Bridge)** → **Firehose (Aggregator)** → Lambda (Transformer) → S3 → Athena/SageMaker

Each component has a **specific job** that others can't do! ⌚

Amazon Data Firehose

Firehose is an option for data ingestion in near real-time. This fully managed service provides automatic provisioning and scaling. It also delivers data within seconds to data lakes, warehouses, and analytics services.



Another service to mention is **Amazon Data Firehose**, which is a **fully managed near-real-time streaming ETL solution**. Firehose **collects data from a source and delivers it to a destination**. For example, a smart home device manufacturer might use Firehose to collect device data from all its devices for long-term storage and analysis.

Once data is ingested and stored, the **next step** is to **catalog** that data using a **centralized data repository**. This provides an inventory of your organization's data. It's like when you take a selfie on your smartphone. It contains metadata, like the time you took the picture, where you took it, and its format. Is it a GIF? GIF me a break!

AWS Glue is a managed service with a feature called the **Data Catalog**. You can use **this centralized repository** to **store metadata** about your organization's data sets.

Next, we need to **clean** and **transform** our data so it's ready to be **analyzed**. For this, you can use **AWS Glue**. It's a fully managed service that offers **visual ETL job creation**, **built-in job scheduling**, and support for **various data sources** and **formats**. It's ideal for organizations looking for a **simplified** approach, and it offers **code-free script creation** as well.

AWS Glue

AWS Glue is a **fully managed ETL service** that makes data preparation simpler, faster, and cost effective. AWS Glue ETL jobs can use the AWS Glue Data Catalog to access metadata about data sources, which can help inform transformations defined in the ETL script.



AWS Glue Data Catalog - (also explained in Amazon Athena)

AWS Glue Data Catalog provides a centralized, scalable, and managed metadata repository that enhances data discovery. It improves the overall efficiency of data pipelines by delivering metadata to various data stores and analytics services.



For more **complex data processing**, there's **Amazon EMR**. It's ideal for **large-scale data processing** using popular **frameworks** like **Apache Spark**, Apache Hadoop, and Apache Hive. Amazon EMR is best suited for organizations with existing big data expertise and those requiring **customized configurations**.

Amazon EMR

Amazon EMR is ideal for large-scale data processing and organizations with existing big data expertise. It **automatically handles infrastructure provisioning, cluster management, and scaling**. Amazon EMR supports popular big data frameworks like Apache Spark, Apache Hadoop, and Apache Hive.



OK folks, so we've ingested, stored, cataloged, and processed our data. It's finally ready to be used. Actually, I can see a bunch of consumers lined up already to analyze the data. But they're looking for the right **query solution**. Well, good news for them! They have lots of options, including **Amazon Athena** and **Amazon Redshift**, among others.

Amazon Athena

With Athena, you can run **SQL queries** to analyze data in relational, nonrelational, object, and custom data sources. This **fully managed** serverless service can access data hosted on Amazon **S3**, **on premises**, or even in **multi-cloud environments**. It offers a **cost-effective** solution for data analysis because you only **pay** for the **queries** you run.



Amazon Athena

Let me break this down in simple terms!

🔍 What Athena Does (The Magic Part)

Athena lets you run **SQL queries** directly on files in **S3** - **WITHOUT moving the data anywhere!**

Normal Database Way (Traditional):

1. Load data from S3 into a database ⏳ (takes time)
2. Set up database servers 💰 (costs money)
3. Query the database
4. Pay for storage in TWO places (S3 + database) 💰

Athena Way (Smart):

1. Data stays in S3 ✅
2. No servers needed ✅
3. Just write SQL and query directly ✅
4. Pay only when you query ✅

It's like reading a book in a library vs checking it out - the book never moves, you just read it where it is!

Why Athena Needs AWS Glue Data Catalog

The Problem Without Glue:

Imagine you have a CSV file in S3 with customer data. Athena looks at it and thinks:

Text

```
customer123, John, Smith, 42, New  
York  
customer456, Jane, Doe, 35, Boston
```

Athena's confusion: 🤔

- "What do these columns mean?"
- "Is the first column a customer ID or a name?"
- "Is '42' an age or a zip code?"
- "What data type is each field?"

Without knowing the schema, Athena can't properly query the data!

The Solution: Glue Data Catalog

Glue Data Catalog = A metadata dictionary that tells Athena what everything means

It stores information like:

- **Column names:** customer_id, first_name, last_name, age, city
- **Data types:** string, string, string, integer, string
- **File location:** s3://my-bucket/customer-data/
- **File format:** CSV, JSON, Parquet, etc.

Think of it like a menu at a restaurant:

- Without a menu (no Glue Catalog):
"What is this food? What's in it?"

 - With a menu (Glue Catalog): "Oh,
this is a Caesar Salad with romaine
lettuce, croutons, parmesan..." 
-

How They Work Together

Step 1: AWS Glue scans your S3 files
and creates a "table definition":

 Text

```
Table: customer_data
Columns:
- customer_id (string)
- first_name (string)
- last_name (string)
- age (integer)
- city (string)
Location: s3://my-
bucket/customer-data/
```

Step 2: Data scientist opens Athena
and writes SQL:

 SQL

```
SELECT city, AVG(age)
FROM customer_data
GROUP BY city;
```

Step 3: Athena checks Glue Data
Catalog:

- "Oh! customer_data is in this S3
location"
- "It has these columns with these
types"
- "The files are in CSV format"

Step 4: Athena runs the query directly
on S3 files and returns results!

Why This is "Automatically Recognized"

Without Glue Catalog:

```
⌚ SQL  
-- You'd have to manually tell  
Athena everything:  
CREATE EXTERNAL TABLE  
customer_data (  
    customer_id STRING,  
    first_name STRING,  
    last_name STRING,  
    age INT,  
    city STRING  
)  
LOCATION 's3://my-  
bucket/customer-data/';
```

Every. Single. Time. For. Every. Table.

With Glue Catalog:

```
⌚ SQL  
-- Just query! Athena already  
knows the structure:  
SELECT * FROM customer_data  
WHERE age > 30;
```

⌚ Real-World Analogy

Imagine you're a librarian (Athena):

Without Glue Catalog:

- Someone asks: "Show me all science books published after 2020"
- You: "Um... what's a science book? Which shelf? What does 'published' mean?"
- You have to manually check every book 📚🤔

With Glue Catalog:

- You have a computer system that lists:
 - Book titles
 - Categories (science, fiction, etc.)
 - Publication dates
 - Shelf locations

- You: "Ah yes, science books after 2020 are on Shelf 5, rows 3-7" 📚



Summary

Athena: "I can read data in S3 without moving it!"

Glue Data Catalog: "I'll tell you exactly what that data looks like so you can read it correctly!"

Together: Data scientists can easily query massive amounts of data in S3 using simple SQL, without:

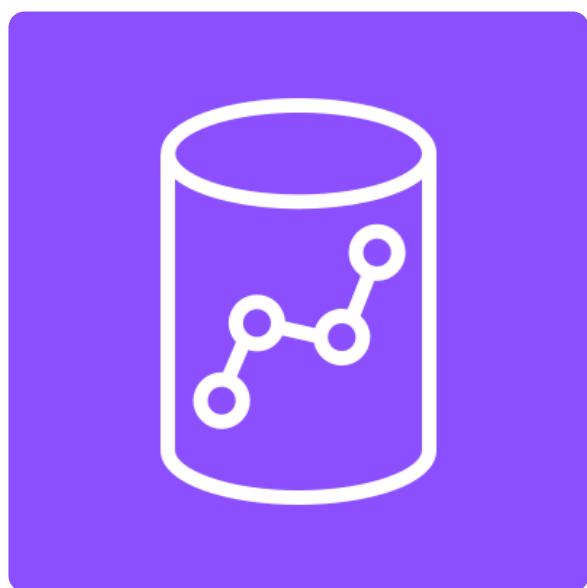
- Moving data around
- Setting up databases
- Managing servers
- Manually defining schemas every time

It's serverless, automatic, and super fast! That's why Rudy says "Athena is so cool!" 😎

Athena is a **fully managed serverless service**. You can submit a **SQL query** to analyze data in relational, nonrelational, object, and custom data sources running on Amazon **S3** or even **hybrid environments**. Oh yeah, it can analyze data outside of AWS, too.

Amazon Redshift

Amazon Redshift is a fully managed data warehouse solution. Its columnar storage and massively **parallel processing architecture** make it ideal for analyzing **large datasets**. You can use it to perform complex SQL queries on large datasets for frequent, high-performance analytical workloads.



If you prefer a **fully managed data warehouse solution**, then **Amazon Redshift** is the ideal choice. It's better suited for complex queries on large datasets and **frequent, high-performance analytical workloads**. Amazon Redshift can store **petabytes** of **structured or semistructured data**. With the **scalability** and **pay-as-you-go** pricing model, organizations can **cost-effectively** analyze large datasets.

Amazon Athena vs Amazon Redshift - in a ETL process

IT CCP AWS Database

Related Project



Cloud Certified Practitioner Essentials note

Related Ideas



AWS Workflows

The core difference between **Amazon Athena** and **Amazon Redshift** lies in their architecture and primary purpose:

- **Amazon Redshift** is a **fully managed cloud data warehouse** (a place to store and analyze structured, cleaned data).
 - **Amazon Athena** is a **serverless query service** that lets you analyze data **directly** in Amazon S3 (your data lake) without needing to load it anywhere first.
-



Comparison of Athena and Redshift

Feature	Amazon Redshift (Data Warehouse)	Amazon Athena (Query Service)
Architecture	Cluster-based (or Serverless Endpoint): You provision and manage a cluster of compute nodes, or use a Redshift Serverless endpoint.	Serverless: No infrastructure to manage. Queries run directly on S3.
Data Storage	Internal: Data must be loaded into Redshift's highly optimized, columnar storage.	External (S3): Data stays in your Amazon S3 data lake . Athena just points to it.
Primary Use	Complex, high-performance analytics and Business Intelligence (BI) on massive, structured datasets.	Ad-hoc, interactive queries and exploratory analysis on raw or semi-structured data.
Performance	Excellent for complex queries (joins, aggregations) on structured data, especially with high-concurrency needs.	Fast for quick scans and exploratory work; performance can slow down on extremely complex joins or full table scans of huge, unoptimized datasets.

Cost Model	Pay for Compute Time and Storage: You pay for the cluster size/uptime (fixed cost) or the compute capacity (serverless).	Pay per Query/Data Scanned: You only pay for the amount of data the query actually scans.
-------------------	---------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------

Use Cases in an ETL Pipeline

In a modern data architecture, the raw data often lands in S3 (the **Data Lake**), and then an ETL (Extract, Transform, Load) process prepares it for a final destination (the **Data Warehouse**).

Amazon Athena Use Case (Exploration/Initial Analysis)

Role in ETL Pipeline	Example Use Case
Data Audit & Validation (T-Step)	A new data source (e.g., application log files) is dumped into S3. Before running a costly, scheduled AWS Glue job to clean it, a data engineer uses Athena to run quick SQL queries on the raw files. This is to validate the data quality, check the schema, and ensure no bad records exist.
Ad-Hoc Reporting (End-User)	A security analyst needs to troubleshoot a specific incident. They bypass the main data warehouse and use Athena to directly query years of raw VPC Flow Logs stored in S3 to look for a specific IP address pattern on a single day.

Amazon Redshift Use Case (Final Destination/BI Reporting)

Role in ETL Pipeline	Example Use Case
Target Destination (L-Step)	An AWS Glue job (the 'T' step) cleans, aggregates, and joins all monthly sales data. It then loads the final, highly structured, and optimized data into Redshift . This is the single source of truth for all financial reporting and long-term storage.
Complex BI & Dashboarding	A Business Intelligence tool (like QuickSight or Tableau) connects to Redshift to run 50 complex analytical queries simultaneously, generating daily executive dashboards. Redshift's architecture is optimized to handle this high-concurrency, repeated workload with consistent, low latency.

Using Them as Complementary Services

Yes, Redshift and Athena are often used together in a modern "Lake House" architecture.

This approach combines the low-cost storage and flexibility of a data lake (S3/Athena) with the high-performance analytics power of a data warehouse (Redshift).

Complementary Example: Redshift Spectrum

The primary way they complement each other is through a feature called **Redshift Spectrum**.

1. **Redshift Spectrum** allows you to run SQL queries from your **Redshift cluster** directly against data stored in **Amazon S3 (the data lake)**.
2. Redshift acts as the **primary analytical tool**, querying data stored both internally (the "hot," frequently accessed, structured data) and externally in S3 (the "cold," historical, or less-structured data).

Scenario Example:

- **Redshift:** Stores the last **6 months** of transactional data (the "hot" data) for fastest, most complex queries for daily reporting.
- **S3/Athena:** Stores **5 years** of historical log and sales data (the "cold" data) in cost-effective formats like Parquet.
- **The Query:** An analyst runs a single query in Redshift to calculate "Total Revenue for the last 5 years." Redshift processes the recent 6 months internally, and then uses **Redshift Spectrum** to seamlessly include the 5 years of historical data from S3, giving a complete result.

This approach lets you minimize your Redshift cluster size (saving cost) while still having access to all your data.

You can learn more about how these services work together in a data pipeline in this video: [Using Amazon Redshift Spectrum, Amazon Athena, and AWS Glue with Node.js in Production](#).

OK, after they have queried their data, analysts usually need to **visualize** everything. For this, analysts can use **Amazon QuickSight** and **Amazon OpenSearch Service**.

Amazon QuickSight

With QuickSight, both technical and **non-technical users** can quickly create modern interactive **dashboards** and reports from various data sources without managing infrastructure. Amazon Q in QuickSight provides natural language queries so business analysts and users can build, discover, and share meaningful insights in seconds.



QuickSight is ideal for **unified business intelligence**, or **BI**, at **scale**. Both technical and **non-technical users** can **quickly create interactive dashboards and reports**. This means they can all meet their specific analytical needs without waiting on development teams. Also, with **Amazon Q in QuickSight**, they can use **natural language to build, discover, and share meaningful insights**.

Amazon OpenSearch Service

With OpenSearch Service, you can **search** for **relevant content** through **precise keyword matching** or **natural language queries**. **Unified dashboards** provide **real-time data visualization** as you **analyze** and **monitor logs, traces, and metrics** for various applications.



Let's flip to **OpenSearch Service**. It can be used for **real-time search, monitoring, and analysis of business and operational data**. This is especially helpful for use cases like **application monitoring, log analytics, observability, and website search**.

QuickSight vs OpenSearch

Feature / Purpose	Amazon QuickSight	Amazon OpenSearch Service
Primary Use	BI dashboards, reports	Full-text search, log analytics, app search
Users	Analysts, managers, business teams	Developers, DevOps, SRE, security teams
Natural Language Querying	Yes (QuickSight Q)	No (not in a BI sense)
Query Style	Ask a question in English	Query DSL (JSON), keyword searches
Data Type	BI datasets (tables, CSV, RDS, Redshift, S3)	Logs, documents, JSON, unstructured text
Visuals	Dashboards, charts, ML insights	Dashboards (Kibana/OpenSearch Dashboards)
Typical Use Cases	Sales dashboards, KPIs, forecasting	Log monitoring, search bars, error analysis

ML Features	Anomaly detection, forecasting	Anomaly detection for logs, relevancy tuning
Pricing	Per-user + SPICE storage	Cluster-based (nodes, storage, OpenSearch versions)

And with that, we've come to the end of this video. Remember that this is just the tip of the data pipeline iceberg.

▼ Data Analytics and AI/ML

Cloud in Real Life: Data Analytics and AI/ML:

Morgan: OK, wow! We covered a lot of great information in this module.

Rudy: Oh fully! And I think it's only fitting that we dive into another diagram. Alan, time to break out that ecommerce app you have lying around.

Alan: That is a great idea! Here, we can see this company has trained a machine learning model to make **recommendations** to customers using its **online app**. The company needs to keep this **model up to date** with the latest data. Data scientists also need to **query** that same data for **important insights**.

Morgan: I see where you're going with this, Alan. It's a perfect use case for an **automated data pipeline** because we need to **continuously collect** and **analyze customer data**. So, this company is **storing historical** customer data gathered from the app in an Amazon **DynamoDB** table.

Alan: Yeah, exactly. So, DynamoDB is a good fit for **low-latency reads and writes**, but it's **not practical** to **scan all the data** in a **DynamoDB** table to **train a model**. We really need to bring this data into **storage**, like a **data lake**, so that it can be available to the **machine learning process** used for personalized **recommendations**. And so, the data needs to go on a little journey.

Rudy: Oh, a journey! I love this part—a data pipeline journey! Learners, an **automated data pipeline** can help deliver data using an **efficient, repeatable, and error-free process**.

Morgan: So, the **first** step is to **ingest** the data. Amazon **Data Firehose** makes a lot of sense here. It provides **near real-time data ingestion**(like 5 minutes intervals). Plus, it requires **minimal setup** and gives you **automatic scaling**. The catch here is that there currently isn't a direct integration between DynamoDB and Firehose. We actually need to send the data changes from DynamoDB through **Amazon Kinesis Data Streams** first.

E-commerce ML Data Pipeline - Firehose role - Amazon Firehose and Kinesis

<https://claude.ai/public/artifacts/695d8737-dc2f-4221-a321-9ff5ae89929c>

Alan: So, DynamoDB sends the data to Kinesis Data Streams, then **Firehose aggregates** that **data** and delivers it to **S3 in near real time**. That's awesome! But the data coming from DynamoDB will be in **JSON** format and a little birdie tells me the **ML engineer and data scientist** need the data in **comma-separated values**, or **.csv**, format. We're going to need to transform the data.

Rudy: Ah, yes. Luckily, **Firehose** has a feature where it can **invoke** an AWS **Lambda function**. That function can **transform** incoming source **data** and then deliver the resulting data to a destination. That means after it's processed and properly **formatted**, say as a .csv file, the data is **available** for delivery to **multiple consumers** in **S3**.

Morgan: Right. And, in this case, the data is delivered to S3, which is the **data lake** for this company. This is where the company stores all of its **data** used for **ML and analytics**. The data scientists will want to run **ad-hoc queries** against the data, but we need a way to make this **data discoverable** so it's **accessible** for **querying**. To achieve this, we'll use the **AWS Glue Data Catalog** as the **metadata repository** and set up a **table** for the **customer data**. The Data Catalog will make it possible for us to define tables that describe the **schema** and **location** of our **data** stored in **S3**.

Rudy: Schweet, bru. That's amazing. And, this is perfect for **Amazon Athena**. Data scientists can simply run standard **SQL queries** against the **data** in **S3**. The **data doesn't** even need to be **moved out of S3** itself. Better yet, Athena can use the **table stored** in the **Glue Data Catalog**. This helps to **automatically recognize** the **schema** and **structure** of the **data**. Athena is so cool. I mean, seriously. It's so easy to set up and use.



Amazon Athena

Morgan: Yeah, I agree, I really love using Athena. Also, for our **pipeline**, **Amazon SageMaker AI** can **read** the latest information **directly from S3** to **train new versions** of the model.

Alan: That's right. And the beauty of the **data pipeline** is that after it's set up, it can **repeat automatically** based on a set **schedule**. The **data scientists** can focus on **gathering insights**, and the **ML engineers** can focus on keeping the **model up to date**.

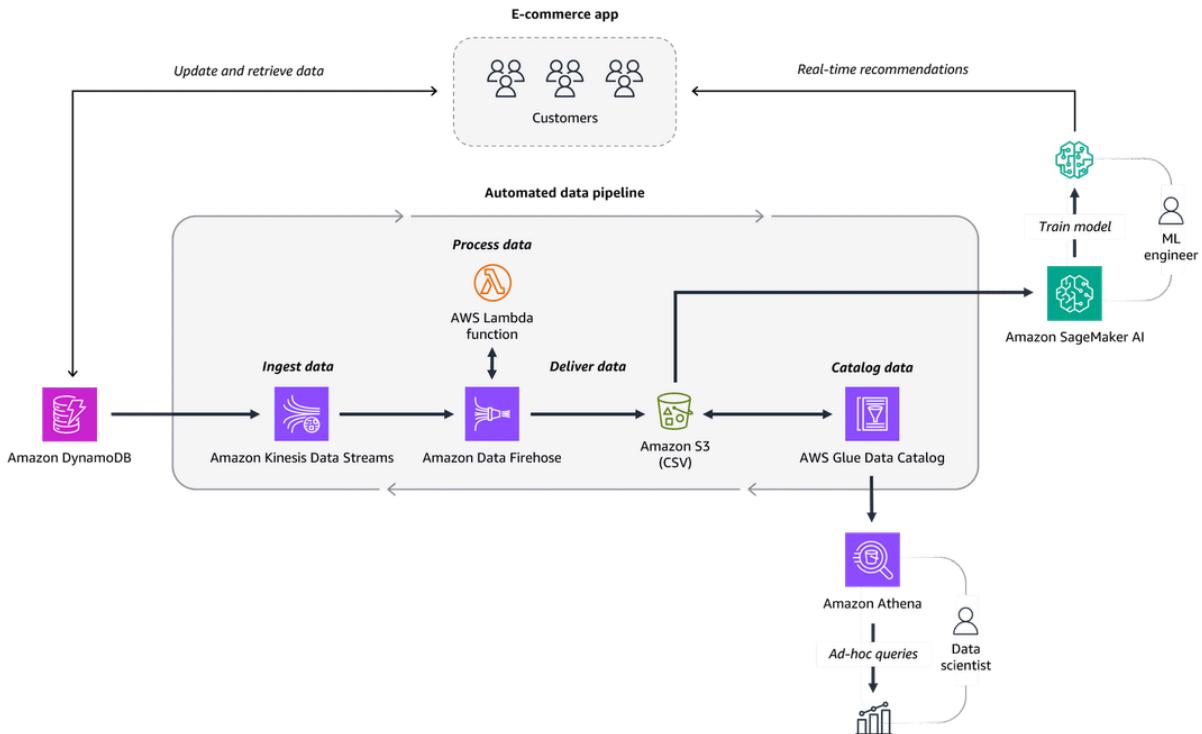
Morgan: OK, well, that's great! This is a real time saver. It's super common that the **same data** needs to be **used** for **more than one purpose**. So, this is a really good example of that.

Rudy: Oh, and this is just a basic pipeline. Imagine manually performing all these steps for a more complex one?

Morgan: Yeah, for real. As usual, **automation** is key for **efficiency** and innovation.

Complete E-commerce ML Data Pipeline Architecture

<https://claude.ai/public/artifacts/59a8c3c1-2aa1-40cc-b5df-06709ab19357>



1. Make recommendations

An e-commerce company uses an ML model to make product recommendations.

2. Store app data

An Amazon DynamoDB database stores the historical customer data gathered through the app. This makes sense for low-latency reads and writes but isn't ideal for ML model training.

3. Ingest data

Kinesis Data Streams ingests the data from DynamoDB. Amazon Data Firehose then aggregates the data.

4. Process data

The data is in JSON format, so Firehose invokes an AWS Lambda function that transforms the data into .csv format.

5. Deliver data

Firehose then delivers the data to the company's Amazon S3 data lake, where it is available for multiple consumers.

6. Catalog data

AWS Glue Data Catalog serves as a metadata repository with tables that describe the schema and location of the Amazon S3 data.

7. Perform data analytics

Data scientists use Athena to gather insights through queries.

8. Train model

SageMaker AI reads the same dataset directly from Amazon S3. ML engineers can then train new versions of the recommendation model using the latest information.

AWS Data Pipeline Services

CCP AWS

Service	What It Is	Purpose	Core Features	Most Important Use Case	Role in Data Pipeline
Kinesis Data Streams (KDS)	Real-time streaming ingestion service	Capture, buffer, and process streaming data in milliseconds	Sharded streams, low latency, real-time processing, consumer apps (Lambda, Kinesis Analytics)	Real-time data ingestion (IoT, logs, clickstreams)	Ingest → Stream raw real-time data
Kinesis Data Firehose	Fully managed delivery/ingestion pipeline	Automatically load streaming data into destinations	No coding, auto-scaling, format conversion (JSON → Parquet), compression	Send streaming data directly to S3/Redshift/OpenSearch without managing servers	Ingest → Transform → Load (streaming ETL to storage or analytics systems)
AWS Glue	ETL & data integration service	Prepare, clean, transform, and move data	Serverless ETL jobs, Spark-based jobs, DataBrew (no-code), workflows	Data transformation and cleaning for analytics	Transform → Prepare data for databases or data warehouses
Glue Data Catalog	Central metadata store	Store table metadata + schemas	Schema registry, crawlers, integration with Athena/Redshift/EMR	Keep track of datasets across S3 and databases	Metadata layer powering query engines like Athena & Redshift Spectrum
EMR (Elastic MapReduce)	Big data processing cluster	Large-scale processing using Hadoop/Spark/Hive	Fully managed Spark clusters, scalable compute, customizable runtime	Petabyte-scale processing (ML training, ETL, batch processing)	Heavy Transform & Compute for big data workflows
Athena	Serverless SQL query engine	Run SQL queries directly on S3	No infrastructure, pay-per-query, integrates with Glue Catalog	Interactive queries on S3 data (CSV/JSON/Parquet)	Query Layer for data stored in S3
Redshift	Cloud data warehouse	High-performance analytics for structured data	MPP architecture, Redshift Spectrum, columnar storage	BI analytics at scale, dashboards, complex joins	Warehouse/Analytics Layer (store structured, query-ready data)

QuickSight	Business Intelligence (BI) dashboarding tool	Visualize data + share insights	Dashboards, ML insights, natural language querying (Q)	Executive dashboards, KPIs, business reporting	Visualization Layer consuming Redshift/Athena datasets
OpenSearch	Search engine + log analytics	Index and search text/log data	Full-text search, log ingestion, dashboards, real-time analysis	Application search, log analytics (ELK alternative)	Search/Observability Layer — analyze logs & semi-structured data

One-Sentence “If I’m a PM, What Should I Remember?”

- **Kinesis Streams** → Real-time data in.
- **Kinesis Firehose** → Stream → Destination automatically.
- **Glue** → Clean/transform data.
- **Glue Catalog** → Metadata so other tools know the schema.
- **EMR** → Heavy big data processing.
- **Athena** → SQL directly on S3.
- **Redshift** → Data warehouse for analytics.
- **QuickSight** → BI dashboards.
- **OpenSearch** → Fast search and log analytics.

 Complete AI-Powered E-commerce Platform

 AWS Data Pipeline(s)

Module 9 - Security

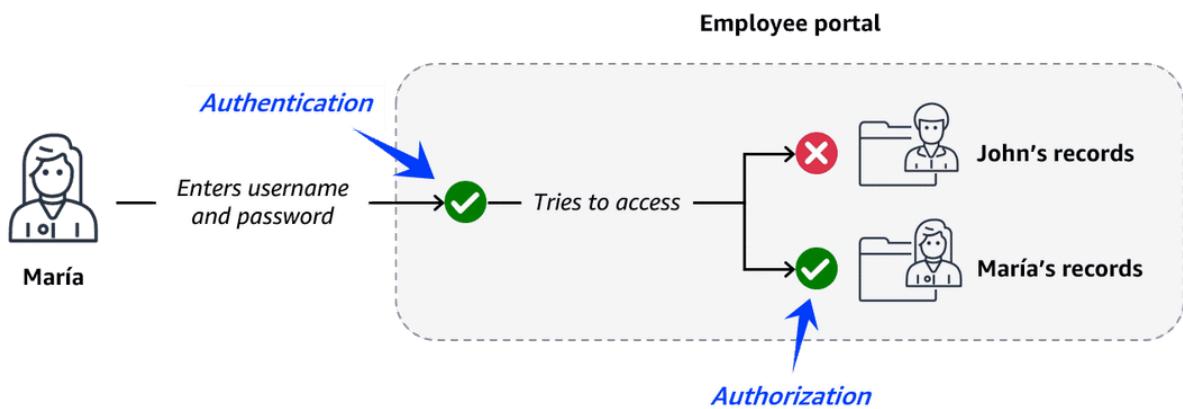
▼ Introduction to Security

In fact, every one of us has a role to play in security. To help you on your way of becoming a security guru, two crucial security components you should know about are authentication and authorization.

Authentication is the process of **verifying** the **identity** of a **user** or entity through **credentials** like a **username** and **password** combination.

Authorization, on the other hand, determines which **actions** users **are permitted to perform** in a system or application. This is usually done by **granting** a user **certain access rights** and **permissions**.

For instance, **authentication** permits me to **log in** to our employee portal. But **authorization** limits me to just the **areas** that I'm **allowed to access**, such as my own employee records.



Authentication and authorization play a vital role in data privacy and protection. Organizations must, therefore, safeguard all their users' personal information. This protection helps to maintain customer trust and prevent identity theft and financial fraud.

In essence, **organizations must prevent unauthorized access and misuse** as much as possible.

In this spirit, AWS offers multiple security mechanisms like the AWS Shared Responsibility Model - SRM, and a few other examples like:

- Preventing security incidents through proper permission and **access management**
- Proactively addressing security issues through **network, application, and data protection**
- And quickly **detecting** and **responding** to security incidents as they occur

And with that, let's get things started and take a look at the different security services, mechanisms, and features that AWS has to offer.

▼ Preventing Unauthorized Access

When you create an AWS account, you are given what is called the **AWS account root user**. This root user is the **owner** of the account and has permission to do anything they want inside of that account.

This is like being the owner of the coffee shop.

In this situation, let's say I am the owner of the coffee shop. I can come into the shop, use my credentials to work the register, work the inventory system, or any other system in the coffee shop. I cannot be restricted.

With the AWS account **root user**, you can access and control any resource in the account. You can spin up databases, EC2 instances, machine learning services, or literally whatever you want. Because that user is so powerful, we recommend that when you create an AWS account, you associate a strong password with your root user account. Then, as soon as you log in with your root user, you turn on **multi-factor authentication**, or **MFA**. This ensures that you need not only the email and password, but also a randomized token to log in.

Now, that's great. But even with MFA turned on, for security reasons, you really **don't** want to use the **root user** for **daily tasks**. So, with that being said, you can control access in a granular way on AWS by using the service, **AWS Identity and Access Management**, or **IAM**.

In IAM, you can create **IAM users** to represent **individual identities**. When you create an IAM user, by **default**, they have absolutely **zero permissions**. They can't launch an EC2 instance. They can't create an S3 bucket. Nothing. You have to **explicitly grant** the **user permission** to do anything in the account. So, remember, by default, all actions are denied. You have to explicitly allow any action done by any user. You give people **access only to what they need** and nothing else.

This idea is called the **least privilege principle**. And the way that you grant or deny permission is to associate what is called an **IAM policy** to an IAM user. An IAM policy is a **JSON document** that **describes** what **API calls a user can or cannot make**. Let's take a look at this quick example.

In this example, you can see we have a **permission statement**. This permission statement has the **effect** defined as **Allow**, the **action** as **s3>ListBucket**, and the **resource** is a **unique ID(arn)** for the **S3 bucket**.

```
{} JSON

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3>ListBucket",
      "Resource": "arn: aws : s3 : : : coffee_shop_reports"
    }
  ]
}
```

So, if I attach this policy to a **user**, that user could **view** the **bucket** `coffee_shop_reports`, but perform **no other action** in this account. To break this down further, it's good to note that there are only **two potential options** for the **effect** on any policy: either **allow** or **deny**.

For **action**, you can **list any AWS API call**, and for **resource**, you would list what AWS resource that **specific API call** is for.

Now, as a businessperson or **non-technical person**, you likely would not need to write these types of policies yourself. But, they are used all over in AWS accounts. One way to make it more convenient to manage your users and their permissions is to organize them into **IAM groups**. Groups are, well, they are groupings of users. You can **attach a policy** to a **group** and all of the **users** in that group will **inherit those permissions**.

Alright, so far with IAM, you have the root user—they can do anything. You have users that can be organized into groups. And you have **policies**, which are documents that describe permissions that you can then attach to users or groups. There is one other major identity in IAM, and that's called a **role**.

This idea of **temporary access** is important to understand when thinking about how **IAM roles** work in AWS. Roles, similar to users and groups, can have **associated permissions** that allow or deny specific actions. And these roles can be **assumed** for **temporary** amounts of time. It's similar to a user, but has **no static credentials** like a **username** and **password**. Instead, it's an identity that can be **assumed to gain access to temporary permissions**. You use roles to temporarily grant access to AWS resources, to users, external identities, applications, and even other AWS services.

Roles are particularly helpful when trying to manage **permissions at scale** in an organization. By using roles, you can actually avoid creating IAM users for every person in your business that needs access to your AWS account. You can accomplish this by federating users into your account. This means that they could use their regular corporate credentials to log into AWS by **mapping their corporate identities to IAM roles**.

IAM Identity Center is a service that can help you set up and manage this process. With this service, you can set up single sign-on so that your users get a streamlined login experience when accessing AWS account resources.

With IAM, we have authentication and authorization in place. So, I know my employees will only have access to what they need, which prevents some security incidents before they can even start.

AWS Identity and Access Management (IAM)

Securely manage identities and access to AWS services and resources.

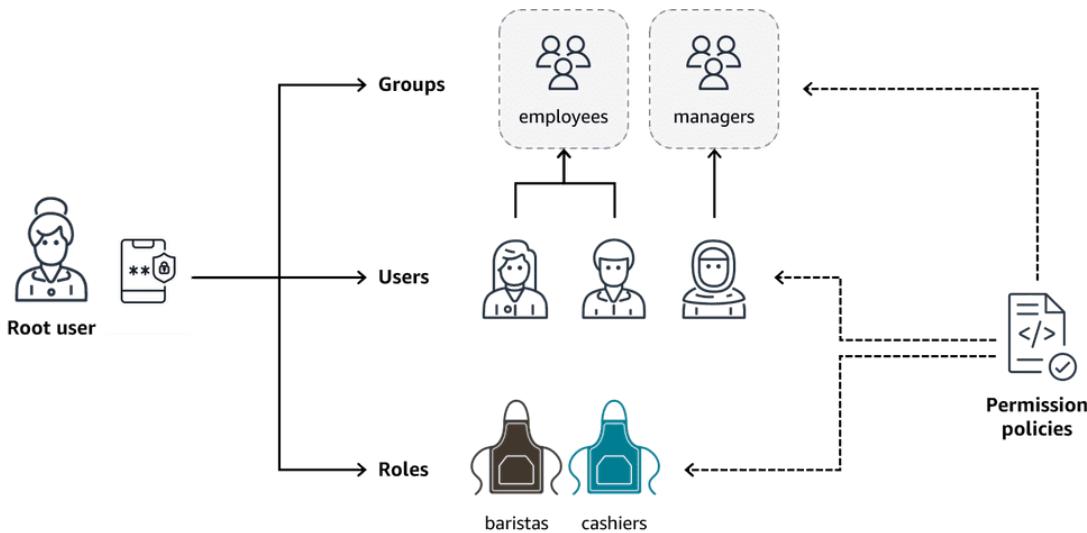
One of the best ways to prevent security incidents before they happen is through proper permission and access management. With IAM, by default, all actions are denied. You must explicitly grant permission to someone before they can perform any actions in your account.



When you grant permissions, you should provide access only on a need-to-have basis. This concept is called the principle of least privilege.

*The **principle of least privilege** dictates that you should only give people and systems access to what they need and nothing else.*

IAM provides users, groups, and roles so you can configure access based on your company's specific operational and security needs. IAM policies define the needed access for these identities.



Now that you understand the fundamentals of IAM, let's learn how this works in the **AWS Management Console**.

▼ Demonstration: AWS Identity and Access Management

Now that you understand the basics of IAM, let's see how this really works. In this demonstration, I'll create a new user and then add them to a group. Then I'll create a role that can be assumed to gain access to temporary credentials.

I'll begin by searching "IAM" in the search bar. And then, I will choose the IAM service. The **IAM dashboard** provides a bunch of different resources for managing user permissions. In this demo, we'll be focusing on the **Access management section of the navigation pane**.

The first thing we need to do is to **create** a new **IAM user profile** for an employee named John Doe. I'll select Users from the navigation. And you can see we already have one user here named "**admin**." I will go ahead and select Create user.

And, this brings me to a series of steps that help me define this user's information. In the first step, I'll give my user a **unique name**. `john_doe` works! And then, I'm going to check the checkbox to provide user access to the AWS Management Console so that John can login to the console. Then, I will scroll down and select **Next**.

And now we need to set the **permissions**. At this point, I could choose to **attach permission policies** directly to John's **user identity**. But, we have a standard set of permissions for our employees that I'll need to update from time to time. It would be much more convenient to **manage these permissions all at once**, instead of modifying each employee's permissions individually. Thankfully, we can do this using an **IAM group**. And wouldn't you know it, that's the recommended action! Any existing IAM groups will be listed on this page with the Add user to group option selected. I can conveniently **add John to the appropriate group** if it's already been created.

Because I want to establish a new IAM group, I will select Create group. I will name this group "employees." Next, I'll assign permission policies to the group. I'm presented with a long list of policies to choose from. I can shorten this list by searching for a specific policy.

Or, I can filter the list by choosing from the Filter by Type drop-down menu. By default, this menu is set to show me all permission policy types.

There are two permission policy types to be aware of. **AWS managed policies** and **Customer managed policies**. AWS managed policies are standalone policies **created and administered by AWS**. I can apply these policies to as many users as I want.

I can also create **my own permission policies** if I have a specific use case where I need **custom control**. These policies can be found under the **Customer managed policies**.

For now, I'm going to provide my employees group with the **ViewOnlyAccess** that is managed by AWS. I can add more permissions in the future, if I'd like. I'm curious about how this policy works. So, I can expand it to **view the JSON** and learn more about this specific policy.

Next, I will go ahead and select **Create user group**. And, after the group is created, I can then select it. And now, I can choose **next**. And John will be added to this group, and he will receive the ViewOnlyAccess permissions granted to the group. Any new group members will also have these permissions.

Now we are presented with a page to view the **user details** and **permissions summary**. Everything looks good, so I will choose **Create user** to complete John's user account.

Now let's move on to creating an **identity** that can be **assumed as temporary, rotating credentials** to access AWS resources. For this, I will create an **IAM role**.

I will return to the Users list, select Continue, and then we are brought back to the **dashboard**. Now, I will select **Roles** from the **navigation**, and then **Create role**. For **Trusted Entity Type**, I'll choose **AWS account**. And then I will require **MFA** to turn on multi-factor authentication. This will provide an extra layer of security.

Let's say this temporary role needs **access** to Amazon **S3**. I'll select **Next**. And then we can search for S3 permissions. I'll type in "s3." I'll select **AmazonS3ReadOnlyAccess** and **AmazonS3TablesReadOnlyAccess** permission policies for **minimum access** to our data. Then, I will scroll down and select Next. At this step, I'll enter "s3_read_only" into the **Role name** field and then we can review the role details. Everything looks good, so I will scroll down and then select **Create role**.

Now, someone can **assume** the role and the **needed permissions** to access Amazon **S3**, as defined in the policy. When they assume this role, they will have access to our Amazon S3 bucket, but nothing else.

IAM roles are used for many different use cases, including granting temporary permissions to access specific AWS resources like our S3 bucket. **Roles** allow AWS **services** to **interact with each other** to enable federated identity access for users and applications, and to support **cross-account access** for secure **resource sharing**.

AWS IAM Identity Center

IAM Identity Center centralizes identity and access management across AWS accounts and applications. IAM Identity Center can also connect to an **existing identity source** and provide your workforce with **single sign-on access** to all your connected AWS services and accounts. This is called **federated identity management**.



Federated identity management is a system that allows users to access multiple applications, services, or domains using a single set of credentials.

AWS Secrets Manager

Secrets Manager provides a secure way to manage, **rotate**, and retrieve **database credentials**, **API keys**, and other **secrets** throughout their **lifecycle**. This helps keep your applications, services, and IT resources safe.



Secrets Manager



Secrets are confidential or private information intended to be known only to specific individuals or groups. Examples include passwords, database credentials, and API keys.

AWS Systems Manager

Systems Manager provides a **centralized view of nodes** across your **organization's accounts** and **Regions** and **multi-cloud and hybrid environments**. With this service, you can quickly access **node information**, such as **ID** and **operating system details**, and **automate registry edits, user management, and security patching**.

Nodes are connection points in a network, system, or structure.



AWS Secrets Manager(SM) vs SSM Parameter Store (PS)

Feature	AWS Secrets Manager (SM)	SSS Parameter Store (PS)
Primary Focus	Secrets Management (database credentials, API keys, OAuth tokens).	Configuration Management (app settings, AMI IDs, URLs) and simple secrets.
Automated Rotation	Built-in Native Support for databases (RDS, Redshift, DocumentDB). Primary value proposition.	No built-in rotation. Requires custom code (e.g., a Lambda function scheduled via EventBridge).
Encryption	Mandatory (always encrypted by default). Cannot store plaintext data.	Optional (<code>SecureString</code> for encryption or <code>String</code> for plaintext/config data).
Cost	Paid service (billed per secret stored per month + API calls).	Standard tier is FREE (up to 10,000 parameters) for storage and standard throughput.
Cross-Account Access	Easier to configure, supporting central secrets management.	Supported, but requires more manual IAM setup.
Maximum Size	Can store secrets up to 64 KB in size.	4 KB (Standard) or 8 KB (Advanced tier).
Versioning	Supports staging labels to manage multiple active versions during rotation.	Supports version history, but only one version is active at any time.

A software development team needs to centrally manage its database credentials and API keys on AWS.

Which of these services should the team choose?

- AWS Identity and Access Management (IAM)
- AWS IAM Identity Center

- AWS Secrets Manager
- AWS Systems Manager

Secrets Manager can provide the team with a secure way to manage, rotate, and retrieve database credentials, API keys, and other secrets throughout their lifecycle.

▼ Protecting Network and Applications

Network and Application Protection:

Now, let's discuss how you can **proactively** address some security issues by protecting your network and applications.

You might have heard of **DDoS**, which stands for **distributed denial of service**. It's an attack on your enterprise's infrastructure. Your security team might have written a plan for it, and many businesses have been devastated by it. But what exactly is it, and more importantly, how can you defend against it?

In normal operations, your application takes requests from customers and returns results. In a denial-of-service attack, a bad actor tries to **overwhelm** the **capacity** of your **application**, basically to **deny anyone your services**. But a single machine attacking your application has no hope of providing enough of an attack by itself.

So, the **distributed** part is when the attack uses other machines around the internet to **unknowingly attack** your infrastructure. The bad actor creates an army of zombie bots, brainlessly assaulting your enterprise as you **process an unbearable workload**.

For example, the **UDP** flood Denial of Service attack is based on the helpful parts of the internet, like the National Weather Service. Anyone can send a small request to the weather service, and say, "Give me the weather forecast." In return, the weather service's fleet of machines will send back weather telemetry, forecasts, and updates. To initiate the attack, the **bad actor**, and its army of zombie bots, send a **simple request**: give me the weather forecast. But they give a **fake return address** on the request—**your return address**. So, the weather service very happily **floods** your server with megabytes of rain forecasts. And your system could be brought to a standstill by just sorting through the information it never wanted in the first place. That's just one example of half a dozen low-level, brute-force attacks, all designed to **exhaust your network**.

Fortunately, AWS can **automatically defend** your **infrastructure** from these crippling assaults. How? Let's examine the **UDP flood attack**.

Security groups

Security groups only allow in proper request traffic. They operate at the AWS network level so they can shrug off massive attacks using the entire AWS Region's capacity.

The AWS solution here is **Security groups**. Security groups only allow in **proper request traffic**. Things like weather reports use an entirely **different protocol** than the ones **your customers** use. Not on the list, you don't get to talk to the server. And what's more, **security groups** **operate** at the **AWS network level**, **not at the EC2 instance level**, like an operating system **firewall** might. So, massive attacks like UDP floods just get shrugged off by the scale of the entire **capacity** of the **AWS Region**, **not your individual EC2 capacity**.

Elastic Load Balancing (ELB)

ELB handles traffic first before handing it off, so your frontend server is not overwhelmed. Like security groups, it runs at the Region level.



Another strategy is the use of **AWS managed services**. Simply making the **front door** of your **application** an **Elastic Load Balancer** instead of an **EC2 instance** does a lot to **mitigate an attack**. That's because **AWS Shield Standard** automatically protects AWS resources from the most common, frequently occurring types of DDoS attacks. **Shield Standard is built into AWS managed services like Elastic Load Balancing, CloudFront, and Route 53 at no extra cost**. The **enormous capacity of Regions** makes them **extremely difficult to overwhelm**. It would be massively expensive to achieve.

AWS Shield

AWS Shield Standard is designed to **automatically** protect AWS customers from the most common, frequently occurring types of DDoS attacks at no cost. It uses a variety of analysis techniques to detect and mitigate incoming malicious network traffic in real time.

AWS Shield Advanced is a **paid** service that provides detailed attack diagnostics and the ability to detect and mitigate **sophisticated DDoS attacks**. It also integrates with other services, such as Amazon CloudFront, Amazon Route 53, and ELB.



Additionally, you can integrate AWS Shield with AWS WAF by writing custom rules to mitigate complex DDoS attacks.

AWS WAF

AWS WAF is a **web application firewall** that monitors **network requests** that come into **your web applications**. When a request comes into AWS WAF, it checks the IP address against a web access control list (**web ACL**). If the request comes from a **blocked IP address** on the web ACL, AWS WAF denies access. Legitimate requests are allowed access.



AWS Shield can be **combined** with **AWS Web Application Firewall**, or **WAF**, to further protect your environment. AWS WAF **filters incoming traffic** for the signatures of **bad actors**. It has extensive **machine learning capabilities**, and can **recognize new threats as they evolve**. So, it can **proactively** help **defend** your system against an ever-growing list of destructive vectors.

For even more protection, you can use **AWS Shield Advanced**. This is a **paid service** that provides **detailed attack diagnostics** and the ability to **detect** and **mitigate sophisticated DDoS attacks**.

Unfortunately, DDoS network and application threats exist. But AWS infrastructure and specialized security services can help you stop these attacks to protect your enterprise.

DoS attacks

In a denial of service attack, an attacker floods a web application with excessive network traffic. Legitimate customer requests are denied if the web application becomes overloaded and can no longer respond.

In a distributed denial of service (DDoS) attack, an attacker can use multiple infected computers (called *zombie bots*) to unknowingly send excessive traffic to a web application.

AWS Services for DDoS Protection

Service	Primary Function in DDoS Defense	DDoS Layer(s) Covered	Key Features / Distinction
AWS Shield (Standard/Advanced)	Managed DDoS Protection. Detects and mitigates volumetric and sophisticated attacks.	L3, L4, & L7	Standard is free , automatic protection against common attacks. Advanced provides 24/7 Shield Response Team (SRT) access and DDoS Cost Protection.
AWS WAF	Application Layer Filtering. A Web Application Firewall that monitors and blocks malicious web requests.	Application (L7)	Filters common web exploits (SQLi, XSS) and uses rate-based rules to mitigate

			HTTP floods and sophisticated attacks.
Amazon CloudFront	Volumetric & Caching Defense. Acts as a global Content Delivery Network (CDN) to absorb traffic at the edge.	L3, L4, & L7 (via Caching)	Absorbs traffic across 450+ Points of Presence (PoPs) globally, preventing requests from reaching the origin.
Amazon Route 53	DNS Availability & Resilience. Highly distributed and scaled Domain Name System (DNS) service.	DNS (a form of L3/L4)	Designed to withstand large DNS query floods due to its massive scale and Anycast routing.
Elastic Load Balancing (ELB)	Traffic Distribution & Scalability. Acts as the secure "front door" that scales automatically to absorb traffic volume.	L3 & L4	Scales automatically up to enormous capacity, making it extremely difficult to overwhelm; has Shield Standard built-in.
Security Groups	First-Line Network Filtering (Stateful Firewall). Restricts traffic based on port and protocol.	Network (L3) & Transport (L4)	Shrugs off massive attacks (like UDP floods) by filtering unwanted traffic at the AWS network level before it reaches the EC2 instance.

▼ Protecting Data

Let's return to our coffee shop. Our new app is increasing sales while also saving our customers time! No more long lines. Order your coffee, pick it up, and sip that caffeine goodness.

We need to protect customer data that is shared through the app. This data includes sensitive **personal information**, such as **phone numbers** and **credit card information**. Think about it. The last thing we want is for some hacker to steal our customers' data and run up high bills on our customers' credit cards! I mean, look, we would lose customer trust and potentially face **legal consequences**. These are big no-nos.

So, let's **proactively secure** that **data**. One well-known method is called **encryption**. It's like a lock and key mechanism. If you have the right key, you can open the lock. If not, then you are locked out. In this case, we only provide **authorized parties** with the right key to **encrypt** the **data**. Then, to unlock the lock, they **decrypt** their **data** by using that same exact key.

This comes in two flavors: **encryption at rest** and **encryption in transit**. At rest means that the data is idle and not moving.

For example, the secret recipe for Rudy's Rhubarb Refresher, Trademark, is stored in an Amazon S3 bucket. It's encrypted at rest but... you know, we've gotten to know each other a bit by now... so I'll let you in on the secret.... pssssh come closer... closer... The secret ingredient is....

Alright, let's examine how some AWS services encrypt data at rest.

With Amazon **S3**, all new buckets have **encryption** configured by **default**. Moreover, all **new objects** that are **uploaded** to an S3 bucket are **automatically encrypted at rest**.

Amazon **EBS volumes** and **snapshots** can be encrypted **at rest** as well. In fact, you can **encrypt** both the **boot** and **data volumes** of an **EC2 instance**.

And with **Amazon DynamoDB**, server-side encryption at rest is enabled on all table data using encryption keys stored in **AWS Key Management Service**, or **KMS** for short. In fact, KMS can be used across a wide range of AWS services for that extra bit of control.

KMS helps you **create and manage cryptographic keys**. These keys look like random strings of digits to the human eye, and they can be used to encrypt and decrypt your data. Additionally, you can set **specific levels** of access control for your keys. For example, you can specify which **IAM users and roles are able to manage keys**. You can also **disable keys** so they can no longer be used. And one other important thing to point out is that **your keys never leave KMS**, so you have complete control over them.

Flipping to **encryption in transit**, it means **data** is moving **between locations**. For instance, our coffee shop app needs to access a customer's phone number to text them that their order is ready. This phone number could be stored in a database. But, it would need to be sent over the network from one AWS service to another so we know where to send the text message.

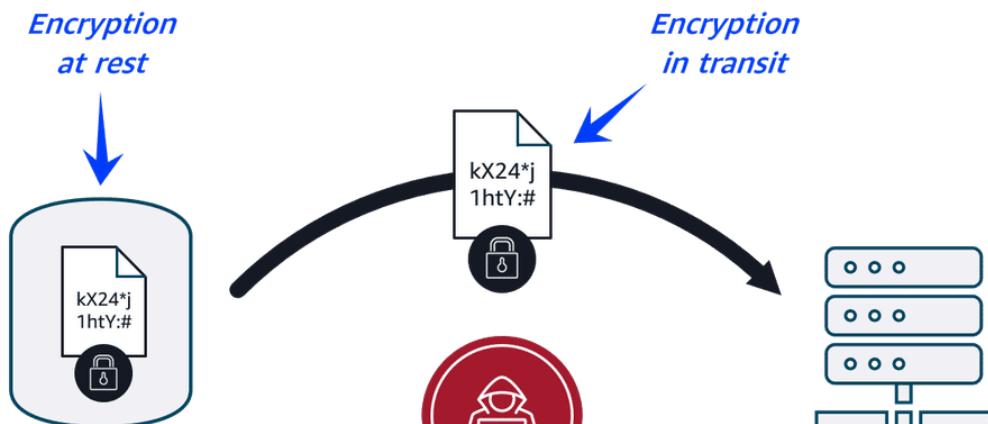
For this, we can use a **protocol** called **Secure Sockets Layer**—or **SSL**, or **Transport Layer Security**—or **TLS**. TLS is an updated version of SSL. With SSL and TLS, you use **certificates** to **verify the identity** and subsequently **establish an encrypted network connection** from one system to another. This means bad actors can't access our customer's sensitive data in-transit. Actually, if you accessed our coffee shop website in a browser, you'd see this lock icon on the left here. The URL would also have **HTTPS** in the front. HTTPS stands for **hypertext transfer protocol secure**, and it means that the **site** is **secured** by an **SSL, or TLS certificate**.

The AWS service that **centralizes management of certificates** is aptly named **AWS Certificate Manager**, or **ACM**. What's nice about ACM is that it can be used to protect various AWS services, in addition to your connected **on-premises resources**.

Types of data encryption

Data encryption comes in the following two forms:

- **Data encryption at rest:** The data is idle and not moving, like when it's stored in a database.
- **Data encryption in transit:** The data is moving between locations, like when it's being sent from a database to an application. SSL/TLS certificates are used to establish encrypted network connections from one system to another.



AWS built-in data protection

- Amazon S3

By default, all new S3 buckets have encryption configured, and all uploaded objects are encrypted at rest.

- **Amazon EBS**

Amazon EBS volumes and snapshots can be encrypted at rest, including both boot and data volumes of an Amazon EC2 instance.

- **Amazon DynamoDB**

Server-side encryption at rest is enabled on all DynamoDB table data using encryption keys stored in AWS Key Management Service (AWS KMS).

AWS data protection services

AWS Key Management Service (AWS KMS)

You can use AWS KMS to create and manage cryptographic keys. These keys can then be used to encrypt and decrypt your data. You can also control the use of keys across a wide range of services and in your applications. For example, you can specify which IAM users and roles can manage keys. Your keys never leave AWS KMS, and you can temporarily disable them so they can no longer be used.



A **cryptographic key** is a random string of digits used for locking (encrypting) and unlocking (decrypting) data.

Amazon Macie

With Amazon Macie, you can **monitor your sensitive data at rest** to make sure it's safe. Macie uses **machine learning** (ML) and **automation** to discover sensitive data stored in Amazon S3. You can use Macie to **assess your security posture**, which is especially helpful for meeting **compliance** requirements.



AWS Certificate Manager (ACM)

ACM centralizes the management of your **SSL/TLS certificates** that provide data encryption **in transit**. It can be used to protect various AWS services and your connected on-premises resources.

SSL/TLS certificates are used to establish encrypted network connections from one system to another.



▼ Detecting and Responding to Security Incidents

By now, you can tell that security is extremely important for all businesses. You should always be working to prevent and proactively address security events. You also need to be able to detect and respond to security issues quickly. Remember our attempted break-in?

Sometimes, security events occur due to unaddressed **software vulnerabilities**, simply because you didn't know they were there.

Amazon Inspector

Amazon Inspector helps improve the security and compliance of applications by running automated security assessments for Amazon **EC2 instances**, **containers**, and **Lambda functions**. It checks applications for security vulnerabilities and deviations from security best practices, such as **open access to EC2 instances** and installations of vulnerable software versions.



Amazon Inspector helps to bring **attention** to these potential vulnerabilities. The way it works is that Amazon Inspector runs **automated security assessments** against your **infrastructure**. It helps to **check** on deviations of **security best practices**, **exposure of Amazon EC2 instances**, and **vulnerable software version installations**.

After Amazon Inspector has performed an **assessment**, it provides you with a **list of security findings** prioritized by severity level in the Amazon Inspector **console**. Each security **issue** includes a **detailed description** and a **recommendation** for how to fix it. You can also **retrieve** findings through an **API**.

But wouldn't it be nice if we had something looking for security threats across some of the other resources in our account, too?

Amazon GuardDuty

Amazon GuardDuty provides **intelligent threat detection** across your infrastructure and resources. GuardDuty identifies threats by continuously monitoring streams of your account metadata and network activity in your environment. It uses known malicious IP addresses, anomaly detection, and machine learning to identify threats more accurately.



Well, that's why we have **Amazon GuardDuty**. This service **analyzes continuous streams** of your **account metadata** and **network activity** as it looks for threats. It uses integrated **threat intelligence**, such as known **malicious IP addresses**, **anomaly detection**, and **machine learning** to **identify threats** more accurately.

You can review detailed findings about any GuardDuty detected threats in the AWS Management Console. Findings include **recommended steps for remediation**. You can also configure AWS Lambda functions to perform remediation steps automatically.

Amazon Detective

After a threat has been detected, you can use Amazon Detective to further investigate the root cause. Detective helps you analyze threats with interactive visualizations contained in a unified AWS Management Console view. These visualizations include resource and user interactions over a configurable timeline with recommended steps for remediation.



After you've **detected** a security **issue**, you can use **Amazon Detective** to uncover the **root cause**. Amazon Detective streamlines the investigative process across your AWS accounts. This service **automatically collects log data** from your AWS resources and uses **machine learning** and **graph analytics** to build interactive **visualizations** of detected **issues**.

These **visualizations** provide a **unified, interactive** view of your **resource** and **user interactions** over a configurable **timeline**. These insights help you to quickly comprehend security threats so you can focus on fixing problems.

AWS Security Hub

Security Hub brings multiple security services together into a **single place and format**. With this service, you can quickly see your security and compliance state in one comprehensive view. Security Hub automatically aggregates security findings from AWS and **partner services** and organizes them into actionable, meaningful groupings called **insights**. It can accelerate time to resolution (TTR) with **automated remediation**.



As you can tell, AWS provides multiple services to help keep your resources secure. But it would be really useful if we could somehow consolidate them all in one place. **AWS Security Hub** was designed for this exact purpose. With this service, you can quickly see your **AWS security and compliance state** in **one comprehensive view**. Security Hub **automatically** aggregates findings and organizes them into **actionable, meaningful groupings** called **insights**.

This helps you efficiently maintain a secure, and compliant environment.

AWS Security Services Comparison

CCP AWS

Service Name	Primary Function	Key Focus (What it Looks For)	Output / Action
Amazon Inspector	Vulnerability Assessment and Compliance Checking	Proactive security flaws in resources, such as: * Vulnerable software versions (CVEs) * Unintended network exposure * Deviations from security best practices	Produces a list of security findings prioritized by severity with recommendations on how to fix the code/configuration.
Amazon GuardDuty	Intelligent Threat Detection and Continuous Monitoring	Active, Malicious Activity and Anomalous Behavior, such as: * Compromised EC2 instances/credentials * Unauthorized API calls * Communication with known malicious IPs	Generates detailed threat findings in near real-time, often leading to automated responses via Lambda/EventBridge.

Amazon Detective	Security Investigation and Root Cause Analysis	Relationships and Historical Context between resources, users, and findings over time to establish a timeline.	Creates interactive visualizations (using graph theory and ML) to uncover the <i>who, what, when, and how</i> of a security incident.
AWS Security Hub	Centralized Security Posture Management and Compliance	Consolidation and Standardization of security alerts (findings) from dozens of integrated services (including the three above).	Provides a single, unified dashboard view of your overall security state, compliance checks, and prioritized insights (actionable groupings).

▼ Additional Security Resources

AWS security documentation

There are lots of considerations when dealing with security on AWS. Make sure you read the documentation on securing your AWS resources because it varies from service to service.

- For general information on AWS security, identity, and compliance services, refer to [Security, Identity, and Compliance on AWS](#)(opens in a new tab).
- To find answers to questions, troubleshoot issues, and learn more about AWS security services, refer to the [Knowledge Center](#)(opens in a new tab).
- To search through documentation by product category, refer to [AWS Security Documentation](#)(opens in a new tab).
- For expert insights, best practices, and updates on security-related features, refer to the [AWS Security Blog](#)(opens in a new tab).

AWS Marketplace security resources

The AWS Marketplace provides a digital catalog where you can purchase third-party software and services that run on AWS. This includes the following types of security services.

Threat detection and prevention tools	Identity and access management tools	Data protection tools	Compliance and governance tools
Identify and block malicious activities.	Control user permissions and authentication.	Encrypt and safeguard sensitive information.	Meet security regulatory requirements.

Module 10 -Monitoring, Compliance and Governance in the AWS Cloud

▼ Introduction to Monitoring, Compliance and Governance in the AWS Cloud

Monitoring your resources in the AWS Cloud

To effectively monitor your Amazon Web Services (AWS) Cloud solutions, you will need ways to provide insights into resource utilization, identify potential issues, and facilitate proactive problem resolution.

The progression you generally want to use is as follows:

1. Securing systems

Protect data, systems, and infrastructure from **unauthorized access**, use, disclosure, disruption, modification, or destruction.

2. Monitoring activities

Continuously observe and analyze system activity, network traffic, and security events to detect potential threats or **anomalies**.

3. Conducting audits

Periodically review and **assess** the **effectiveness of security controls** and **check** that all **requirements** are met and security policies and procedures are adhered to.

4. Ensuring compliance

Help ensure that an organization's security practices and controls meet the requirements of relevant **regulations, industry standards**, and **contractual obligations**.

▼ Introduction to Monitoring

As the owner of the coffee shop, I want to watch what's going on throughout the day to make sure that things are running smoothly. In the coffee shop, I can see that people are getting their coffees, and things look generally fine. But I can't just sit there and watch things all day long. As the owner, I am a pretty important person. And, I would like to eventually leave and then get a **report** about how things went.

For example, I might want to know how many coffees were sold any given day. How long was the average wait time for someone when ordering a coffee? Did we run out of any inventory for anything today? Even better yet, I'd love to be **automatically alerted** if the wait times become too long.

Every business, including this coffee shop, can use **metrics** to measure how well systems and processes are running. This idea of **observing systems, collecting metrics, evaluating** those metrics over time, and then using them to make **decisions** or take **actions** is what we call **monitoring**.

It's important to set up monitoring in the cloud. AWS services can dynamically scale up and down. So, you'll want to keep a close watch on your AWS resources to ensure that your systems are running as expected.

For example, if an EC2 instance is being over-utilized, a scaling event can be triggered to automatically launch another EC2 instance using EC2 Auto Scaling. Or if an application starts sending error responses at an unusually high rate, notifications can be sent to the employees to troubleshoot the issue.

AWS monitoring tools help you **measure** how your **systems** are **performing**, **alert** you when things aren't right, and even help you **debug** and **troubleshoot issues** as they come along.

Monitoring your cloud resources is important. It provides a way for you to **continuously observe** and **analyze** system **activity**, **network traffic**, and **security events** to **detect** potential threats or **anomalies**. Monitoring and observability are critical components for ensuring the **security**, **availability**, **reliability**, and **performance** of your cloud-based workloads and data.

Monitoring is performed using real-time monitoring tools, log collection and analysis, and dashboards.

▼ Amazon CloudWatch

In both our coffee shop and the AWS Cloud, there are so many systems to monitor. Wouldn't it be nice to have one **centralized place** to go? Yep. That's where **Amazon CloudWatch** comes in. CloudWatch makes it possible for you to monitor your **infrastructure** and the **applications** you run on AWS. It accomplishes this through **monitoring and tracking of metrics**.

Metrics are **variables** that are **tied** to your **resources**. For example, the number of espressos made by an espresso machine or the **CPU utilization** of an Amazon **EC2 instance**. For espresso machines, they need to be cleaned after every 1000 espressos. So, we go into CloudWatch and create a **custom metric** called Espresso Count. This tells us what the espresso count is at any given time. But after it reaches 1000, how do we let our employees know they need to clean the machine?

Well, with another CloudWatch feature called **CloudWatch alarms**. We create an alarm and then set the **threshold** for a **metric**. After the threshold is reached, the **alarm** is **triggered** and we can **add a corresponding action**.

With our Espresso Count metric, we set the threshold to 1000 and set the action to let the manager know the machine needs to be cleaned. Even better, because **CloudWatch alarms** are **integrated with Amazon SNS**, we can set up the action to send a **text message** to the manager directly.

We can create all sorts of custom alarms for metrics from all different types of AWS resources. But I can see you asking, "Rudy, where do I see all these metrics?" I got you, boo. You can use the **CloudWatch dashboard** feature. A dashboard, or screen, **lists out all the metrics in near real time**.

In our case, we create a CloudWatch dashboard that shows us all our espresso machines and their espresso counts. No need to click the refresh button on your browser either. These dashboards **auto refresh** so you can see up-to-date information with ease. But where does all the data on the dashboard get stored? And what if we want to review what happened in the coffee shop, say last week, or search for an issue on a specific machine, or even **analyze data further**?

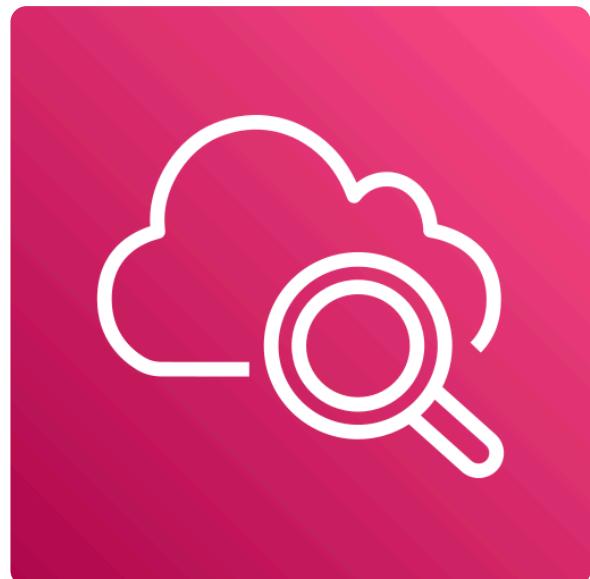
That's where **CloudWatch logs** can help. This feature **centralizes log management and analysis** by **collecting**, **storing**, and **monitoring log files from AWS resources**. You can **view**, **search**, and **filter logs**, depending on what you need. In fact, it's very useful for **filtering certain error codes** or searching for why espresso machine two started making decaf coffee last week!

Let's wrap up by touching on the benefits of CloudWatch. The first is **centralized access** to all your **metrics** from **all** your different **sources**. That means all your AWS resources and your **on-premises servers**. No more siloed monitoring. Use a single pane of glass for **system-wide visibility**. Reduce **mean time to resolution** or **MTTR**. Improve **total cost of ownership**, or **TCO**, and much more.

In application terms, this means freeing up important resources like developers to focus on adding business value like coding the next great app. Lastly, you can drive insights to optimize applications and operational resources. For example, **aggregating usage across** an entire fleet of **EC2 instances** to derive **operational** and **utilization insights**.

Amazon CloudWatch

CloudWatch monitors your AWS resources and the applications that you run on AWS in real time. With CloudWatch, you gain system-wide visibility into resource utilization, application performance, and operational health. CloudWatch does more than just monitor. It has several features that work together:



- **CloudWatch metrics**

CloudWatch **collects metrics** from all your AWS resources, applications, and services that run on AWS and on-premises servers.

- **CloudWatch alarms**

With CloudWatch alarms, you can **define thresholds** on CloudWatch metrics and **send notifications** or **automatically make changes** to the **resources**.

- **CloudWatch dashboards**

Dashboards are customizable home pages in the CloudWatch console that you can use to monitor your resources in a **single view**.

- **CloudWatch logs**

CloudWatch Logs **centralize** the **logs** from all of the systems, applications, and AWS services that you use.

Example: A retail company is using CloudWatch features to **monitor** their **application** running on Amazon Elastic Compute Cloud (Amazon **EC2**) instances. CloudWatch **automatically collects metrics**, like utilization, on the EC2 instances. The company sets up CloudWatch to collect **logs** on the **application performance**. They also have **alarms** for when the Amazon **EC2** utilization gets too **high** for an extended period. They even have an **action configured** to **automate** and **scale** up the number of **EC2** instances when the alarm sounds. Finally, they create a **custom dashboard** to visualize everything all in one place. Now they can **analyze the logs** to gain **insights** on **performance** issues or application **errors**.

▼ AWS CloudTrail

Being able to **audit transactions** of IT events is a critical ability. In a physical datacenter there are so many places where a human can, even by accident, make changes without any record of that change getting saved.

In the AWS Cloud, that problem goes away because **every action** in AWS is an **API call**. They can be **centrally logged** and **audited** with **AWS CloudTrail**. This service is straightforward. It **logs every request made to AWS**. It doesn't matter if it's to launch an Amazon Elastic Compute Cloud, or Amazon EC2 instance, or add a row to an Amazon DynamoDB table, or change a user's permissions.

Every request gets logged in CloudTrail. It **records** exactly **who** made the request, **what** operator, and **when** the API call was sent. **Where** were they? What was their **IP address**? And what was the **response**? Did something change? And what is the new state? Or was the **request denied**?

This is great for us, and this makes an auditor's job much easier which in turn makes our job easier! CloudTrail is an **audit log** of the **AWS API calls**, so it is a **history of configurations and changes** in a **deployment**. CloudTrail even comes with features to **validate log file integrity** to **show evidence of tampering**.

Furthermore, if we are concerned an actor with **root level permissions** could still **tamper** with the CloudTrail **logs**, we can **ship the logs to another AWS account** which by design has **different permissions** to further **protect** the **log integrity**.

AWS CloudTrail

CloudTrail tracks **user activity** and **API usage** in the **AWS Cloud**, **on premises**, and even with **other cloud providers**.

CloudTrail provides a **detailed history** of API calls, so you can track changes and identify who made them and when. This helps you understand what actions were taken on your AWS resources.



Benefits: CloudTrail provides auditing, security monitoring, and operational **troubleshooting**. It also helps you prove **compliance** and improve your **security** posture.

Use cases: It can be used for compliance and auditing, identifying security incidents, troubleshooting operational issues.

CloudTrail events

CloudTrail events **capture details** about **actions performed** within your AWS account, such as **API calls**, **console actions**, or other activities. Event history provides a viewable, **searchable**, downloadable, and immutable record of the **past 90 days** of management events in an AWS Region. There are **no** CloudTrail **charges** for viewing event history.

CloudTrail logs

CloudTrail monitors events and delivers those **events** as **log files** to your Amazon Simple Storage Service (Amazon **S3**) bucket. Because CloudTrail logs are securely stored, they can be used to **prove compliance** with regulations such as **Payment Card Industry (PCI)** and **Healthcare Insurance Portability and Accountability Act (HIPAA)**.

CloudTrail Insights

CloudTrail Insights **analyzes** your normal **patterns** of API call **volume** and API **error rates**. CloudTrail Insights also generates Insights events when API call volumes and error rates **deviate** from these normal patterns. You can enable CloudTrail Insights in your trails or event data stores to **detect anomalous behavior** and **unusual activity**.

AWS CloudTrail vs. Amazon CloudWatch

AWS CloudTrail vs. Amazon CloudWatch

AWS

Feature	AWS CloudTrail	Amazon CloudWatch
Primary Focus	Governance, Auditing, and Compliance	Operational Health, Performance, and Monitoring
Data Collected	API Activity Logs (User actions, console sign-ins, service calls).	Metrics (CPU, latency, disk I/O), Application Logs, and Events.
Key Question Answered	Who made the call? When did they do it? What resource was changed?	What is the CPU utilization? How fast is the application responding? Are there errors?
Data Granularity	Detailed log of every action (not aggregated).	Aggregated Metrics (e.g., 1-minute or 5-minute intervals).
Data Latency	Higher Latency (Events delivered typically within a few minutes, suitable for historical analysis/auditing).	Near Real-time (Metrics as frequently as every minute or second, critical for instantaneous alerting).
Core Use Cases	Forensic analysis, security incident investigation, regulatory compliance reporting (HIPAA, SOC 2).	Setting performance alarms, auto-scaling triggers, dashboard visualization, and application troubleshooting.
Default Retention	Event history retained for 90 days; infinite retention requires custom S3 Trail.	Metrics retained for 15 months; Logs retained for a configurable period (default is often "Never Expire" or 30 days).

▼ Compliance

For every industry, there are specific standards that need to be upheld, and you could be audited or inspected to ensure that you have met those standards. You rely on documentation, records, and inspections to pass audits and compliance checks as they come along.

Depending on what types of solutions you host on AWS, you will need to ensure that you are up to compliance for whatever standards and regulations your business is specifically held to. If you run software that deals with consumer data in the European Union, you would need to make sure that you're in compliance with general data protection regulation or GDPR. Or if you run healthcare applications in the US, you will need to design your architectures to meet Health Insurance Portability and Accountability Act of 1996, or HIPAA, compliance requirements.

Whatever your compliance need is, you'll need some tools to be able to collect documents and records. You will also need to inspect your AWS environment to check if you meet the compliance regulations that you're under.

The first thing to note is that AWS has already built out data center infrastructure and networking using industry best practices. As an AWS customer, you inherit all the best practices of AWS policies, architecture, and operational processes. AWS complies with a long list of assurance programs that you can find online. This means that segments of your compliance have already been completed, and you can focus on meeting **compliance** within your **own architectures** that you build **on top of AWS**.

The next thing to know in regard to compliance and AWS is that the **AWS Region** you choose to operate out of might help you meet **compliance regulations**. If you can only legally store data in the country that the data is from, you can choose a Region that makes sense for you, and AWS will not automatically replicate data across Regions.

You should also be very aware of the fact that **you own your data in AWS**. As shown in the **AWS Shared Responsibility Model**(AWS Shared Responsibility Model - SRM), **you have complete control over the data that you store in AWS**. You can employ multiple different **encryption** mechanisms to keep your **data safe**, and that varies from service to service. You can either use the features that already exist in many services or **build it yourself on top of AWS**. For a lot of services though, **enabling data protection** is a **configuration setting** on the **resource**.

AWS offers multiple **whitepapers** and **documents** that you can **download** and use for **compliance reports**. One place you can **access** these documents is through a service called **AWS Artifact**. With AWS Artifact, you can gain **access to compliance reports** done by **third parties** who have validated a wide range of compliance standards.

Check out the **AWS Compliance Center**. It will show you **compliance-enabling services** in one place. And you can also use resources like the **AWS Risk and Security whitepaper** which contains a lot of awesome information that you can learn from.

When it comes to compliance, the good news is that the underlying infrastructure "of" the cloud is secure.

Compliance in AWS

Compliance refers to your cloud resources and data adhering to relevant regulations, industry standards, and internal policies regarding security and data protection. AWS helps you meet compliance goals and requirements in the following ways:

- Inheriting the latest security controls that AWS uses on its own infrastructure
- Third-party validation for thousands of global requirements
- Streamlining and automating compliance
- On-demand compliance reports

AWS Artifact

AWS Artifact is a service that provides **no-cost, on-demand access** to AWS security and **compliance reports** and select online agreements.

Benefits: AWS Artifact helps you **manage at scale**, save time with on-demand access to compliance reports, and deploy with more confidence.



Use cases: It can be used to manage **select online agreements** and assess third-party security and compliance.

AWS Artifact consists of two types:

- AWS Artifact **agreements** and AWS Artifact **reports**.

AWS Artifact Agreements

Suppose that your company needs to sign an **agreement** with AWS regarding your **use of certain types of information throughout AWS services**. You can do this through AWS Artifact Agreements.

In AWS Artifact Agreements, you can **review, accept, and manage agreements** for an **individual account and for all your accounts in AWS Organizations**. **Different types of agreements** are offered to address the needs of customers who are subject to **specific regulations**, such as the Health Insurance Portability and Accountability Act (HIPAA).

AWS Artifact Reports

Next, suppose that a member of your company's development team is building an application and needs more information about their responsibility for complying with certain **regulatory standards**. You can advise them to access this **information in AWS Artifact Reports**.

AWS Artifact Reports provide **compliance reports** from **third-party auditors**. These auditors have tested and verified that AWS is compliant with a variety of global, regional, and industry-specific security standards and regulations. **AWS Artifact Reports remains up to date** with the latest reports released. You can provide the AWS audit artifacts to your auditors or regulators as **evidence of AWS security controls**.

To learn more about AWS compliance programs, visit [AWS Compliance Programs](#).

AWS Compliance

The [AWS Compliance portal](#) contains resources to help you learn more about AWS compliance. You can read **customer compliance stories** to discover how companies in regulated industries have solved various compliance, governance, and audit challenges.

You can also access compliance **whitepapers** and **documentation** on topics such as the following:

- AWS answers to key compliance **questions**
- An overview of AWS **risk** and compliance
- An **auditing security checklist**

For essentials and best practices, guides, and workbooks, including training, visit [Additional compliance resources](#)(opens in a new tab).

▼ Auditing AWS Resources for Compliance

You might be wondering how you know if the resources and architectures you create are in compliance with your own company standards.

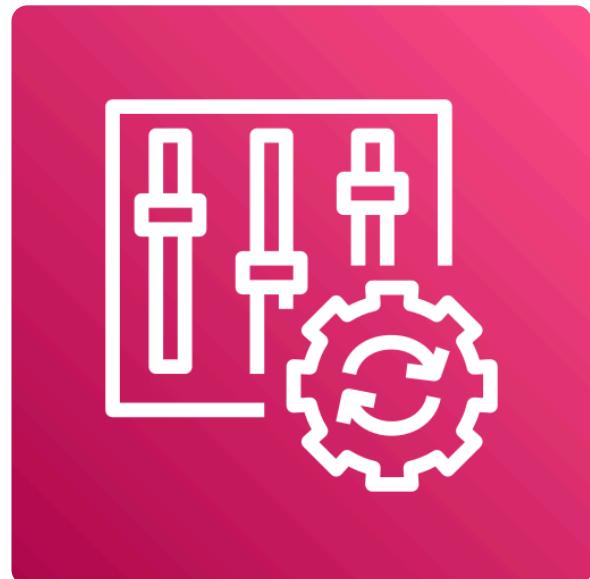
Your **organization** will likely have **rules** you want your **architectures** to adhere to including specific **configuration guidelines** for different AWS resources. You're going to want a way to assess and **audit the resources you create** on AWS to ensure that they meet these rules. That's where **AWS Config** comes in.

AWS Config

AWS Config is a service that you can use to assess, audit, and evaluate the **configurations** of your AWS resources.

Benefits: AWS Config helps evaluate configurations against a **desired state**, manage resource configuration changes, and simplify troubleshooting and remediation.

Use cases: It can be used to continually audit security monitoring and analysis and to streamline operational troubleshooting and change management.



AWS Config is a service that **monitors** and **records** AWS **resource configurations** and **evaluates** them against the **configurations you want to implement**. It helps you **assess**, **audit**, and **evaluate your resources** so your solutions can be in **compliance** with **internal policies** and **regulations**.

Config continuously tracks changes to your AWS resources. You can create **rules** that define your **compliance standards**, and then Config will automatically evaluate if your **resources** meet those **standards**. You can receive **notifications** when Config identifies any **noncompliant** resources, and you can even set up **automated remediation actions** to get your resources **back into compliance**.

You can also **generate reports** showing which resources are **compliant** and which ones are not. After all, compliance **isn't** something you do **once** and you're done. It's an **ongoing** thing. With **Config**, you can **monitor** your **configurations** to ensure that they **don't change over time**.

Sounds like you'd be all set, right? Well, not so fast. If you're in an industry that requires high levels of compliance, it's not enough to be compliant. You must be able to **provide evidence** that you meet the **standards** and **regulations**.

AWS Audit Manager

Audit Manager is a service that **continually audits** your AWS usage to simplify risk and compliance assessment. It helps collect evidence and manage audit data.

Benefits: Audit Manager saves time with automated evidence collection, streamlines collaboration across teams, and helps ensure integrity of audits with read-only permissions.



Use case: It can be used to automate evidence collection, continually audit to assess compliance, and deploy internal risk assessments.

So, for example, a healthcare company might need to provide evidence that patient data is kept private and secure. Or, a financial firm might need to provide evidence to regulators that it's handling money safely and correctly. That's where you can use **AWS Audit Manager**, a **fully managed** service that **automates evidence collection** to reduce the manual effort of several cross-functional teams that audit activities often require.

You can assess if your policies, procedures, and activities, also called **controls**, are operating **effectively**. And it helps you manage **stakeholder reviews** of your controls as well as provides information for building **audit-ready reports** with less manual effort.

As you can see, **Audit Manager** has **pre-built frameworks** that cover a range of compliance standards and **maps your AWS resources** to the **requirements for industry standards and regulations**.

AWS Audit Manager vs. AWS Config Comparison

How They Work Together

1. **AWS Config** acts as your environment's "flight recorder," continuously tracking every change and evaluating compliance against defined rules (e.g., "All S3 buckets must be encrypted").
2. **AWS Audit Manager** takes the **compliance results** (the "Non-compliant" findings) from AWS Config and automatically transforms them into **formal, human-readable evidence** that can be easily packaged into an **audit report** for an auditor.

Feature	AWS Config	AWS Audit Manager
Primary Goal	Continuous Governance & Tracking. Record and assess the configuration state of resources.	Automated Audit Evidence Collection. Simplify the preparation for formal external audits.

Core Functionality	Resource Change Tracking and Compliance Assessment via rules.	Evidence Aggregation and Report Generation using prebuilt frameworks.
Key Question	"Is my EC2 instance compliant with Rule X right now ?" and "What did this security group look like last month?"	"How can I prove to my external auditor that I followed SOC 2 standards last quarter?"
Data Type	Configuration Items (CIs) , Configuration History, and Rule Compliance Status (Compliant/Non-compliant).	Audit-ready Evidence (converted CloudTrail logs, Config findings, Security Hub data, etc.)
Typical Audience	Cloud Engineers, Security Analysts, Governance Teams (daily operational use).	Internal Audit Teams, Compliance Officers, and External Auditors.
Cost Basis	Billed per Configuration Item recorded and per Rule Evaluation .	Billed per Assessor Resource (the AWS services included in the scope of the assessment).

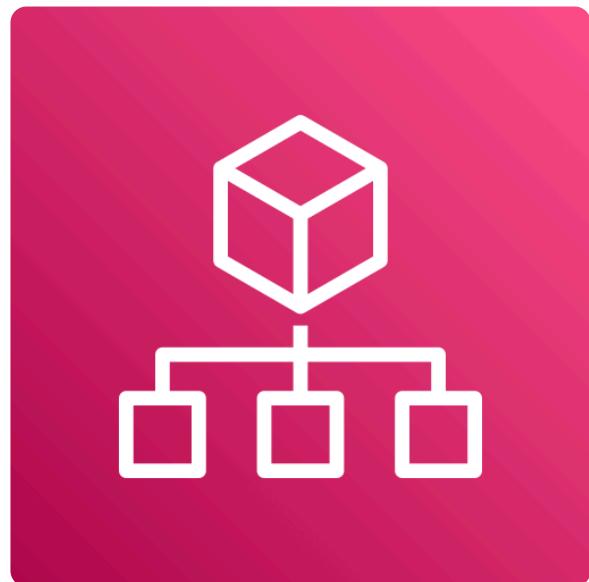
▼ AWS Organization

With your first foray into the AWS Cloud, you will most likely start with a single AWS account. This is great if you are experimenting with AWS and moving test workloads. However, as your company's AWS footprint matures, you will start to create more and more separate AWS Accounts.

These accounts will all have specialized purposes like production and non-production, or an account just for your developers, and another just for your infrastructure team. These accounts could also limit access to certain AWS Services and have other customizations that are not shared across all accounts.

AWS Organizations

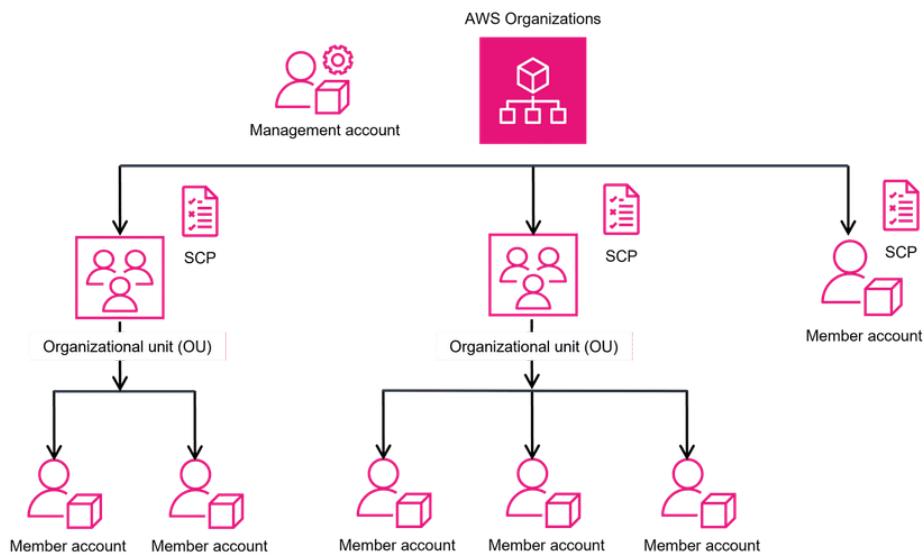
Organizations helps you **centrally manage** and govern your environment as you grow and scale your AWS resources. It helps you manage policies for groups of accounts and **automate account creation**.



Benefits: Organizations provides several benefits like quickly **scaling your environment** by **programmatically creating new AWS accounts** for **resources** and **teams**. It also helps by simplifying permission management through **SCPs** and managing and optimizing costs across your AWS accounts and resources.

Use cases: It can be used for automating AWS account creation, providing tools and access for your security teams, controlling user access to designated services, and sharing common resources across accounts.

Moreover, you need a way to **manage your accounts** since doing it manually is not ideal. This is where **AWS Organizations** can help. Organizations is an **account management service** that allows for the **consolidation and central management of multiple AWS accounts** within an organization. It allows you to manage **billing, access, compliance, and security** as well as **share resources across accounts**.



You designate a main, or **parent account**, and all other accounts are secondary, or **children**, to that main account. Billing is consolidated into the primary account. So, all the bills for the children accounts roll up to the parent account. Then, **discounts** are applied at the **top level**.

Another feature is **hierarchical account groupings** to meet security, compliance, or budgetary needs. For example, you can group all the accounts into **organizational units** or **OUs**. So, if you have accounts that must access only the services that meet certain regulatory requirements, you can put those accounts into one, single OU.

Or if, say, you have accounts that fall under the developer OU, you can **group them** accordingly. As an admin of the primary account of an organization, you even have control over the **AWS services** and **API actions** that each account can access.

You can even use **service control policies**, or **SCPs**, to specify the **maximum permissions** for member accounts in the organization. In essence, with SCPs you can control which services, resources, and individual API actions, the **users** and **roles** in each member account **can access**.

So, consider using Organizations, sooner rather than later, so you don't end up with ghost AWS accounts. After all, nobody wants ghosts running around. That's spooky.

In AWS Organizations, you can apply **SCPs** to the organization root, an **individual member account**, or an **OU**. An SCP affects all IAM users, groups, and roles within an account, including the AWS account root user.

You can apply IAM policies to IAM users, groups, or roles. You cannot apply an IAM policy to the AWS account root user.

▼ Governance

Imagine you're the mayor of a rapidly growing city: Cloud Town. As your city expands, you need a way to oversee all the new buildings, roads, and services while making sure everything follows the city's rules. You want to empower your builders to act autonomously, but you need to ensure **governance**.

Governance is the **framework** that makes it so **all the work of building and operating your deployment is efficient and supports the overall goals**. In Cloud Town, we might want a zoning code limiting the height of buildings because we know we're building an airport later.

You might want to enforce that all **resources** are **tagged** with the **department** that **creates** them for simple **cost tracking**. How can you do this if every AWS account's root user can in theory do whatever they want?

AWS Control Tower

AWS Control Tower is a service you can use to **enforce** and manage governance rules for security, operations, and compliance at scale across all your organizations and accounts in the AWS Cloud.



Benefits: AWS Control Tower can help you **save time** while providing **governance**. It uses **preconfigured controls**, which can help you to quickly set up multi-account environments, **automation with built-in governance**, and **integration of third-party software at scale**.

Use cases: Use AWS Control Tower to quickly deploy applications and provision compliant AWS accounts.

With multiple accounts, that's where **AWS Control Tower** can help. It's like a **high-tech control center** for your cloud city, helping you **govern**, **manage**, and **organize** your AWS environment as it grows. AWS Control Tower is especially helpful because the more people you have building in the cloud, the more complex and time-consuming set up and governance can be.

You need **everyone** creating new accounts and spinning up AWS resources to **follow the rules**. You can provision new AWS accounts, and you have peace of mind knowing your **accounts conform to your company-wide policies**. It helps you **standardize initial account setup according to best practices**.

Let's look at how it works. First, **Control Tower** makes it possible for you to **automate setup** using established **blueprints** for configuring your AWS environment with federated access management and account provisioning workflows.

Next is applying **guardrails**. These are like the **safety barriers** on a highway. They help prevent accidents by **automatically enforcing rules** and best practices across your **AWS environment**.

You can apply **security** and **compliance policies** using established **guardrails** to prevent resources from being deployed that don't conform to policies. Plus, you can **detect** and **remediate noncompliant accounts** and **resources** as your team provisions them.

Guardrails (Controls): A mix of **Preventive** (using SCPs) and **Detective** (using AWS Config rules) policies for continuous compliance monitoring.

Finally, you can **monitor compliance**. Control Tower provides a **visual summary** of your **AWS environment** through a **dashboard** that you can use to observe your **accounts**, **guardrails**, and **compliance status** all in one place.

With Control Tower, you can save time and have greater control over your AWS environment so you can govern your workloads more efficiently while making sure they also conform to your company-wide policies.

So, whether you're starting out on your journey to AWS, building a new AWS environment, or launching a new cloud initiative, Control Tower will help you with governance and best practices built-in to make sure you're successful.

Governance in the AWS Cloud

- **AWS Control Tower**
- **AWS Service Catalog**
- **AWS License Manager**

Landing zone

A landing zone is a well-architected **multi-account environment** that's based on security and **compliance best practices**. It's the enterprise-wide container that holds all of the organizational units (OUs), accounts, users, and resources you want to regulate for compliance.

Managing **employee requests** for new AWS services or **resources** can be time consuming, but you also **don't want everyone guessing which type** of services and settings should be used. That's where **Service Catalog** can help.

A Landing Zone includes **foundational elements** like:

- A multi-account structure (e.g., separate accounts for Logging, Security, and Development).
- Pre-configured security best practices (e.g., centralized logging via CloudTrail).
- Identity and access management ready to go.

AWS Service Catalog

With Service Catalog, you can **create**, **share**, and **organize** from a **curated catalog of AWS resources**. You can **deploy baseline networking resources** and **security tools** for **new AWS accounts** so you can **govern consistently**.



Benefits: Service Catalog saves **time** by making it **quick to find and deploy approved, self-service cloud resources**. It also helps you stay **agile** while improving **governance** over resources across multiple accounts.



Use cases: Use it to provision resources across AWS accounts, apply access controls, and accelerate provisioning of **continuous integration and continuous delivery (CI/CD) pipelines**.

Access Controls + CI/CD

When companies move from on premises to the cloud, they must decide how to handle their software licenses. With **AWS Bring Your Own License** model (**BYOL**), they can use existing **software licenses purchased** directly from vendors, such as Microsoft, on AWS services like **Amazon EC2 Dedicated Hosts** and **Amazon WorkSpaces**. This can result in significant **cost savings** compared to purchasing licenses directly from AWS. By using **BYOL** with **existing licenses** in a cloud environment, you get flexibility and potential optimized costs. The service that helps you **manage** and **govern** your software **licenses** is **AWS License Manager**.

AWS License Manager

License Manager is a service that helps you manage your software licenses and **fine-tune your licensing costs**.

Benefits: License Manager helps with **visibility and control, tracking and managing** licenses, and reducing the **risk** of **noncompliance** with licenses.



Use cases: Use it to streamline license management and to simplify the **Microsoft License Mobility** through Software Assurance experience. You can also use it to **automate** the **distribution and activation of software entitlements** across AWS accounts for end users.

▼ Simpler explanation:

- "Streamline license management"
 - **Simple Meaning:** Keeping track of what you own.

- It helps you maintain a perfect, digital count of all the software licenses your company owns and uses. This prevents your company from accidentally buying too many licenses (wasting money) or using too few (breaking the software company's rules).
- "Simplify the Microsoft License Mobility through Software Assurance experience"
 - **Simple Meaning:** Moving licenses you already own to the cloud legally.
 - This is about saving money on specific Microsoft licenses. If your company already paid for expensive Microsoft software for your old office servers, License Manager handles the tricky rules that let you legally move and use those *existing* licenses on AWS cloud servers.
- "Automate the distribution and activation of software entitlements across AWS accounts for end users"
 - **Simple Meaning:** Giving employees access instantly and automatically.
 - When an employee needs a specific piece of software (the "entitlement"), this service automatically sets up the permission for them to use it across all the different AWS accounts your company might have. IT doesn't have to manually install or activate the software on every new machine.

▼ AWS Health

Health of your AWS Cloud resources

Notifications on service events

AWS Health is the go-to **data source** for **events** and **changes** affecting the health of your AWS Cloud resources. It **notifies you** about service events, planned changes, and account notifications to help you manage and take actions.

AWS Health Dashboard

With AWS Health Dashboard, you can view **account-specific health information** and get AWS Health event **updates**. You can also use AWS Health programmatically using the **AWS Health API**, which is available with **AWS Premium Support**.



Benefits: AWS Health Dashboard provides valuable information as a **data source** for **events** and **changes**. It gives you timely and **actionable guidance** to remedy **issues**. It also helps manage service health and is integrated and **automated** to use at scale.

Use cases: Use AWS Health Dashboard to view account-specific health information. You can also use it to **plan for lifecycle events** or troubleshoot an **incident**.

Example: If AWS Health reports a service degradation affecting a database instance, EventBridge can automatically trigger a Lambda function to failover the database to a healthy Availability Zone (AZ) or notify your support team.

Use **CloudWatch** to monitor your performance and **CloudTrail** for security/auditing. Use **AWS Health** to monitor the health of the platform *underneath* your resources.

AWS Health vs CloudWatch vs CloudTrail

CCP AWS

Feature	AWS Health	Amazon CloudWatch	AWS CloudTrail
Focus	The AWS Cloud (Events/Changes/Health impacting your resources).	Your Resources (Performance, operational health, and metrics).	Your Account Activity (API calls, user actions, and security auditing).
Question Answered	"Is the problem with AWS (e.g., a service degradation, scheduled maintenance) or is it with my application?"	"Is my application (e.g., CPU, latency, errors) performing correctly?"	"Who changed what in my account, and when did they do it?"
Scope	Account-specific and Service-specific events (e.g., "Your specific EC2 host needs maintenance next Tuesday").	Collects metrics and logs from your applications and resources.	Logs every API call made by or on behalf of your account.
Type of Event	Service incidents, planned lifecycle events (e.g., an RDS patch window), or scheduled maintenance.	CPU utilization threshold breaches, high latency, custom application errors.	User "Rudy" terminated an EC2 instance.
Latency	Near instant alerts, focused on proactive and remediation guidance.	Near real-time (1-minute intervals) for operational monitoring.	Delayed (typically minutes) for auditing and forensic analysis.

▼ AWS Trusted Advisor

When running a business, you might need some advisors who can come in from the outside and say, "Hey, this process should be streamlined." Or, "Hey, I have some good tips on how to save money on your overhead." Or even, "Hey, I noticed I was able to waltz right in, go behind the cash register, and open the drawer without anyone noticing."

Not good!

The point I'm trying to make is, sometimes it's nice to have someone who knows the **industry best practices**, and **knows what to look for**, come in, and tell you what you need to change to run more **efficiently**, be more **secure**, or to **lower costs**.

Trusted Advisor

Optimizing large scale cloud deployments is extremely important to do, and it's not a one-time thing. You must look for ways to **optimize for costs, performance, security, and resilience**. With **AWS Trusted Advisor**, you can **continuously evaluate** your AWS environment by using best practice checks across several categories. All AWS Support plans include access to dozens of Trusted Advisor checks. With Business Support and other advanced plans, you can benefit from hundreds of checks.



Benefits: Trusted Advisor helps you align with AWS best practices, prioritize recommendations, and optimize your AWS resources at scale.

Use cases: It can be used to optimize cost, efficiency, security, improve performance, and track service limits.

AWS has an **automated advisor** called **AWS Trusted Advisor**. This is a service that you can use in your AWS account that will **evaluate your resources against five categories** of checks.

The checks look for areas of improvement related to

1. **cost optimization**
2. **performance**
3. **security**
4. **fault tolerance**
5. **service limits**

Trusted Advisor runs through a series of **checks** for each category in your account in **real time**, based on **AWS best practices**. It **compiles categorized items** for you to look into, and you can view them directly in the **AWS Management Console**.

Some checks are included in your AWS account, and others are available, depending on the level of your support plan. For example, if you don't have **multi-factor authentication** turned on for your root user, it's going to **let you know**. It also lets you know if you have **underutilized EC2 instances** that might be able to be **turned off to lower costs**. Another example, is if you have **EBS volumes** that **haven't been backed up in a while**. To get a better idea, let's look at AWS Trusted Advisor in my own account.

Here I am in the **Trusted Advisor dashboard** and I'm going to choose Cost Optimization first to view what it has found. You can see that we have red here, which means action recommended, orange, which means investigation recommended, and green, which means that there were no problems detected.

Lucky for me, I don't have any critical items for Cost Optimization. However, I can go through and read more about each check that Trusted Advisor ran. For example, there is a **check** that **looks** to see if there are any **RDS instances** that are idle in this account. And here is another example of a check that **evaluates ELB utilization**. And then here is another one that checks for **EBS utilization**. If I had resources in this account that got **flagged** with these checks, I could decide to scale these resources down to save on cost. Or, if they aren't being used at all, I could decide to delete them.

Now, let's move on to **Performance**. There are no action recommended or investigation recommended items, but we can **review** some of the **checks** that Trusted Advisor performed. Scrolling down, I can **type** in **EBS**, and there is this one here that checks for EBS volumes whose **performance** might have been **affected** by the **throughput capability of the EC2 instance** that it's **attached** to.

Next up, let's check out **Security**. In this demo account, you can see there are two alerts that are at the **action recommended level**. There are some **security groups allowing public access to EC2 instances**, which is putting those resources at **risk**, and this should be **resolved**.

Now let's move on to **Fault tolerance**. For this, there were a few things that are **found** to be **insufficient**. First, Trusted Advisor is telling me that there are **EBS volumes without snapshots** in this account. Remember, a snapshot is a **backup**. So, without backups, if I had an EBS volume fail, I would lose that data.

Another alert we have here is that my **EC2 AZ balance** is at the **action recommended level**. This means that my **EC2 workload** is **not** properly launched across AZs. So, if one AZ or instance has trouble, my application might have a **disruption**. **Deploying across AZs** would be the answer to this one.

And then lastly, we have one for Amazon **S3 Bucket Logging**. And this is checking to see if **logging is enabled properly** for my **S3 buckets**, which I have some buckets in this account that do not have proper logging.

Finally, let's go ahead and move onto **Service limits**. Service limits tell you when you are **approaching** or hitting any **AWS service limits**. A lot of these are **soft limits**, meaning they are **restrictions that can be lifted to some degree**. It's good to know when you are approaching one of those limits, and when it's time to use the **Service Quota console**, which you can use to **request a change to these limits**. There is nothing to report for this specific account on that front.

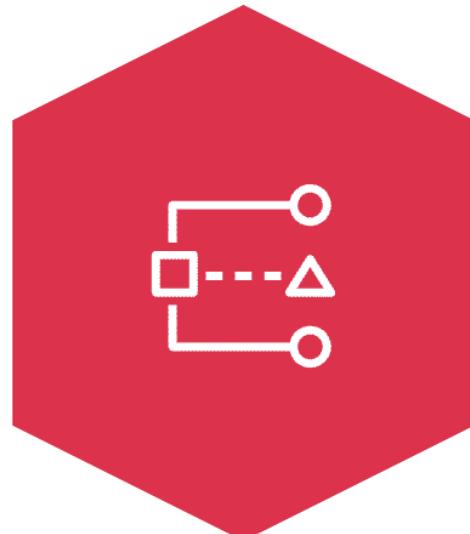
Overall, you can see how **Trusted Advisor** can help **point you in the right direction**. You can set up email alerts that go out to billing, operations, and security contacts, as checks are run in your account. Make sure you have Trusted Advisor turned on so that you, too, can start taking action to optimize your AWS account.

Although Trusted Advisor does check to optimize security, you might need help to check the fine-grained permissions of your AWS Identity and Access Management (IAM). IAM Access Analyzer can help meet your goals for least privilege access within your AWS environment.

IAM Access Analyzer

IAM Access Analyzer provides capabilities to **set**, **verify**, and **refine permissions** by **analyzing external access** and **validating** that your policies **match your corporate security standards**.

Benefits: IAM Access Analyzer provides benefits like refining permissions, **validating IAM policies**, helping you meet your **least privilege goals**, and **automating IAM policy reviews**.



Use cases: It can be used to set fine-grained permissions, verify who can access what, remediate **unused access**, and **refine and remove broad access**.

Module 11 - Pricing and Support

▼ Introduction to Pricing and Support

We've covered a lot at this point. Now it's time to talk about pricing and support.

Just like how we have monthly bills that we need to track and manage for the coffee shop, you'll have to manage costs and understand pricing for your AWS resources. In this section, we'll dive into key pricing concepts, helping you understand how AWS charges for its services. Then, you'll learn more about how you can use services and features like the **AWS Billing Dashboard**, **AWS Budgets**, and **AWS Cost Explorer** to help you track your AWS **spending**, **set budgets**, and **forecast future costs**.

Now, in our coffee shop, what happens when the espresso machine isn't functioning like we'd expect or if we have a question about our lease? We would want reliable support, right? We'd reach out to the equipment provider and locate their resources to troubleshoot whatever issue or question we might have. Well, when it comes to your AWS resources, you have a lot of options for support. In this section, we'll also examine **support plans** and resources, and you can find which level of support is right for your business.

And just like we'd look for ways to reduce waste and increase efficiency in the coffee shop, AWS has some great resources for optimizing cost in the cloud. We'll finish the section by looking at some use cases that illustrate cost optimization with AWS.

▼ AWS Pricing Concepts

You've heard a little bit about AWS pricing concepts here and there. Let's unpack AWS pricing a bit more as well as talk about the main drivers of cost. Let's start with the three fundamental principles of AWS pricing.

Pay as you go

With pay as you go, you can adapt to changing business needs and reduce the risk of overprovisioning or missing capacity.

First, a concept you're familiar with: **pay as you go**. With this model, you only pay for the resources you actually **consume**, without any **upfront costs** or **long-term commitments**. It's perfect for **variable workloads** or when you're unsure about your exact needs.

Save when you commit

For certain services, such as Compute services on AWS, Savings Plans offer savings over On-Demand prices when you commit to a 1-year or 3-year plan.

The next key concept is that you **save** when you **commit**. By committing to a **certain level of usage** for a certain **period of time**, usually **1 or 3 years**, you can receive pretty significant discounts. This is ideal for **steady-state workloads**.

Pay less by using more

With AWS, you can realize important savings as your usage increases. For some services, pricing is tiered, meaning the more you use, the less you pay.

The third concept is that you pay less by using more. I know, this one sounds kind of similar to save when you commit. Here's what we mean—we are talking about **volume-based discounts**. As your usage of AWS services increases, you can benefit from **economies of scale**, resulting in **lower per-unit costs**.

OK, so now you have the fundamental concepts down. You know the how, so let's discuss the what—what are the **primary drivers of cost** at AWS? Well, the technical answer is that it depends. Every AWS service bills differently, so you'll want to check the **pricing page** of each service to know the full details and associated costs. That said, there are pretty much **three main drivers** when it comes to AWS pricing: **compute**, **storage**, and **outbound data transfer**.

Compute cost includes services like **EC2**, **Lambda**, and **ECS**, when you're charged based on the **processing power** and **time used**. Storage includes services like **S3** and **EBS**, and pricing is based on the **amount of data stored** and for **how long**. And then we have **outbound data transfer**. One example for this, is if we host a static website in S3, anytime someone **accesses objects** from that bucket that data is being **transferred** out of AWS and would **incur charges**.

Now, there is nuance to this. Each service has different considerations, so ensure you are reading the documentation so that you fully understand how billing works.

Driving factors of cost

The pricing of AWS services varies based on several factors, such as service category or type, configuration, **AWS Region**, and which **pricing model** you choose. Refer to the pricing tab on a service's webpage for details on its specific pricing factors.

Compute

For compute resources, you pay by a certain span of time, like by the hour or by the second. Unless you've made a reservation for which the cost is agreed upon beforehand, you pay from the time you launch a resource until the time you stop the instance.

Storage

You can choose from a broad portfolio of storage solutions with deep functionality for storing, accessing, protecting, and **analyzing** data. Pricing for storage largely depends on how much storage you have provisioned or how much you are using.

For some storage options, such as Amazon Simple Storage Service (Amazon S3), storage cost is **tiered**. This means you can **optimize storage costs** based on how **frequently** and **quickly** you need to **access** data. With Amazon S3, consider the following **six cost components** when storing and managing customer data:

- **Storage pricing**
- **Request and data retrieval pricing**
- **Data transfer and transfer acceleration pricing**
- **Data management and analytics pricing**
- **Replication pricing**
- **The price to process your data with Amazon S3 Object Lambda**

To learn more about Amazon S3 pricing, choose the following link and navigate through the pricing tabs: [Amazon S3 Pricing](#)(opens in a new tab).

Data transfer

In **most cases**, there is **no charge for inbound** data transfer or for **data transfer between AWS services** within the **same Region**. There are some **exceptions**, so be sure to verify data transfer rates before beginning.

Outbound data transfer is aggregated across services and then **charged at the outbound data transfer rate**. The **more data** you transfer, the **less you pay per gigabyte**. For data storage and transfer, you typically pay per gigabyte.

AWS pricing scenario

Let's apply these AWS pricing concepts and drivers of cost to a possible scenario. Use Amazon Elastic Compute Cloud (Amazon **EC2**), a service in the **compute** category, as an example.

An *AWS customer decides to provision an EC2 instance for their nonprofit organization. To learn how this customer's EC2 instance might be impacted by the driving cost factors of compute, storage, and data transfer, choose each of the numbered markers.*

Compute:

The organization built their own **application** that processes and manages **online donations**. There are over **500 types of EC2 instances** that they can choose from to run that application. Each type of EC2 instance has a different quantity of **CPU** and **memory**, both of which are **compute factors** of Amazon **EC2**. Based on their requirements, they chose to use **t4g.nano instance**, which provides the **lowest price** while still meeting their needs.

Storage

The organization's donation processing application requires **storage**, so they configure their **EC2** instance with Amazon **Elastic Block Store** (Amazon **EBS**). Price will be impacted by the **amount of capacity** configured.

Data transfer

The organization's donation processing application will **transfer data** from their **EC2** instance to another solution to run **analytics**. This transfer would be considered **outbound** data transfer. Price will be impacted by **capacity** and **which internet source** or **AWS Region** the data is being transferred to.

For the chart associated with the cost of data transfer out, navigate to the following webpage and scroll to "Data Transfer": [Data transfer on Amazon EC2](#)(opens in a new tab).

▼ AWS Pricing and Billing Services

OK, now that pricing concepts are fresh in your mind, let's talk about billing options for AWS accounts: **single** or **consolidated**. With a **single** account, it's all encompassed in that one account. Use AWS services. Deploy workloads. Receive a bill for what you use. **Rinse and repeat every month**.

For **consolidated** account billing, it's ever so slightly more complex. With this approach, you use **multiple AWS accounts**. One is a **primary** and the others are **subaccounts** linked to the primary account. The subaccounts can be separated by **department**, **project**, **team**, or however you see fit. The important part is that when your AWS **bill arrives**, it is only sent to the **primary account**. All other account bills are **consolidated** into that **primary account**. In addition to consolidated billing, the **primary account owner** gets a clear view of your **entire organization's AWS spend**.

And I use the word organization specifically because, yep, you guessed it: You can use **AWS Organizations** to set it all up. With **Organizations** and **consolidated billing**, you can take advantage of some great benefits, like **centralized management**, **discounts**, and **improved security**.

OK, with that out the way, it's demo time! The first service that can help with billing is the **AWS Billing Dashboard**. It gives you an **overview** of your **forecasted spend** for the **month**, and a breakdown of your **most-consumed services** by **cost**. It's a great starting point for understanding your expenses. But what if you want to budget and plan things out ahead of time?

Billing and Cost Management

Choose billing view New Primary view

Home Getting Started Billing and Payments Bills Payments Credits Purchase Orders Cost and Usage Analysis Cost Explorer Cost Explorer Saved Reports Cost Anomaly Detection Free Tier Data Exports Customer Carbon Footprint Tool Cost Organization Cost Categories Cost Allocation Tags Billing Conductor Budgets and Plans Budgets New Advanced Details

AWS Billing and Cost Management Console Home

Billing and Cost Management home

Cost summary

Month-to-date cost: Last month's cost for same time period Mar 1 – 30

Total forecasted cost for current month: Last month's total cost

View bill

Cost breakdown

Group costs by Service Costs (\$)

Recommended actions (0)

No recommended actions

Well, check out **AWS Budgets**. You can create **custom budgets** to track your **costs** and **usage**. You can generate budgets for specific services, cost categories, or even by tags. **Tags** are **metadata** that you can **add** to your **AWS resources**. They help you **identify resources easily** and **group them** together to find all related resources much quicker. For example, if we tag all our development resources with the "dev" tag, we can easily find all the related resources with one search instead of multiple. One final note on **AWS Budgets** is that you can **set up alerts**. These alerts can notify you when you're approaching or have exceeded budget **limits** that you have set.

Finally, let's look at **AWS Cost Explorer**. It helps with **reviewing costs** and **usage over time**. You can **break down costs** by **service**, **linked account**, and again, even **tags**. For example, you might tag resources by project, department, or environment. With tag-based cost allocation in Cost Explorer, you can use these tags to organize your cost and usage information. This is incredibly useful for understanding which **projects** or **departments** are **driving** your **AWS costs**.

Billing and Cost Management

Choose billing view New Primary view

Home Getting Started Billing and Payments Bills Payments Credits Purchase Orders Cost and Usage Analysis Cost Explorer Cost Explorer Saved Reports Cost Anomaly Detection Free Tier Data Exports Customer Carbon Footprint Tool Cost Organization Cost Categories Cost Allocation Tags Billing Conductor Budgets and Planning Budgets New Advanced Details

New cost and usage report

Cost and usage graph

Total cost: \$46,967.19 Average monthly cost: \$7,827.87 Service count: 22

Costs (\$)

Cost and usage breakdown

Oct 2024 Feb 2025 Mar 2025

VPC GuardDuty Systems Manager Others

AWS Cost Explorer

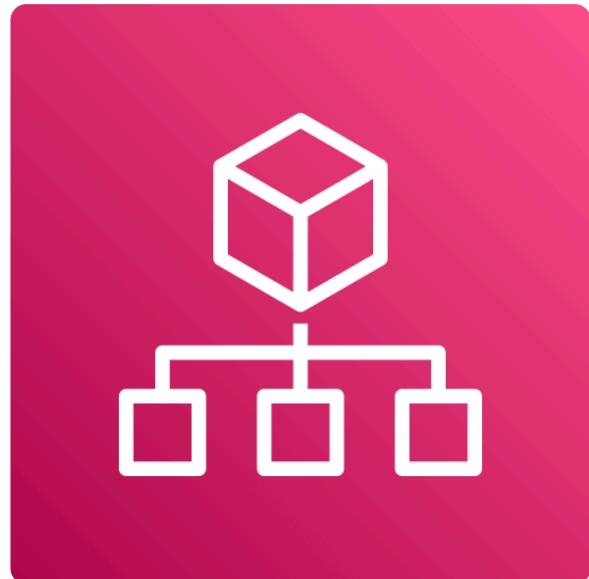
Download as CSV

AWS pricing and billing services

You learned about various AWS pricing and billing services and tools. These services are purpose-built to help you forecast, track, manage, and view your AWS costs. When you are first starting out with the AWS Cloud, they can be hard to tell apart. To help you differentiate the services, let's review these services and their key use cases.

AWS Organizations

AWS Organizations provides **centralized management and governance** of your AWS environment. Using AWS Organizations, you can create, group, and manage accounts. You can also apply security policies at the account level and consolidate billing with multiple accounts using a single payment method.



Use cases:

- **Consolidate** multiple AWS accounts into one central organization.
- Implement **organization-wide policies**.

To learn more about AWS Organizations, refer to the [AWS Organizations User Guide](#)(opens in a new tab).

AWS Billing and Cost Management dashboard

The AWS Billing and Cost Management dashboard **centralizes cost management**, showing **current charges**, usage, **forecasts**, and detailed **breakdowns**. It also provides **tools** to manage **payments**, view **invoices**, set **budgets**, and **consolidate billing**.



Use cases:

- Use helpful **visualizations** and billing reports of monthly AWS spend.
- Set up and manage **payment methods**.

To learn more about the AWS Billing and Cost Management dashboard, refer to the [AWS Billing User Guide](#)(opens in a new tab).

Feature	AWS Organizations
---------	-------------------

		Billing & Cost Mgmt Dashboard
Focus	Governance, Structure, and Permissions	Financial Reporting, Invoicing, and Payment
Core Action	Creating and grouping accounts (OUs) and setting SCPs .	Viewing bills, paying invoices, setting up payment profiles.
Billing Role	Enables Consolidated Billing (The engine).	Displays the Consolidated Bill (The speedometer/invoice).
Data Type	Account IDs, Organizational Units, Policy documents.	Dollar amounts, invoices, payment history, cost forecasts.
Key Distinction	Manages who can do what at the account level.	Manages how much was spent and how to pay for it.

AWS Budgets

AWS Budgets helps **set custom budgets** and sends **alerts** when costs, usage, or **Savings Plans and Reserved Instances** (RIs) utilization or coverage exceed defined **thresholds**.



Use cases:

- Set up alerts for when projected costs **exceed** predefined thresholds.
- **Forecast** future expenses based on **current usage trends**.

To learn more, refer to [Managing your costs with AWS Budgets](#)(opens in a new tab).

AWS Cost Explorer

AWS Cost Explorer helps **visualize**, **analyze**, and manage AWS costs and usage with **interactive graphs**, **reports**, and **forecasts**. It provides **insights** into **spending patterns**, **trends**, and **Reserved Instance recommendations**.



Use cases:

- **Analyze historical spending trends** to identify cost-saving opportunities.
- Forecast future AWS costs based on current usage patterns to budget effectively.

To learn more, refer to [Analyzing your costs and usage with AWS Cost Explorer](#)(opens in a new tab).

AWS Pricing Calculator

Another helpful tool is the AWS Pricing Calculator. The AWS Pricing Calculator is a **web-based planning tool** that you can use to create **estimates**. You input specific **configurations** such as instance types, storage options, and data transfer volumes. Then, based on your configurations, you receive a **detailed cost breakdown** to help you budget for your AWS resource allocation.



Use cases:

- **Estimate potential costs before deployment.**
- **Compare costs of different AWS services and configurations.**

To learn more, refer to [Generating estimates with AWS Pricing Calculator](#)(opens in a new tab).

AWS Cost and Governance Services Comparison

AWS Cost and Governance Services Comparison

Service Name	Primary Focus / Goal	Visualization & Analysis	Forecasting & Estimation	Key Distinction
AWS Organizations	Account Structure & Centralized Billing	Minimal/None	None	Governance Framework: Manages account structure (OUs) and enables consolidated billing and SCPs.
Billing & Cost Mgmt Dashboard	Centralized Summary & Invoicing	Shows current charges and usage reports.	Shows simple current usage forecasts.	Financial Management: The single source for managing invoices, payments, and tax information.
AWS Cost Explorer	Deep Historical Cost Analysis	Provides interactive graphs and detailed, customizable reports on historical spending.	Generates detailed forecasts of future costs based on historical usage.	Optimization: Provides Savings Plan & RI Recommendations by analyzing historical data.
AWS Budgets	Cost Control & Alerting	Shows usage/cost relative to the defined budget threshold.	Forecasts usage/cost against the custom thresholds set.	Active Control: The only service designed to send alerts and trigger actions when costs or usage exceed a limit.
AWS Pricing Calculator	Pre-Deployment Cost Estimation	Provides a detailed breakdown of estimated costs for planned services.	Generates estimates for potential costs before any resources are deployed.	Planning: Helps model and compare the costs of different service configurations prior to creation.

▼ AWS Support Plan

One of the great things about AWS is no matter how big or small your business is, you are never alone. From small start-ups to large enterprises, private sector and public, all businesses have support options available that are designed to fit your specific needs.

Let's start off with our support plan options, leading off with our first plan: **Basic Support**. Every customer automatically gets Basic Support at **no cost** at all. Any customer can access support functions like **24/7 access** to customer service, documentation, whitepapers, support forums, AWS Trusted Advisor, and the **AWS Personal Health Dashboard**. This dashboard is a personalized view of the health of AWS services and any **alerts** when your **resources might be impacted**. These functions are available to everyone at no cost, but as you begin to move **critical workloads** into AWS, we offer higher levels of support to match your levels of need.

The next support plan is our **Developer Support** which includes everything from the Basic Support plan. Plus, you can also **email customer support directly** about any questions you have. You can expect a **response** within **24 hours** or less than **12 hours** if your **systems are impaired**. This is great for businesses that are experimenting or just **getting started**.

As customers begin to take **production workloads** live, we find they often choose **Business Support**. Everything from the previous plans is included, plus **Trusted Advisor** now opens up the **entire suite of checks** for your account. You are given **direct phone access** to our **AWS Support team**. They have a **4-hour response time** if your production **system is impaired** and a **1-hour response time** for when your production **system is down**. Additionally, as part of the Business Support plan, we provide **access** to an **infrastructure event management team**. This means that for an **extra fee**, we can help you **plan for massive events like brand new launches or global advertising blitzes**.

Finally, for companies running **mission-critical workloads**, we recommend the **Enterprise Support**. There are two types: **Enterprise On-Ramp** and **Enterprise**. These options have everything from the previous plans, plus a **reduced response time** for business and **mission-critical workloads** and resources such as **technical account managers**, or **TAMs**, to provide **guidance**. Your **TAM serves as your primary point of contact with AWS** and **proactively monitors your environment and assists with optimization**.

AWS Support looks at the customer holistically, not just if they have problems, but how we can help them be successful. There are even more teams and roles at AWS that are purpose-built to support your cloud journey.



AWS offers a range of support plans tailored to meet the needs of different customers, from those just getting started to large enterprises with complex requirements. Each plan builds onto the previous one, adding more advanced tools, personalized support, and faster response times to help you get the most out of your AWS experience. To learn more about which plan is right for you, refer to [Compare AWS Support Plans](#)(opens in a new tab).

AWS Support Plans comparison table

Basic Support	Developer Support	Business Support	Enterprise On-Ramp Support	Enterprise Support
Included for all AWS customers	Recommended for experimenting or testing in AWS	Recommended minimum tier for production workloads in AWS	Recommended for production and business critical workloads in AWS	Recommended for business critical and mission critical workloads in AWS
Includes access to documentation, whitepapers, and AWS re:Post	Response times: <ul style="list-style-type: none">• < 24 hours for general guidance• < 12 hours when systems impaired	Response times: <ul style="list-style-type: none">• <i>Includes previous plan response times</i>• < 4 hours when production system impaired• < 1 hour when production system is down	Response times: <ul style="list-style-type: none">• <i>Includes previous plan response times</i>• < 30 minutes when business-critical system is down	Response times: <ul style="list-style-type: none">• <i>Includes previous plan response times</i>• < 15 minutes when business- or mission-critical system is down
Core AWS Trusted Advisor checks	Core AWS Trusted Advisor checks	Full set of AWS Trusted Advisor checks	Full set of AWS Trusted Advisor checks	Full set of AWS Trusted Advisor checks and prioritized recommendations by AWS account team
Technical Account Management not included	Technical Account Management not included	Technical Account Management not included	A pool of technical account managers (TAMs) provide proactive guidance	A designated TAM provides consultative architectural and operational guidance

A **technical account manager (TAM)** is included with the Enterprise On-Ramp and Enterprise Support plans. The TAM serves as your primary AWS contact, offering expert guidance on using AWS services, optimizing architectures, managing costs, and connecting you with AWS programs and experts. For example, if you're building an app using multiple AWS services, your TAM can advise on the best integration approach.

Additional resources for your cloud journey

AWS re:Post

AWS re:Post is a community-driven, question-and-answer platform where users can seek help, share knowledge, and find solutions related to AWS services and technologies.

AWS re:Post also houses AWS Knowledge Center. AWS Knowledge Center contains articles and videos covering the most frequently asked questions and requests from AWS customers. To learn more about AWS re:Post, refer to [AWS re:Post.\(opens in a new tab\)](#). To learn more about AWS Knowledge Center, refer to [AWS Knowledge Center\(opens in a new tab\)](#).

AWS Trust and Safety Center

The AWS Trust and Safety Center provides information on how to report activity or content on AWS that you suspect is abusive. To learn more about this service, refer to [AWS Trust and Safety Center\(opens in a new tab\)](#).

AWS Solutions Architects

For Business and Enterprise Support customers, AWS solutions architects (SAs) provide architectural guidance, best practice recommendations, and help in designing scalable and secure applications.

AWS Professional Services

AWS Professional Services is a consulting service that offers deeper, project-based support. They help with complex migrations, security audits, performance tuning, and more. To learn more about this service, refer to [AWS Professional Services\(opens in a new tab\)](#).

AWS Professional Services offers deep technical expertise and can assist with security audits, best practices, and strategic guidance tailored to specific needs.

Self-support at AWS

AWS also provides extensive documentation and self-support resources that you can use to research, answer a question, or troubleshoot an issue. Documentation includes user guides for AWS services, SDK guides, blog posts, and whitepapers for specific solutions.

To learn more about these resources, refer to [AWS documentation\(opens in a new tab\)](#).

▼ AWS Marketplace and AWS Partners

In addition to the AWS Support options you've already covered, you also have the benefit of the AWS Marketplace. The **AWS Marketplace** is a curated digital catalog that you can use to **find, test, buy, deploy, and manage third-party software running in your AWS architecture**. This can help you **reduce the total cost of ownership and accelerate innovation by not spending development time recreating things that already exist**.

So, what are some of the key services included? Well, you'll see lots of **software** from independent software vendors, or **ISVs**, across a wide range of categories. Security, networking, storage, machine learning...you name it, it's probably there. There are both **free** and **paid** options and flexible pricing models, such as **pay-as-you-go** and annual **subscriptions**. The AWS Marketplace can reduce complexity in the procurement process, help you quickly deploy software, scale your software, and even help you **consolidate your billing in your AWS account**.

Another component of the AWS support world is the **AWS Partner Network** or **APN**. This is a **global program for technology and consulting businesses** who use AWS to build solutions and services for customers. As you can tell, AWS is super interested in your success.

With all these support options and resources dedicated to helping you succeed in the cloud, what will you build next? We can't wait to see.

AWS Marketplace

The AWS Marketplace is a digital catalog that includes thousands of software listings from independent software vendors. You can use AWS Marketplace to find, test, and buy software that runs on AWS. For each listing in AWS Marketplace, you can access detailed information on pricing options and reviews from other AWS customers. Solutions and services offered in the AWS Marketplace include the following:



- **Software as a service (SaaS):**
 - Business applications such as project management tools
 - Marketing tools such as customer engagement platforms
 - Collaboration tools such as file sharing services
- **Machine learning (ML) and AI:**
 - Prebuilt models for image recognition, natural language processing, and more
 - ML algorithms for training custom models
- **Data and analytics:**
 - Business intelligence platforms for visualization and reporting
 - Data integration tools

You can also explore software solutions by industry and use case. For example, suppose your company is in the healthcare industry. In AWS Marketplace, you can review use cases that software helps you to address, such as implementing solutions to protect patient records or using machine learning models to analyze a patient's medical history and predict possible health risks.

To explore more offerings and find the right fit for your business, refer to [AWS Marketplace\(opens in a new tab\)](#).

AWS Partner Network

The AWS Partner Network (APN) is a global community that uses AWS technologies, programs, expertise, and tools to build solutions and services for customers. Together, partners and AWS provide innovative solutions, solve technical challenges, and deliver customer value.

You can work with AWS Partners to create or use **specialized solutions** that are tailored to your unique business needs. For example, a retail company might use AWS to host their website. They could then work with an AWS Partner who specializes in advanced analytics and machine learning to improve customer personalization on that website. To learn more about working with AWS Partners, refer to [Engage with AWS Partners\(opens in a new tab\)](#).

You can also become an AWS Partner. There are many benefits to becoming a partner, including the following:

- **Funding benefits:** As businesses join the AWS Partner Network and participate in specific programs available to AWS Partners, they can unlock various **funding benefits** to help build, market, and sell with AWS. To learn more, refer to [AWS Partner Funding\(opens in a new tab\)](#).
- **AWS Partner events:** AWS Partner events include **webinars**, **virtual workshops**, and **in-person learning opportunities**. You can use AWS Partner events to **network** with other partners, learn more about new or current offerings, and collaborate with AWS experts. To learn more, refer to [AWS Partner Events\(opens in a new tab\)](#).
- **AWS Partner Training and Certification:** Take advantage of unique, partner-centered offerings from AWS Training and Certification. From certification to a specific service or learning objective, the AWS Partner training portfolio has numerous opportunities to upskill your cloud knowledge. To learn more, refer to [AWS Partner Training\(opens in a new tab\)](#).

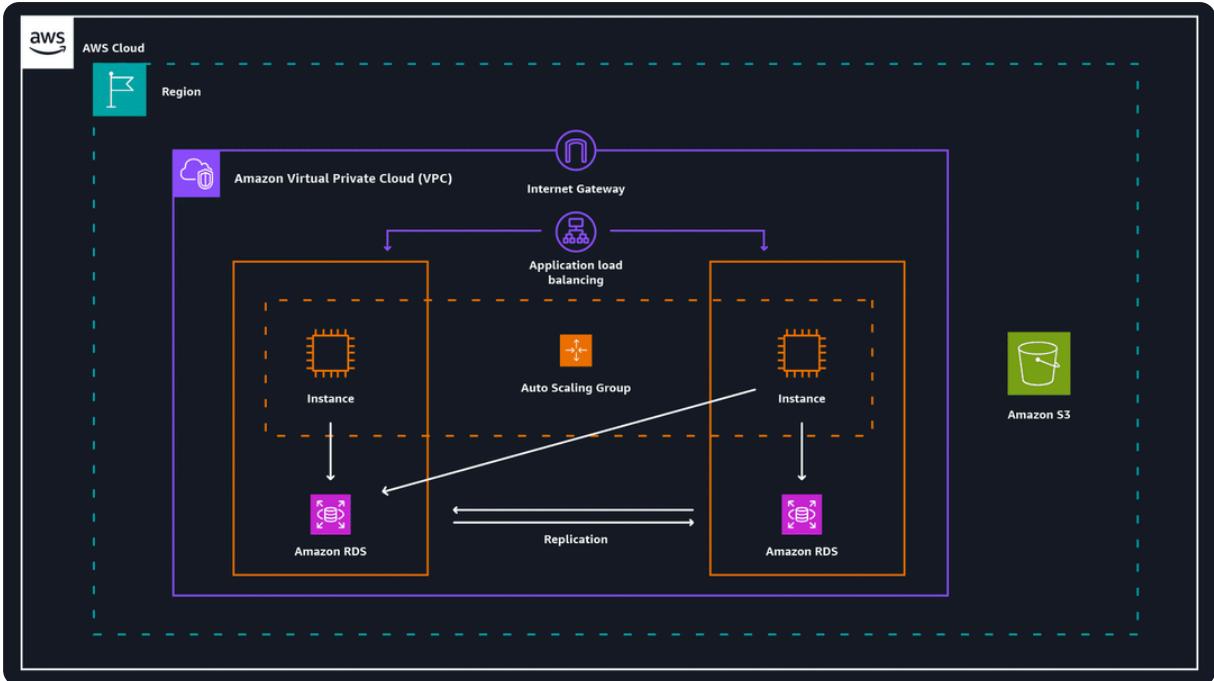
▼ Cost optimization

Rudy: What up, barista buds? I've got this architecture diagram I wanna talk through. We've got an app running on EC2 instances along with an RDS database in a VPC. There's also an S3 bucket. I figured the three of us could discuss some cost-optimization techniques using this diagram. But first, Alan, my man. Digging the jacket.

Alan: Oh, thank you. I cost-optimized my AWS account so much that I put some of my savings towards this lovely jacket.

Alan: Sparkly jackets aside, I like the idea of looking at **cost optimization across multiple services that are working together**. I mean, that really is how AWS usually works...multiple services working together to form a unique solution. So, let's start with **EC2** since that's often a place where people are looking to save.

Morgan: Good idea. I don't know about you two, but the first thing that comes to mind for me for cost optimization in **EC2** is **rightsizing**. Rightsizing is basically making sure that you are **analyzing and adjusting your resources to match the needs of your workload**. I heard of a customer just last week who saved a ton by using the service **AWS Compute Optimizer** to **identify overprovisioned instances**.



Rudy: Oh that, that's awesome. And Optimizer is like having a **cost-conscious** sidekick who's always got your back. It'll help you make sure you **aren't overprovisioning compute**.

Alan: Definitely. And don't forget about the option to use **Spot Instances**. I've seen companies cut their compute costs in half by using Spot Instances for their **noncritical workloads**.

Rudy: That's so cool! And it makes complete sense 'cause Spot Instances are available at up to a **90 percent discount** compared to **On-Demand Instance prices**. As a reminder, Spot Instances are unused EC2 capacity in the AWS Cloud. They are fantastic for **flexible workloads** that can be started and stopped easily.

Morgan: And, you know, **auto scaling** is another big factor of how customers can think about **cost** optimization. If you are manually managing your Amazon EC2 fleet, auto scaling can really save you time and money. Speaking of manual management, I always remind people to **clean up after themselves in your AWS account**. I'm sure you both have seen it, too. That sometimes, unused resources, **Amazon EBS volumes**, and **snapshots** hang around, and customers are **unaware** that these unused resources are **quietly draining the budget**.

Alan: That's a really good point. Now, what about **RDS**? Are there any thoughts there?

Rudy: Well, I like that you brought up **rightsizing**, Morgan. 'Cause it applies to **RDS** as well. No need to overprovision or underprovision. Just spin up RDS, **and it scales up and down** based on actual **demand**.

Morgan: True, and for **high-read workloads**, **read replicas** or **caching** can really help performance without breaking the bank.

Alan: Can you speak a little bit more about **read replicas** and caching can help somebody with cost optimization?

Morgan: Yeah, absolutely. So, I'm just thinking that RDS read replicas provide the capability to **scale read traffic horizontally**. This helps **spread** out your **read capacity** instead of unnecessarily **upgrading your primary instance** to a larger, more expensive instance. Or by storing frequently accessed data in a cache, like Amazon **ElastiCache**, you can also **reduce the load** on your primary instance.

Alan: That's a great callout. Now, Rudy, I know you mentioned that Amazon **S3** is a part of this architecture, too. When thinking about cost **efficiency** with S3, I always emphasize **storage classes**. **S3 Intelligent-Tiering** is good for **data** with unknown or **changing patterns**.

Rudy: Oh, definitely. And customers should also consider **compressing their data** to save on **S3 storage costs**, especially for **text-heavy data**. As objects are **uploaded** to **S3**, you can **configure a Lambda function to automatically compress them**. Even better, use **lifecycle policies** to **delete old versions** of objects. Actually, I had a customer who was paying for 10 years of daily backups when they only needed 30 days! Ten years compared to 30 days, seriously. You can imagine how happy they were when their AWS spend dropped like that.

Alan: Wow, that is impressive. Another thing I wanted to mention is that you can look into **architecting your applications to minimize inter-Availability Zone traffic and traffic to the internet**. It's important to remember **data transfer isn't always free**.

Morgan: Yeah, that's such a good point. Also, we didn't get to talk too much about **VPC endpoints** in our **networking section**, but that's very relevant to this conversation, and it's something to look into for **cost optimization**. **VPC endpoints** are a way to **privately connect your AWS VPC to supported AWS services without using the public internet**. Using **VPC endpoints** for Amazon **S3 access**, for example, can really **cut down on data transfer costs**, especially for **out-of-Region traffic**.

Rudy: It's just like traffic in real life: the less the better. Also, this is what teamwork does! Look, it's amazing how several small optimizations can add up to significant savings. And, most of these optimizations not only make things more cost-effective, but can also improve **performance** and **reliability**.

Morgan: Yeah, I couldn't agree more. It's all about finding that sweet spot between cost, performance, and operational efficiency.

Module 12 - Migrating to the AWS Cloud

▼ Introduction to Migration

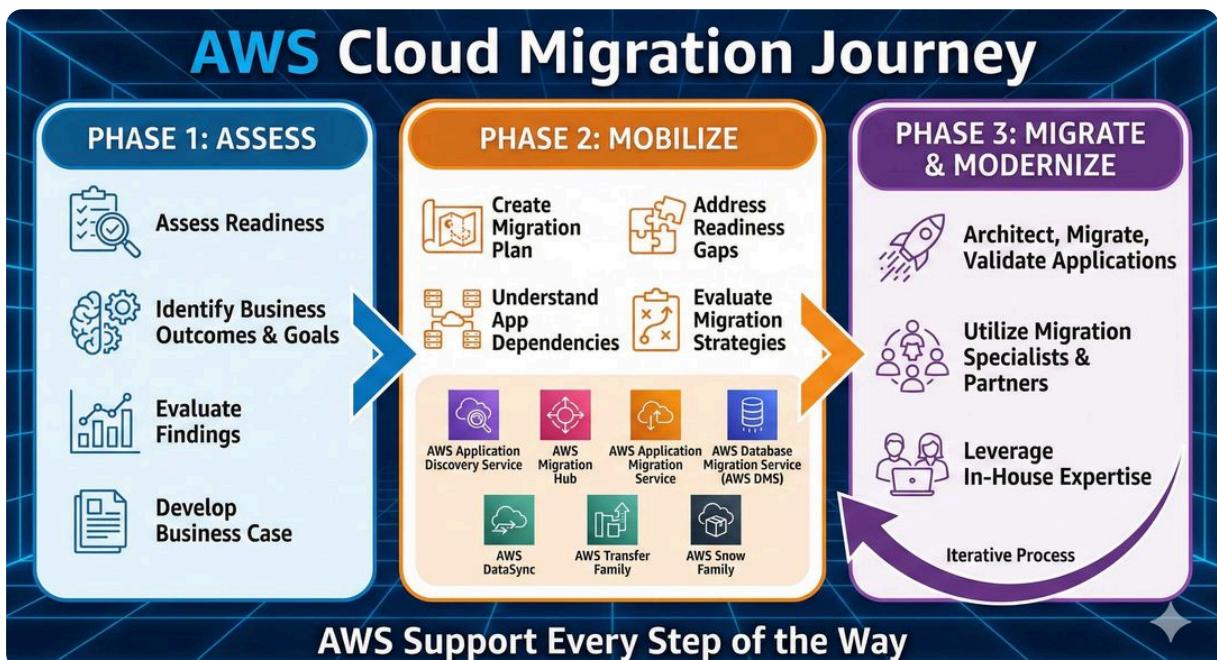
We've established how to get going on the AWS Cloud with your brand spanking new AWS account. But what if you have existing on-premises environments? Maybe you even already started your cloud journey with another cloud service provider?

That's all good. You can always migrate those workloads to AWS if you want to. We offer lots of tools, services, solutions, and architectures to **support** your **migration** journey. You can even use our **three-phase migration process** which provides resources and guidance for each individual phase.

First is the **assess phase**. You start with assessing your organization's **current readiness** for operating in the **cloud**. Then, you identify **business outcomes, goals, and the need for migration**. In this critical step, you evaluate your findings and develop the **business case for migration**.

Next is the **mobilize** phase. Here, you'll create a **migration plan** and address any of the gaps in your readiness. A strong plan starts with a deep **understanding of the interdependencies** between **applications** in your environments. You then evaluate **migration strategies** to meet your specific **business objectives**. Two services you might use in this phase are **AWS Application Discovery Service** and the **AWS Migration Hub**. Tools to support you include **AWS Application Migration Service** and **AWS Database Migration Service (AWS DMS)**. If you are transferring data, you might use **AWS DataSync**, **AWS Transfer Family**, and the **AWS Snow Family**.

The last phase, called **migrate and modernize**, is where you get things moving. Each **application is architected, migrated, and validated**. You can also take advantage of **migration specialists** and **competency partners** during this phase. Or, if you have the expertise in house, you can move things on your own. Just remember, AWS is always here to help you with this **iterative process** every step of the way.



▼ AWS Cloud Adoption Framework

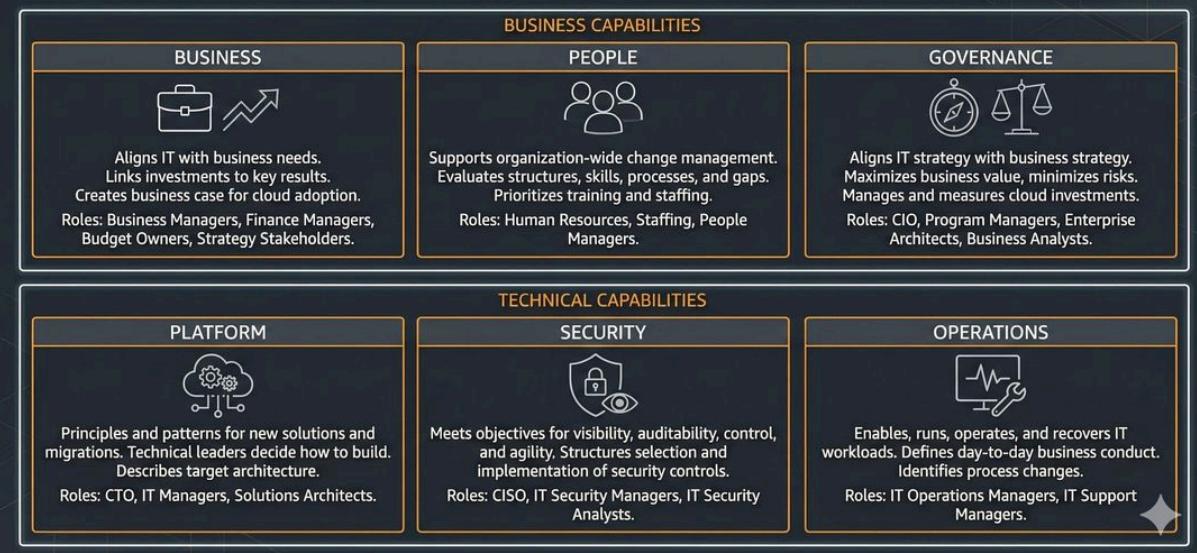
Luckily, a lot of knowledge about how to migrate to AWS has been captured and shared, making cloud migrations much easier for organizations. Depending on your role, you will approach cloud migrations differently. If you are a **developer**, your role and **viewpoint** will be much **different** than a **cloud architect, business analyst, or financial analyst**.

Different people bring **different perspectives** to the table for migration, and you want to harness those different perspectives and make sure everyone is on the same page. You also want to make sure that you have the right team to help support your migration. This can be a lot to keep track of, and someone new to the cloud might not think of all of the different roles who need to be involved.

The **AWS Professional Services team** has created something called the **AWS Cloud Adoption Framework, or AWS CAF**, to help you manage this process through **guidance**. **CAF** provides **advice** to your company to enable a quick and **smooth migration** to AWS.

The framework organizes guidance into **six areas** focused on the **different people** you will need to involve for your migration. Just like the different perspectives we talked about earlier, each one covers **distinct responsibilities** covered by **different groups**. They include **Business, People, and Governance**, which focus on the **business capabilities**, and then you have the **Platform, Security, and Operations**, which focus on the **technical capabilities**.

AWS Cloud Adoption Framework (CAF): Six Areas of Focus



Someone who is a business or finance analyst would fall under the Business perspective. HR would fall under the People perspective, and a cloud architect would fall under the Platform perspective. Each perspective is used to uncover **gaps in your skills and processes** which are then recorded as **inputs**. These **inputs** are used as the basis for creating what is called an **AWS CAF Action Plan**. You can then use the plan to **guide** your **organization's change management** as you journey to the cloud. Migrating to the cloud can be complicated, but again, you're not alone in this. There are tons of resources to help you get started, and the AWS CAF is a great place to begin.

AWS CAF

The AWS Cloud Adoption Framework (CAF) is a framework that brings **AWS** experience and **best practices** to companies preparing to migrate to the AWS Cloud. The framework provides tools to help accelerate the migration journey, organize resources, and align management **during the transition**.



Benefits: AWS CAF provides benefits for migrations to **reduce business risk** and improve sustainability and corporate transparency. Companies can grow revenue by creating new products and services in their cloud transformation. They can also reduce operational costs, increase productivity, and improve customer experience in their new cloud environment.

Use cases: You can use AWS CAF to migrate technology like **legacy infrastructure** and **applications**. You can also use it to migrate and optimize business processes, operations, and even create new business models with the move to the cloud.

Functional and business-related stakeholders

Business

The Business perspective makes sure that **IT aligns with business** needs and that **IT investments** link to **key business results**.

Use the Business perspective to create a strong **business case** for cloud adoption and **prioritize** cloud adoption initiatives. Make sure that your business strategies and goals align with your IT strategies and goals.

The following are common Business perspective roles:

- Business managers
- Finance managers
- Budget owners
- Strategy stakeholders

People

The People perspective supports development of an **organization-wide change management** strategy for **successful** cloud **adoption**.

Use the People perspective to evaluate organizational structures and roles, assess **new skill** and process requirements, and identify **gaps**. This helps **prioritize training, staffing, and organizational changes**.

The following are common People perspective roles:

- Human resources
- Staffing
- People managers

Governance

The Governance perspective focuses on skills and processes to **align IT strategy with business strategy**. This perspective helps you **maximize business value** and minimize risks.

Use it to understand how to update the staff skills and processes necessary to maintain business governance in the cloud. Manage and **measure cloud investments** to **evaluate business outcomes**.

The following are common Governance perspective roles:

- Chief information officer (CIO)
- Program managers
- Enterprise architects
- Business analysts

- Portfolio managers

Governance vs. Business: The Distinction

Perspective	Primary Focus	Role in Maximizing Value	Analogy
Business	Defining the Strategy and Making the Case.	Defines what success looks like and <i>why</i> the investment is being made.	The CEO: Sets the company goal: "We will increase market share by 20%."
Governance	Control, Alignment, and Risk Management.	Creates the systems to ensure IT investments actually deliver the defined business value while preventing expensive failures.	The Board of Directors/CFO: Creates policies and metrics to ensure the goal is met safely, legally, and within budget, minimizing risk of fraud or overspending.

Platform

The Platform perspective includes principles and patterns(microservices) for implementing new solutions in the cloud and migrating on-premises workloads to the cloud.

In simple terms, the Platform perspective is where the technical leaders decide **how to build everything** in the cloud.

Use a **variety of architectural models** to understand and communicate the structure of IT systems and their relationships. **Describe the architecture of the target state environment** in detail.

The following are common Platform perspective roles:

- Chief technology officer (CTO)
- IT managers
- Solutions architects

Security

The Security perspective makes sure that the organization meets security objectives for **visibility, auditability, control, and agility**.

Use AWS CAF to structure the selection and implementation of security controls that meet the organization's needs.

The following are common Security perspective roles:

- Chief information security officer (CISO)
- IT security managers
- IT security analysts

Operations

The Operations perspective helps you to enable, run, use, operate, and recover IT workloads to the level agreed upon with your business stakeholders.

Define how **day-to-day**, quarter-to-quarter, and year-to-year **business is conducted**. Align with and support the operations of the business. AWS CAF helps these **stakeholders define current operating procedures and identify the process changes and training needed to implement successful cloud adoption**.

The following are common Operations perspective roles:

- IT operations managers
- IT support managers

▼ Seven Migration Strategies

So, it's finally time to migrate to AWS. Every application will have seven possible options for cloud migration. We call these the seven Rs.

1. Rehost
2. Relocate
3. Replatform
4. Refactor
5. Repurchase
6. Retain
7. Retire

The first strategy is **rehost**. This is also called **lift and shift**, and it's convenient for businesses because they **don't need to make any changes**. At least, not at first. Just pick up the applications and move them pretty much as-is onto AWS. Usually this involves **turning existing servers into virtual machines or VMs**. You might not get all the possible benefits. But some companies find that even **without any optimization**, they can save up to **30 percent** of their total costs just by rehosting.

The next strategy is **relocate**. This could be if **applications are already VMs or containers** running on premises and it's **changing the hosting location to the cloud**.

The third option is called **replatform**, or **lift, tinker, and shift**. It's basically a lift and shift, but instead of a pure **one-to-one migration**, you might make a few cloud **optimizations**. But you're **not touching any core code in the process**. No new development efforts are involved here. For example, you might take your existing **MySQL database** and **replatform** it onto **Amazon RDS without any code changes** at all. Or you might even consider **upgrading** to **Amazon Aurora**. This can give significant benefits to your database team and provides improved performance.

Number four: **refactor** or sometimes called **rearchitecting**. Now, you're **writing new code**. This is driven by a strong business need to **add features or performance** that might **not be possible on premises** but now are within your reach. **Dramatic changes** to your **architecture** can be very beneficial to your enterprise, but this will come at the **highest initial cost** in terms of planning and human effort.

Number five is **repurchase or drop and shop**. This is a common strategy for companies that want to **change software vendors** and get a **fresh start** as part of their migration. For example, ending your licensing with a database vendor in favor of cloud-based database offerings. Now, this sounds great, but remember you will have to deal with a **new software package**. Some are straightforward to implement, but others take time. The total **upfront expense** of this strategy might **increase**, but the **potential benefits** can be substantial. Repurchasing involves moving from a traditional license to a software-as-a-service (SaaS) model.

The last two strategies, **retain** and **retire**, don't actually involve migrating anything. Let's look at retain first. Retain can also be called **stays where it lays**. This is when some applications are about to be deprecated but maybe not just yet. They still **need to run for another couple of months**. You can migrate these applications to AWS, but why? You should only migrate the applications that **make sense for your business**. And then as time goes on, these applications can be deprecated where they live and, **ultimately, retired**.

And speaking of **retiring**, that's our seventh strategy. It's not unusual for companies to find that more than 10 percent of their workloads in an enterprise IT portfolio are **no longer being used**. Using the AWS migration plan as an opportunity to **end-of-life** these applications can save significant cost and effort for your team. Sometimes, you just need to turn off the lights. And that's it: the seven Rs of Migration.

The Seven R's of Cloud Migration

Strategy (The R)	Simple Action	Description	Real-World Example
Rehost	Lift and Shift	Moving an application as-is from an on-premises server to an EC2 instance. No code changes are made.	Using AWS Application Migration Service (AWS MGN) to move a legacy Windows CRM application directly onto an EC2 VM to quickly exit a data center.
Relocate	Hypervisor Shift	Moving virtual machines or containers from an on-premises virtualization platform to a compatible managed service in AWS.	Moving a fleet of virtual servers running on VMware vSphere in a company data center to VMware Cloud on AWS (VMC) , maintaining all existing operational tools.
Replatform	Lift, Tinker, and Shift	Moving an application but making minor cloud optimizations to gain some benefits without	Migrating an on-premises MySQL database to Amazon RDS for MySQL . The application code stays the same, but the operations team no longer manages

		touching core code.	patching, backups, or high availability.
Refactor	Rearchitect	Writing new code and fundamentally changing the application's architecture to fully utilize cloud-native features.	Breaking down a monolithic application into separate microservices using containers (Amazon ECS/EKS) or serverless functions (AWS Lambda) to enable agility and scale.
Repurchase	Drop and Shop	Ending a license for an existing software application and replacing it with a new cloud-native SaaS solution.	Replacing an outdated, on-premises Exchange email server with a subscription to a cloud-based service like Microsoft 365 or Google Workspace .
Retain	Stay Where It Lays	Deciding not to migrate an application because it is not strategic, will be decommissioned soon, or requires a critical hardware dependency that cannot move.	An application is scheduled to be retired in six months after a new system is fully launched, so the business decides to leave it on-premises until the final sunset date.
Retire	Turn Off	Decommissioning applications that are no longer being used or are redundant.	Discovering that 15% of the IT portfolio consists of unused "zombie" servers and applications from old projects, which are then permanently shut down to realize immediate cost savings.

▼ Migration Services and Tools

Let's explore four services that help businesses move their applications and data to the cloud.

Application Discovery Service

The Application Discovery Service discovers **on-premises server inventory and connections**. It gathers **configuration, performance, and connection details** for both servers and databases to create a **detailed migration plan**.



Benefits: With Application Discovery Service, you get a comprehensive snapshot of your on-premises inventory. You also can integrate discovery data with other services like Migration Hub and protect the data Application Discovery Service collects.

Use cases: You can use the AWS Application Discovery Service to conduct discovery and inventory, map the connections and dependencies, and generate a migration plan.

Think of this as your migration **detective**. It explores your current IT setup, taking notes on what programs you use, how they work together, and what resources they need. This helps you plan your move to AWS more effectively by performing the automatic discovery of applications and providing detailed reports on your IT landscape. You can use this to make informed decisions about what you want to move and how to do it, saving time and reducing risks during migration.

Application Migration Service

Application Migration Service is a tool to move and improve your on-premises and cloud-based applications. It helps customers streamline, expedite, and reduce the cost of migrating and modernizing applications.



Benefits: The benefits include support to migrate from any source infrastructure that runs a supported operating system (OS). It makes it possible to modernize your applications during migration. You can maintain normal business operations during the application replication process and also reduce costs by using one tool for a wide range of applications.

Use cases: You can use the Application Migration Service for **on-premises** applications running on **physical servers** or infrastructure, **cloud-based** applications, or moving **between AWS Regions**. You can also use it to modernize applications.

Next is the **AWS Application Migration Service**. This is like having a team of **expert movers**. It helps you pack up your applications and smoothly transport them to AWS. It can even make **small adjustments** to ensure everything works in its **new cloud home**. Benefits include **automated migration** and **minimal downtime** during the move. You can use this to **quickly transfer** their applications to the cloud without disrupting business operations.

Migration Evaluator

The Migration Evaluator is a migration assessment service that helps you create a business case for AWS Cloud planning and migration. It does this with a **data driven approach**, analyzing your current state, target state, and developing a **migration readiness** plan with projected cloud costs.



Benefits: The benefits include removing the guesswork when migrating. It provides visibility into **multiple cost-effective** cloud migration **scenarios**. It also gives insights on **reusing existing software licensing**, which can further reduce costs.

Use cases: You can use Migration Evaluator to conduct broad-based discovery, take a snapshot of your current on-premises footprint to fine-tune licensing, view **server dependencies**, and gain visibility into multiple migration scenarios. You can also use it to estimate and reduce your cloud costs.

Then there is the **Migration Evaluator**. Imagine a **moving consultant** who helps you **estimate the cost of your relocation**. This tool **analyzes your current IT setup** and provides **detailed cost estimates for moving to AWS**. It helps you understand **potential savings** and **plan your budget**. Businesses can use this to make **data-driven decisions** about their cloud migration strategy and **justify the investment to stakeholders**.

Migration Hub

The Migration Hub is a centralized hub to take you from discovery, assessment, planning, and execution of your migration. It provides **tools**, **guidance**, and **automated recommendations** to collaborate with your team and **track your migration**.



Benefits: With the Migration Hub, you have one location to go for your migration and, expert guidance in the form of prescriptive journey **templates**. Another benefit is cost savings because there is **no charge to use Migration Hub**.

Use cases: You can use Migration Hub for migration **assessment** and **planning** and migration **completion** and **collaboration** with teams. You can also use it for **modernization efforts** like **fast-tracking application refactoring**.

Finally, there is the **AWS Migration Hub**. This is your **command center for the entire moving process**. It gives you a **unified view of all tasks and progress tracking**. You can use this to keep your migration efforts organized and ensure nothing falls through the cracks. Working together, these AWS migration tools provide several benefits: **streamlined migration process, reduced risk of errors or downtime, cost savings through efficient planning, and faster time to value in the cloud**.

By using these tools, businesses can make their journey to AWS smoother, faster, and more cost-effective. And that in turn, helps businesses start enjoying the benefits of the cloud sooner.

AWS Migration and Modernization Competency Partners

When migrating to the AWS Cloud, if you need help and expertise, you don't have to go it alone. You can work with the AWS Competency Partner Program. You can search for AWS Migration and Modernization Service Partners who specialize in specific phases of the migration or the type of help you need. To learn more, refer to [AWS Migration and Modernization Competency Partners](#)(opens in a new tab).

AWS Migration Tools and Services

Service Name	Primary Focus / Analogy	Function in Migration Lifecycle	Key Benefit / Output

Application Discovery Service (ADS)	The Migration Detective (Inventory & Mapping)	Discovery & Planning: Gathers configuration, performance, and dependency details for on-premises servers and databases.	Comprehensive Snapshot of on-premises inventory; generates a detailed migration plan .
Migration Evaluator	The Moving Consultant (Cost Assessment & Business Case)	Assessment: Analyzes the data collected by ADS to create a data-driven business case for AWS adoption.	Provides projected cloud costs and visibility into multiple cost-effective scenarios, including licensing reuse insights.
Migration Hub	The Command Center (Orchestration & Tracking)	Centralized Control: Provides a unified, single location to track the progress of the entire migration (discovery, assessment, planning, and execution).	Offers prescriptive journey templates and enables collaboration while providing a single view of migration status .
Application Migration Service (AWS MGN)	The Expert Movers (Execution & Replication)	Execution: Automates the block-level replication of on-premises applications and servers to AWS EC2 with minimal downtime.	Streamlined, Expedited Migration with support for modernizing applications during the process, maintaining normal business operations.

▼ Database Migrations

Companies choose to migrate their databases to AWS for a variety of reasons like cost optimization or modernization. No matter the motivation for your migration, **AWS Database Migration Service**, or **AWS DMS**, is a really useful service. DMS is a migration service to move your database to AWS. You can use it to migrate **relational databases**, **data warehouses**, **NoSQL databases**, and even **analytics workloads**.

AWS DMS

The AWS Database Migration Service (**AWS DMS**) makes it possible to quickly and securely migrate databases and perform ongoing data replication tasks for **live databases** and data warehouses. It provides a way to **plan, assess, convert,** and migrate databases even with data warehouses in one central tool.



Benefits: AWS DMS provides benefits for migrating databases including **maintaining high availability and low downtime during the migration process.** It supports **homogenous** and **heterogenous migrations.** It also makes it possible to migrate **terabyte sized databases at a low cost.**

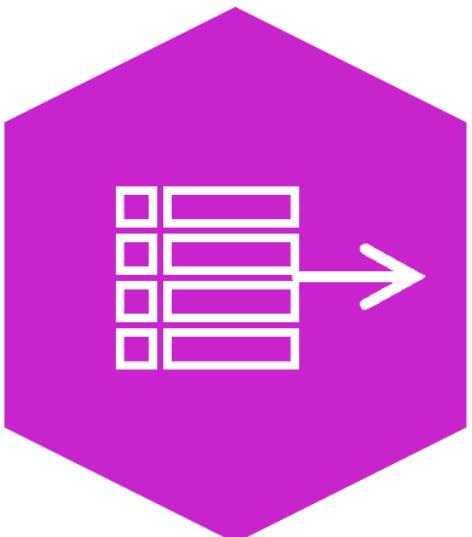
Use cases: You can use AWS DMS to move to managed databases, **remove licensing costs,** **replicate ongoing changes** in your database, and improve integration with data lakes.

At the most basic level, **DMS** is a **virtual machine that runs replication software.** You configure a **source** and **target database** to tell DMS where to **extract data** from and where to **load it.** Then, you **schedule a task that runs on this server to move your data.** When the migration starts, **data is migrated** while your original **database stays live.** AWS DMS handles all the complexity. You even have the **option to fall back to your original database.** Or you can **replicate to other databases** in different AWS Regions or Availability Zones.

Now, what if you need to change from **one type of database to another?** For example, maybe you want to move your legacy commercial database to a different type of engine like **Amazon Aurora.** That's called a **heterogenous conversion**, and it requires a few extra steps. For this, you'd need to figure out the **schema**, or the **structure and organization of the data**, and **where everything will move to** in the new **database.**

AWS SCT

AWS SCT makes it possible to convert database schemas and code objects (like **stored procedures, views, and functions**) from one database **engine** to another. AWS SCT can even give you **estimates** of how big of an **effort** a conversion is, which helps with **planning.**



Benefits: AWS SCT provides benefits to simplify database migrations by **automating schema analysis, recommendations, and conversion at scale**. It is compatible with popular databases and analytics services as source and target engines. It can save weeks or months of manual time and resources, which are typically required in conversions.

Use cases: You can use AWS SCT to move from **commercial databases to open source databases**. You can also use it for migrating **large data warehouse workloads** and **modernizing or updating database schemas** in place.

AWS Schema Conversion Tool, or **AWS SCT**, can be a huge help with these types of conversations. SCT converts the **source database schema and code** into a **format compatible with the target database**. Any code that can't be converted automatically is **marked for manual conversion** for you to review, and then the migration is ready to go.

These two AWS database migration services provide several benefits. They **save you time**. What might take weeks or months to do manually can be done much quicker. They provide a **cost-effective** solution and help reduce the effort of moving to new database systems. They give you flexibility so you can conveniently **explore** and **adopt different database options** that best suit your needs. And they're secure, utilizing **AWS security best practices** during your migration to cloud.

So, whether you need to migrate to the same type of database or to one of the dozen different AWS databases, there are migration tools to help you make your move successful.

▼ Transferring Data Online

Several AWS services facilitate online data transfer to the AWS Cloud. In the last lesson, you learned that AWS Database Migration Service (**AWS DMS**) transfers the database and its data to the AWS Cloud. In this lesson, you will identify services that can help with the **considerations of online transfer** for other types of data and files.

In this lesson, you will review three services:

- **AWS DataSync**
- **AWS Transfer Family**
- **AWS Direct Connect**

When you migrate data to the AWS Cloud, there are a few considerations to keep in mind. You need to ensure **security** (will it get there safely), data **validation** (will it get there in **one piece**), **scheduling** (when is the best time). You would also confirm **bandwidth requirements**. For the majority of data migration workloads, AWS **DataSync** will do the job.

AWS DataSync

AWS DataSync is specifically designed for **automating** and **accelerating** data **transfer**. DataSync simplifies and accelerates moving large amounts of data between **on-premises** storage and **AWS storage** services like Amazon Simple Storage Service (Amazon S3). It **automates** many aspects of the transfer process, including **running instances**, **encryption**, and **network optimization**.

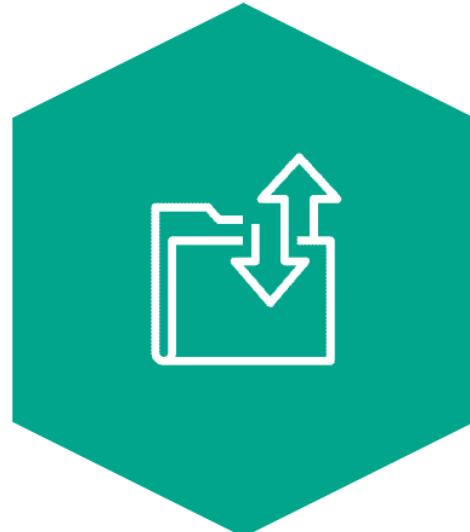


Benefits: The benefits include streamlining and accelerating secure data migrations. DataSync manages data movement workloads with **bandwidth throttling**, migration **scheduling**, **task filtering**, and **task reporting**. It also provides **rapid data replication**.

Use cases: You can use DataSync to migrate your data, **archive** your **cold data**, and manage **hybrid data workflows**.

AWS Transfer Family

The AWS Transfer Family makes it possible to seamlessly manage and share data with simple, secure, and scalable file transfers. This service provides fully managed support for secure file transfers over FTP, **Secure File Transfer Protocol** (SFTP), **File Transfer Protocol Secure** (FTPS), and other protocols. It helps you transfer files directly into and out of AWS storage services like Amazon **S3** and Amazon **EFS**.



Benefits: The benefits include **simplifying** the process of **setting up and managing file transfers** and reducing the need for complex infrastructure management. The Transfer Family provides **secure data transfer** with encryption and **authentication**, to ensure data **integrity** and **confidentiality**. It is built to **scale** and streamline workflows.

Use cases: You can use the Transfer Family to **modernize** and **manage your file transfers**, **simplify data sharing** with your **workforce** and **partners**, and **integrate transactional business data** into a unified data lake.

AWS DataSync vs Transfer Family

Feature	AWS DataSync	AWS Transfer Family
Primary Goal	Automated Migration & Sync	Managed File Transfer (B2B)
Protocols	NFS, SMB, HDFS, S3, Azure Files, Google Cloud Storage	SFTP, FTPS, FTP, AS2
Connectivity	Requires an Agent installed on-premises	Provides a Server Endpoint (no agent needed)
Target Storage	S3, EFS, FSx (all types)	S3, EFS
Use Case	Migrating 100TB of data; nightly backups	Third-party vendor uploading weekly invoices
Speed	Optimized for speed (up to 10x faster than rsync)	Limited by the standard protocol used
Pricing	Per-GB transferred	Hourly server charge + Per-GB transferred

You might remember Direct Connect from the networking module. With the bandwidth of a dedicated connection, it is also a great solution for moving your data online to the AWS Cloud when migrating.

Direct Connect

AWS Direct Connect is a service that makes it possible for you to establish a **dedicated private connection between your network and virtual private cloud (VPC)** in the AWS Cloud. Because it is your dedicated connection, it is a **fast, reliable, and secure** way to transfer your data or files.



Benefits: Direct Connect helps **reduce network costs** and increase amount of bandwidth.

▼ Transferring Data Offline

Alternatives for data migration

Transferring data offline

Many customers prefer online migrations, but there are some customers who need to transfer data offline. An example would be if **bandwidth is limited**, or in remote locations with no internet and Direct Connect is not an option. Or, in cases with **large data volumes**, sending **petabytes of data over the internet** would take **longer** than simply sending a **physical device**. In the following section, you will review **AWS Snowball Edge Storage Optimized devices**.

Snowball Edge Storage Optimized devices

AWS Snowball Edge Storage Optimized devices are a great solution for offline data migration where connecting to the internet might not be an option. These devices deliver **high performance NVMe storage**, making it possible to simplify **multi-petabyte data** migrations from **on-premises locations to AWS**.



Benefits: The benefits include delivering better compute performance and larger storage capacity with gigabytes of data per second for data migration workloads with offline requirements.

Use cases: You can use Snowball Edge devices for data migration when offline migration is required. They can also be used for **edge computing** when a **secure, rugged** device is needed.

Module 13 - Well-Architected Solutions

▼ Introduction to Well-Architected Solutions

Well, well, well, it's good to see you're still here. We're nearing the end of our time together, but with just a few topics left we aren't done yet!

There are actually hundreds of AWS services that we could cover, but we're just going to discuss a few more that can be used for development, business applications, and end user computing.

By now, you're also familiar with how to use AWS services as building blocks for your cloud solutions. With the flexibility AWS provides, you can create virtually any architecture to solve the problem at hand. Whether basic or complex, the options are endless. But here's the key question: how do you know if the architecture you've built is actually good? In other words, how can you ensure that your solution is both effective and optimized for the real world?

Luckily, AWS provides tools to help **evaluate** the quality of your **designs**. One of the most important tools at your disposal is the **Well-Architected Framework**. The Well-Architected Framework helps you evaluate the architectures you build, ensuring that they are aligned with **best practices** in key areas.

Alright, ready to wrap up this party with a few more services and some help on how to architect efficiently? Let's get going.

▼ AWS Specialized Services

As you know, each AWS service is purpose-built for specific use cases, often times based on feedback or requests from customers. Let's look at a small sample size by considering some services for **development**, **business applications**, and **end-user computing**.

Starting with services for **development**, let's think about one that's purpose built for continuous **integration and continuous delivery**, or **CI/CD**, pipelines. **CI/CD pipelines** help developers continuously release code automatically as changes are committed to a source code repository.

AWS CodePipeline is a service that **automates** your **release pipeline**. It can **monitor** your **repository** for **changes**, and then it can kick off a series of steps, like **building**, **testing**, and **deploying the code**. Essentially, it helps developers set up a **workflow** that takes their code from **development to production** smoothly.

Another service to be aware of for developers is **AWS X-Ray**. This service helps you **monitor** and **debug** your **applications**, giving you **insights** into how they're **performing**. When an application isn't behaving as expected, X-Ray helps you **visually pinpoint the issue** by **tracking requests** as they **move through your system**, so you can fix problems faster.

Then there's **AWS AppSync**, which is a service that streamlines building **GraphQL APIs**. GraphQL is a specific type of **API** that has gained **popularity** in **web** and **mobile development** in recent years. **AppSync** helps you more easily **connect frontend apps with backend data**.

Next up; **AWS Amplify**, which makes it more convenient to **develop**, **deploy**, and **manage applications** on **AWS**. Whether you're adding authentication, data storage, or hosting, Amplify handles a lot of the complexity for you, so you can focus on creating a great **user experience**.

Now, let's look at some **business application services**. One service in this category is **Amazon Connect**. It's an **AI-powered**, **cloud-based contact center** service that makes it more efficient to set up and manage **customer service operations**. Whether you're answering calls, handling chats, or running a **support center**, by using **Amazon Connect**, you can design solutions that ensure smooth **communication with customers**.

Another helpful tool is **Amazon Simple Email Service**, or **SES**. If your business needs to **send large volumes of emails**—whether it's **newsletters**, **promotional messages**, or **transactional emails**—SES is a great choice.

So, those were a few specialized services for business use cases. But let's shift gears a bit and talk about **end-user computing**. One of the functions of IT in modern businesses is setting up and configuring services that make it easier to provide **remote access** to resources like **virtual desktops and applications**.

Amazon AppStream 2.0 makes it possible for you to **stream desktop applications** to **users** on any device. Instead of installing software locally, users can simply access the **applications** they need **over the web**.

Then, there's **Amazon WorkSpaces**, a fully managed virtual desktop infrastructure service that users can access from anywhere.

For a more **lightweight solution**, there's **Amazon WorkSpaces Web -now called WorkSpace Secure Browser**, which is designed for people who only need web-based applications.

Okay, wow! That's a lot of different perspectives and services, and it's not even the whole list. But as you can see, of the hundreds of services and tools from AWS, each one is **purpose-built** and tailored to specific use cases.

Types of services

AWS services are purpose-built for specific use cases. In the following section, you will learn more about the following four types of specialized AWS services:

- Development services
- Business application services
- End-user computing services
- IoT services

Development services

AWS offers several services to help developers automate CI/CD pipelines, monitor and debug applications, build GraphQL APIs, and deploy web and mobile applications on AWS. Let's examine these services in a bit more detail.

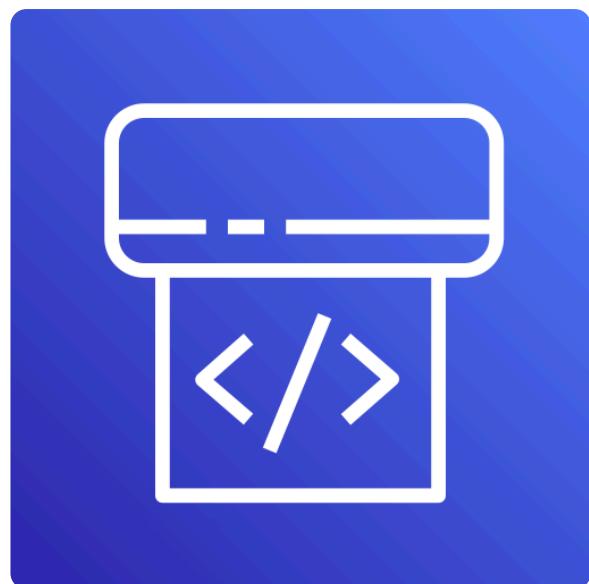
AWS CodeBuild

CodeBuild is a fully managed continuous integration service that compiles source code, runs tests, and produces software packages for deployment. It automatically scales to meet demand, and you only pay for the build time that you use.



AWS CodePipeline

CodePipeline is a **fully managed CI/CD** service that **automates** the **build**, **test**, and **deploy** phases of your **release process**. This helps developers streamline **software release workflows** and **reliably deliver new features** and fixes without needing to provision servers.



- **CI/CD pipelines** automate the *integration, testing, and deployment of code changes to help provide quick and reliable software delivery.*
- **CodePipeline** is the **orchestrator** (the **workflow manager**), and **CodeBuild** is the **executor** (the engine) for one specific step (the build).

AWS X-Ray

X-Ray is a powerful **tracing**, **debugging**, and **performance analysis** tool that helps developers **visualize application behavior**. With X-Ray, developers can quickly identify performance **bottlenecks**, **troubleshoot issues**, and **optimize applications** for better efficiency and reliability.



AWS AppSync

AWS AppSync is a **fully managed GraphQL** service. With AWS AppSync, developers can create a **single GraphQL API** that can securely access, manipulate, and combine data from multiple data sources. This helps developers connect frontend applications with backend data.



- *With **GraphQL**, a query language for APIs, clients request only the specific data they need.*

AWS Amplify

Amplify helps you streamline the process of **developing, deploying, and managing secure** and **scalable full-stack applications** on AWS. With Amplify, developers can quickly **add features**, like **authentication, APIs, storage, and hosting**, with **minimal infrastructure management**.



- ***Full-stack applications** are software systems that involve both frontend (user interface) and backend (server-side) development.*

Business application services

These services are ideal for managing business application needs such as **customer service operations** and **email promotions**. Let's review a couple of examples.

Amazon Connect

Businesses can use this **AI-powered contact center** service to efficiently **set up** and **operate a scalable customer service call center**. Amazon Connect provides capabilities for **call routing**, **recording**, and **analytics** while **integrating** seamlessly with **other AWS services**.



Amazon Simple Email Service (Amazon SES)

Amazon SES is a **scalable** and **cost-effective** **email** service provider that can be integrated into any application for **reliable, high-volume email automation**. It helps businesses optimize the **delivery** of **transactional** and **marketing emails**, resulting in enhanced **customer engagement**.

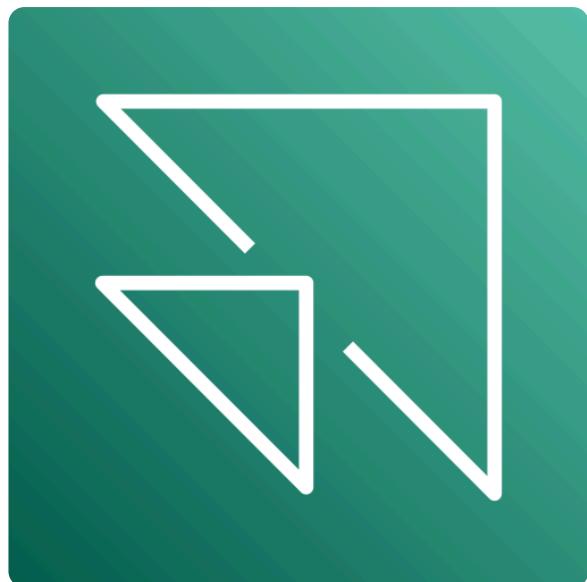


End-user computing services

In modern businesses, IT departments often need to provide **remote access** to resources like **virtual desktops** and **applications**. Let's explore some AWS services that can be used to set up these environments for employees.

Amazon AppStream 2.0

AppStream 2.0 is a **fully managed** service that **streams applications** from the **cloud** directly to any **compatible device**. This includes **software-as-a-service (SaaS)** applications and applications **converted** from **desktop** to **SaaS without code revisions**. This provides instant access to powerful software without the need for high-end local hardware.



- In **SaaS**, applications are hosted on the cloud and accessed through the internet, without the need for local installation or maintenance.

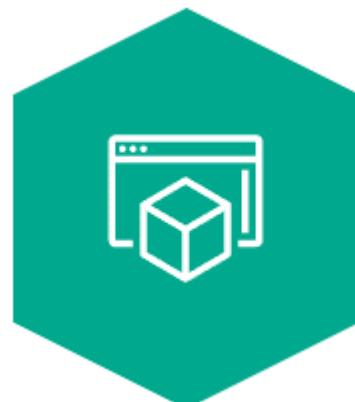
Amazon WorkSpaces

WorkSpaces is a **fully managed cloud-based desktop computing** service. With WorkSpaces, **employees** can securely **access** their **work environment** from any **device** with an **internet connection**. Employees can perform the same tasks as if they were on a physical office computer, while companies can benefit from cost-efficiency and easy administration.



Amazon WorkSpaces Secure Browser (formerly Amazon WorkSpaces Web)

WorkSpaces Secure Browser is a **fully managed remote enterprise browser**. It provides a protected environment for employees to **access private websites, SaaS applications**, and the **public internet**. With WorkSpaces Secure Browser, IT departments **don't need** to manage specialized **client software, infrastructure, or VPN connections**.



IoT services

Internet of Things (IoT) is a network of connected physical devices embedded with sensors and software that collect and exchange data over the internet. These devices can be monitored and controlled remotely to improve efficiency, provide new services, and enhance quality of life.

AWS IoT Core

AWS IoT Core is a managed cloud service used to securely **connect physical devices** with **cloud applications**. It helps you create **efficient IoT solutions** by streamlining the complex process of **ingesting, processing, and acting on device data**. Device **connections** and data are secured with **mutual authentication** and **end-to-end encryption**, and you can choose from several **communication protocols**.



Some IoT solutions include the following:

- **Smart security cameras** – Home monitoring that sends alerts to your phone
- **Smart pet feeders** – A pet feeder that you can control remotely
- **Smart irrigation systems** – A rain machine that adjusts watering based on weather and soil conditions

AWS Cloud Specialized Services Overview

Service Name	Category / Focus	Primary Function	Key Benefit / Distinction
AWS CodePipeline	CI/CD Automation	Fully managed service that automates and orchestrates the entire release workflow (build, test, deploy).	It is the orchestrator that streamlines the software release process without managing servers.
AWS CodeBuild	CI/CD Automation	Compiles source code, runs tests, and produces deployable software packages.	It is the executor that automatically scales to run the build task, and you only pay for the time used.
AWS X-Ray	Monitoring & Debugging	Provides tracing, debugging, and performance analysis across distributed applications.	Helps developers quickly identify performance bottlenecks and optimize application behavior.
AWS AppSync	API Development	Fully managed GraphQL service for	Securely accesses, manipulates, and combines data from

		creating a single API endpoint.	multiple data sources in one query.
AWS Amplify	Application Development	Streamlines the process of developing, deploying, and managing secure, scalable full-stack applications.	Quick way to add features like authentication, storage, and hosting with minimal infrastructure management .
Amazon Connect	Business Application	AI-powered contact center service that is easily set up and scaled.	Provides call routing, recording, analytics, and seamless integration for customer service operations .
Amazon Simple Email Service (SES)	Business Application	Scalable and cost-effective email service provider for high-volume automation.	Optimizes the delivery of transactional and marketing emails to enhance customer engagement .
Amazon AppStream 2.0	End-User Computing	Fully managed service that streams applications from the cloud to any compatible device.	Provides instant access to powerful software (including converted SaaS) without the need for high-end local hardware .
Amazon WorkSpaces	End-User Computing	Fully managed cloud-based desktop computing service .	Provides employees with a secure, managed virtual work environment accessible from any device.
WorkSpaces Secure Browser	End-User Computing	Fully managed remote enterprise browser for employees.	Provides a protected environment to access private websites and SaaS apps without managing VPNs or specialized client software .
AWS IoT Core	IoT Services	Managed cloud service to securely connect physical devices with cloud applications.	Streamlines the complex process of ingesting, processing, and acting on device data (e.g., smart cameras, feeders, irrigation).

▼ AWS Well-Architected Framework

The AWS Well-Architected Framework helps organizations build more secure, high-performing, resilient, and efficient infrastructure for their applications. It's composed of **six pillars**, which can be applied to any workload. These six pillars are **Operational Excellence, Security, Reliability, Performance Efficiency, Cost Optimization, and Sustainability**. As architects, we gotta know those off by heart. But this framework provides a **consistent approach for designing architectures**.

The first pillar is **Operational Excellence**. It focuses on **running and monitoring systems** to deliver **business value** while **continuously improving processes**. This includes things like **automating deployments** through **pipelines** and **responding to events** effectively to ensure smooth operations.

The second pillar is **Security**. Don't let bad actors in and keep everything locked down. This is where this pillar comes in. It helps to make sure you are **building security into your solutions** using **best practices**. Additionally, it shows you how to maintain **data integrity, protect systems**, and adhere to the principle of **least privilege access**.

The third pillar in the framework is **Reliability**. This pillar is centered on **recovery planning** and making sure systems can **withstand failures**. This includes **strategies** for recovering from **disruptions**. The other part of the pillar pertains to **adapting** systems to meet **evolving business** and customer **demands**.

The next pillar is **Performance Efficiency**. It focuses on using **resources efficiently**, like with **rightsizing** your **EC2 instance** based on workload and memory requirements. More so, this pillar promotes making **informed decisions** to ensure efficiency **continues**, even as **business needs change**.

The fifth pillar is **Cost Optimization**. As you can guess, it focuses on **controlling and reducing expenditures** by **optimizing resource allocation**. For example, you provision a specific EC2 instance to start out. And then, as you gain more information on your workload, you realize the instance is being underutilized. Are you stuck with it? No. You can switch to an instance with lower specs, which might actually be more cost-effective. Moreover, if you don't need a service, **deprovision** it. Eliminate the cost.

The sixth and final pillar is **Sustainability**. It emphasizes designing **energy-efficient systems** and minimizing environmental impact. This pillar encourages using the most appropriate AWS infrastructure to **reduce** the need for **always-on resources**. For example, if you don't need an **always-on EC2 instance**, consider switching to **AWS Lambda**. Or maybe using a smaller Amazon RDS instance if you don't need that extra storage. Not only are you reducing cost, but you can **lower energy consumption** and **carbon emissions**.

Now that you have an understanding of the pillars, how do you actually use the framework? Well, here's how: It's a **self-service tool!** Create a **workload**, run the tool against your AWS account, and a **report** is generated that shows areas that should be addressed. Apart from the tool showing you where it has detected potential issues, it **shows you how to remedy them**. These remediations are determined using **established best practices**. But it should be noted that you can always override these settings if the questions don't apply to your specific scenario. It's very **customizable**, so don't stress. Take the **advice** and do the best remediation possible.

You can see the workload name, pillars, and drop-down menus with questions for each one. You can also change the toggle to indicate whether the question is applicable or not. Lastly, there are resources to help you understand how to answer each question.

Anyhow, that's the Well-Architected Framework and tool. Hope you have enjoyed learning how to evaluate your workloads.

1. Operational Excellence:

Focuses on operations, monitoring, automation, and continuous improvement

2. Security:

Protects systems and data through best practices like least privilege and data integrity

3. Reliability:

Emphasizes recovery planning and system adaptability to meet changing demands

4. Performance Efficiency:

Encourages using the right resources for the job and adjusting as needs evolve

5. Cost Optimization:

Helps control and reduce costs through smart provisioning and resource management

6. Sustainability:

Promotes energy-efficient design and environmentally conscious resource usage

AWS Well-Architected Tool

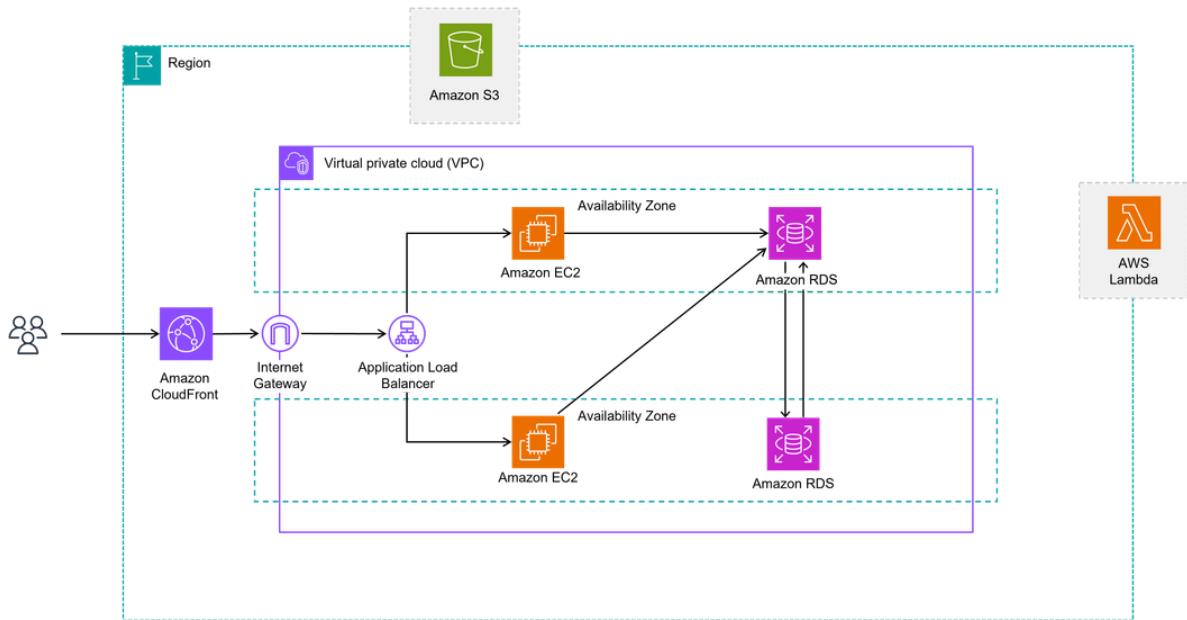
The AWS Well-Architected Tool (AWS WA Tool) is a **free service** that helps **assess** and **improve** **cloud workloads** based on the six key pillars: **Operational Excellence, Security, Reliability, Performance Efficiency, Cost Optimization, and Sustainability**. It offers workload **reviews**, **milestone tracking**, and **custom lenses** for tailored evaluations and **improvement plans**.

Integrated with AWS services like **AWS Identity and Access Management (IAM)** and **APIs**, it supports **team collaboration** and **continuous progress tracking**. The **AWS WA Tool** is ideal for **architects, engineers, and compliance teams**, and it promotes **consistent, actionable, and well-documented architecture reviews**.

Optimizing a cloud architecture

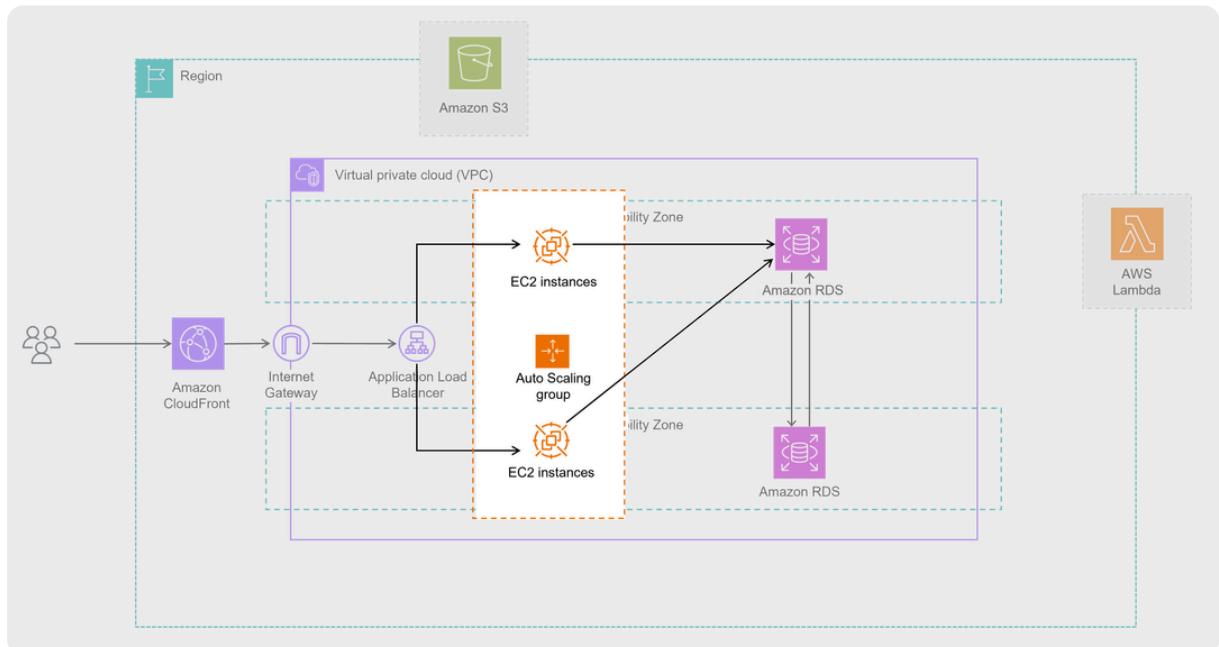
Imagine you own a bustling online florist business, with orders flowing in during peak times like Valentine's Day or Mother's Day. Behind the scenes, the cloud **architecture** needs to be just as **resilient** and **efficient** as the **operations**. Let's use the Well-Architected Framework to optimize the system for **reliability, performance, security, cost savings, and sustainability**.

Starting architecture



Let's look at your current setup. You have a **classic ecommerce architecture**. It includes Amazon Elastic Compute Cloud (Amazon **EC2**) instances for the **website** and Amazon Relational Database Service (Amazon **RDS**) **databases** to handle **orders** and **customer data**. It also has an Amazon Simple Storage Service (Amazon **S3**) bucket **full of product images**. It's functional, but let's evaluate how well it's **scaling** and handling traffic—especially during busy times.

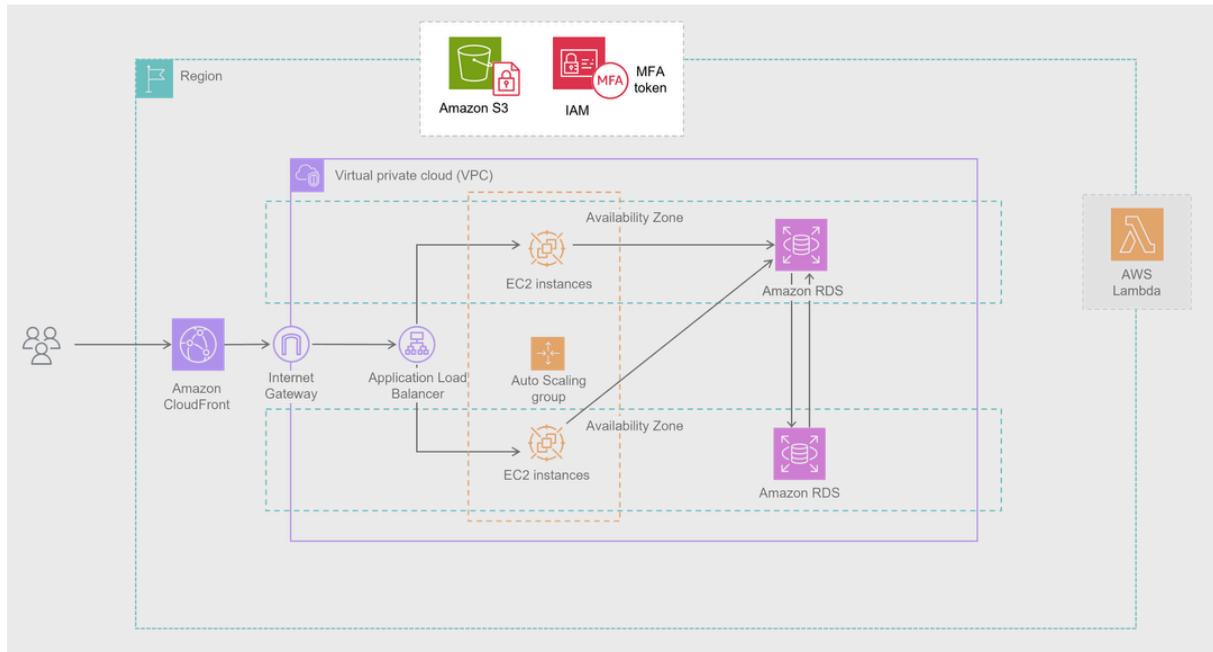
Operational Excellence



Your business is running smoothly, but what happens if an **EC2 instance crashes** during a rush of orders? To be truly **resilient**, you can **automate scaling with EC2 Auto Scaling**. Additionally, to make day to day operations more **reliable** and **efficient**, you can use **infrastructure as code** and implement **self healing mechanisms** like auto-rollback. These practices help your system **adapt** during **high-demand** periods as well as **operate efficiently over time**.

Enhancement: EC2 Auto Scaling

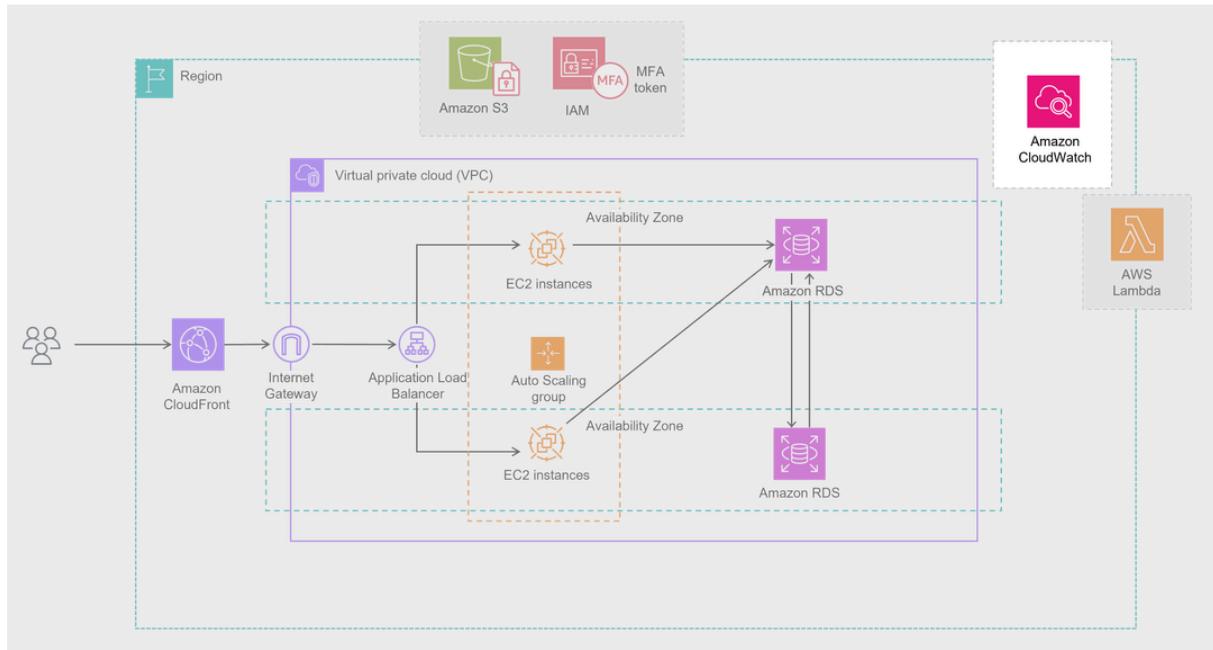
Security



You've already got a secure foundation with an Amazon Virtual Private Cloud (Amazon VPC), but there's more to do. Ask yourself: Are your **EC2** instances regularly patched? Do your **IAM** policies follow the principle of **least privilege**? **Protecting** customer **data**—like names, addresses, and payment info—requires strong **encryption** for **data at rest** and **in transit**, along with **fine-grained access controls**. Strengthening these layers builds trust with your customers and safeguards sensitive information.

Enhancement: Strengthening encryption and IAM policies

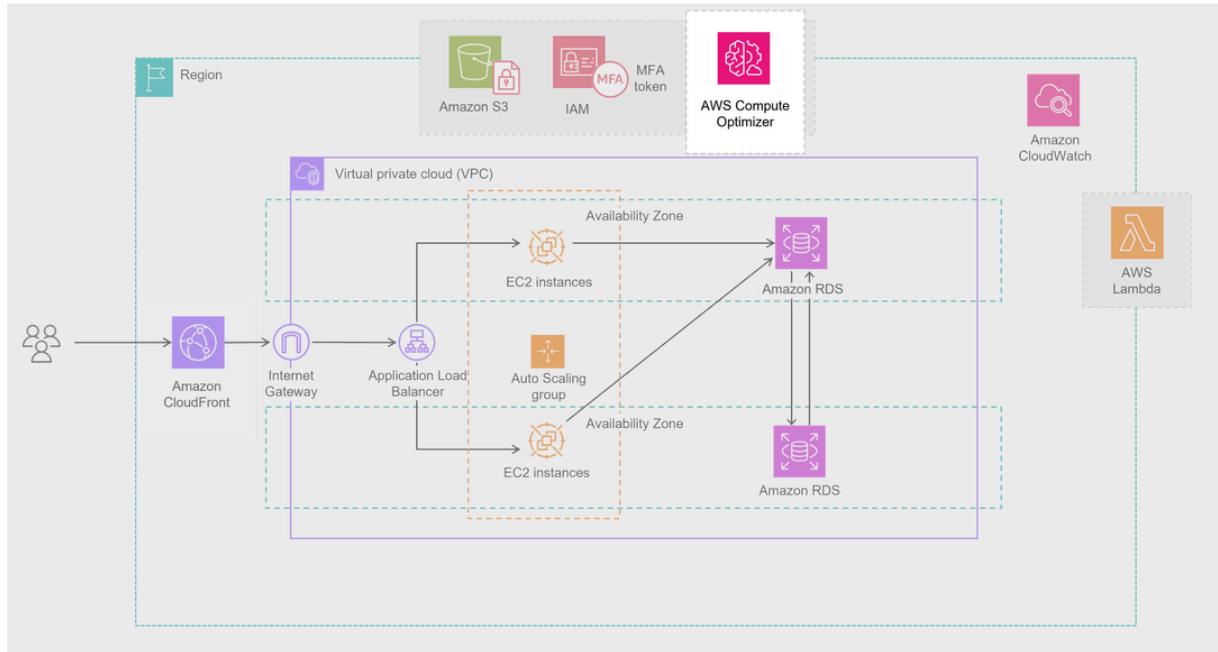
Reliability



During busy seasons, **availability** is everything. You've already taken a great step by deploying resources across **multiple Availability Zones**, but you can increase reliability even further. Use **Amazon CloudWatch** to **monitor** your system's **health** and set up **automated recovery actions**.

Enhancement: Amazon CloudWatch

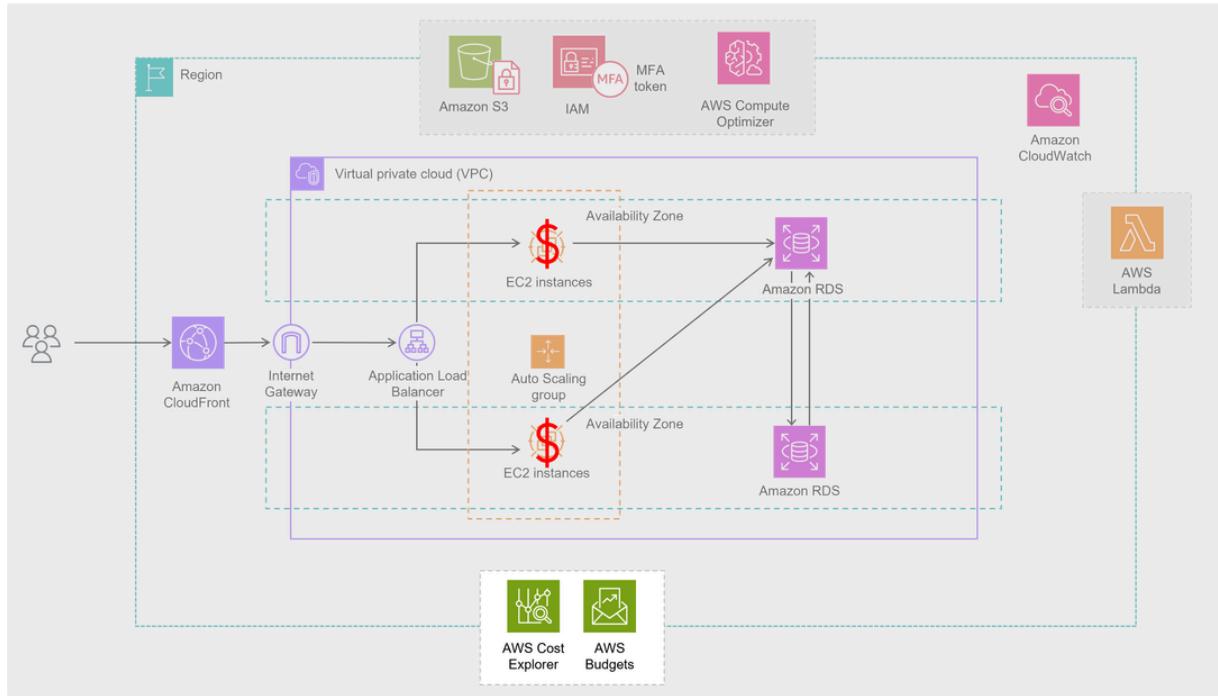
Performance Efficiency



As your business **scales**, your system should scale with it. Are your **EC2 instances** and **RDS instances** **rightsized** for your workload? **AWS Compute Optimizer** can help make sure you're not wasting resources or under **provisioning** your infrastructure. You're already using **AWS Lambda** for **event-driven tasks** like **image processing**, which is great for **flexible scaling**. Make sure those **functions** are **rightsized**, too. And with **Amazon CloudFront**, you can already **deliver product images quickly** to global customers for a smooth, fast shopping experience.

Enhancement: AWS Compute Optimizer

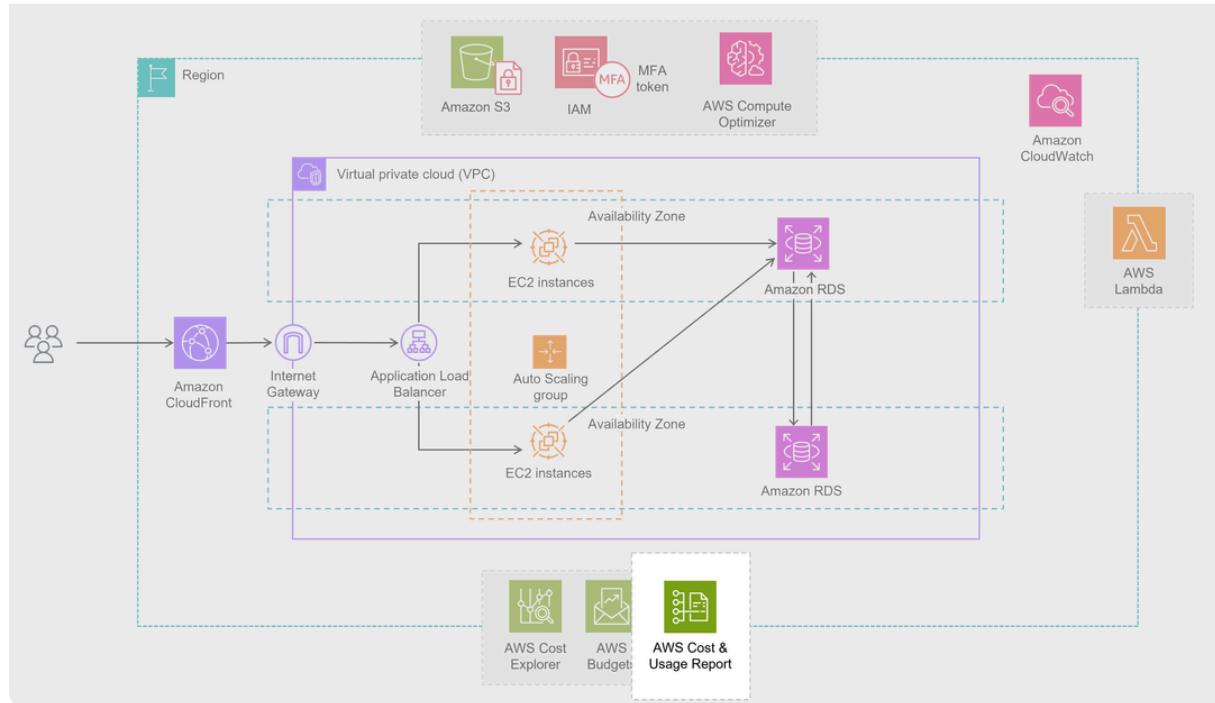
Cost Optimization



You're currently using **On-Demand EC2 instances**, which are great to start with, but switching to **Spot Instances** for variable traffic and **Savings Plans** for steady workloads can **cut costs** significantly. **Track and manage** your cloud **spending in real time** with **AWS Budgets** and **AWS Cost Explorer**. These tools help you make smart, cost-effective decisions while maintaining performance and reliability.

Enhancement: Savings Plans, AWS Budgets, AWS Cost Explorer

Sustainability



Your use of **serverless** and **elastic resources** already reduces your environmental footprint. To go further, continue optimizing workloads to **minimize resource waste**. Doing so benefits both the planet and your bottom line—proving that environmentally conscious decisions can also be business-smart.

Enhancement: AWS Cost & Usage Report

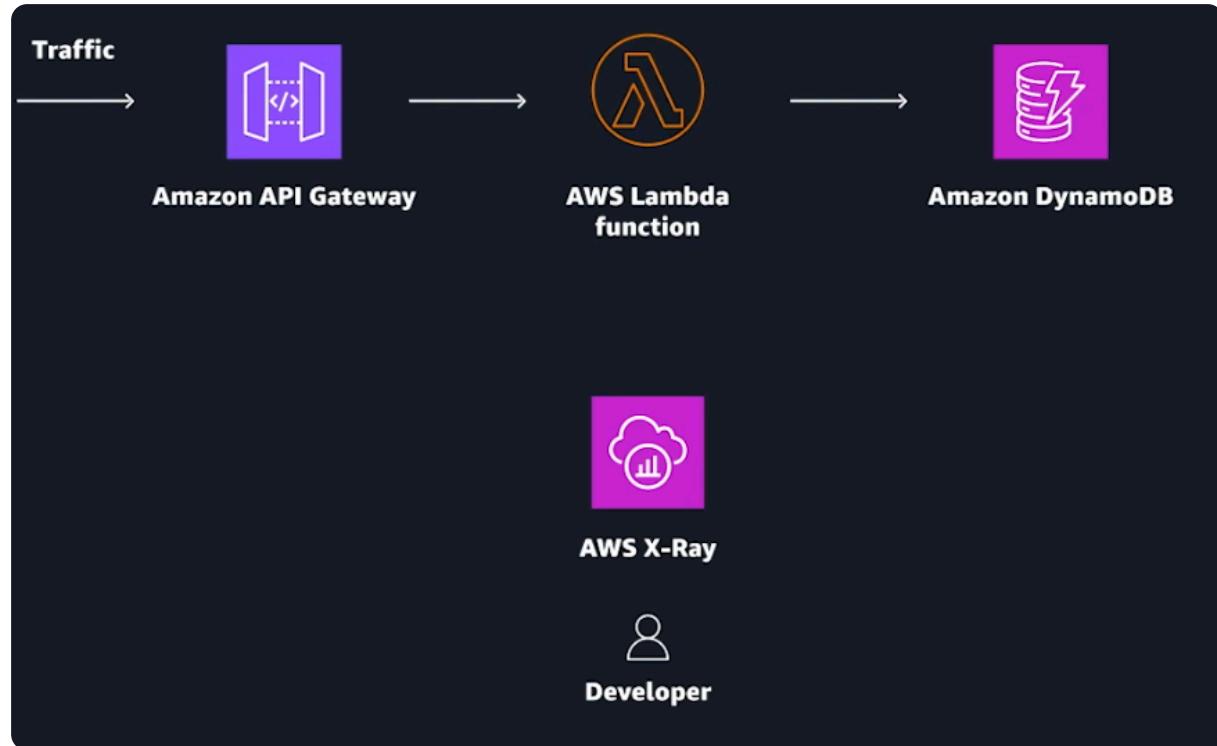
▼ Specialized Use Cases

Alan: Welcome back to Cloud in Real Life. Let's take a look at some really interesting **serverless architectures**. Who wants to kick us off?

Morgan: I do! Okay, I'm gonna start. I love talking about serverless. What if we begin by thinking about a **typical serverless web backend setup**. I'm thinking something like **Amazon API Gateway**, **AWS Lambda**, **Amazon DynamoDB**, and **AWS X-Ray** for tracing. As someone with a developer background, like myself, this is basically the dream team of AWS services for me.

Rudy: This setup makes it possible for you to host your backend services and make them **consumable for frontends** using **API Gateway**. **Data is stored in DynamoDB**, and **X-Ray** helps with **troubleshooting**. Morgan, can you explain how consumers typically interact with these services?

Morgan: Yes. So, customers invoke the APIs exposed by API Gateway by using HTTP. API Gateway receives and validates the request then invokes the Lambda function and then returns that response to the user.



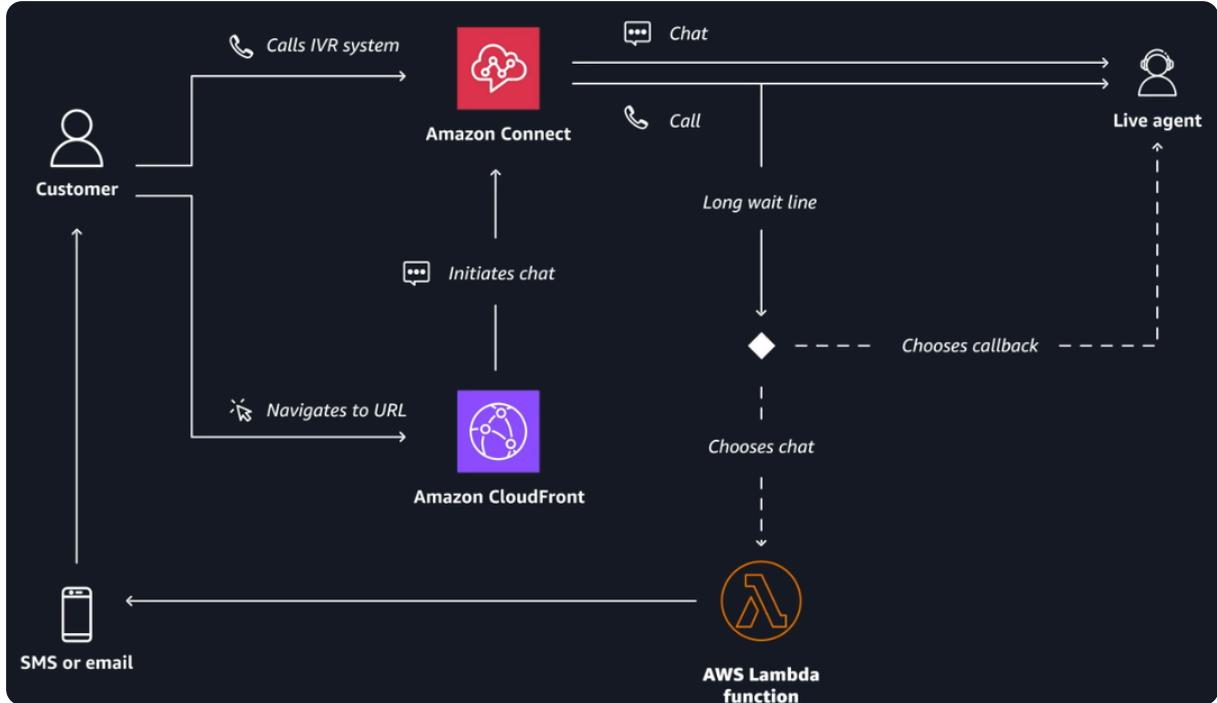
Alan: I really like how this is **entirely serverless**. X-Ray is really cool here because with an architecture like this it's not just a single component like a traditional web server. X-Ray is made to work well in these **distributed environments**, because **X-Ray traces the request through API Gateway, to Lambda, to DynamoDB**, all the way back to the **client**. So, if anything goes wrong, you can use it to determine where the problem is coming from.

Rudy: Ah, sticking with the serverless theme, check this one out. A **static website hosted directly** on Amazon S3. Drag. Drop. Choose Enable. And voila. Static website time. Even better, we have a Contact Us form over here. Before you think we are going to need a server—no, no, no. When the **form is submitted**, it sends a **request to API Gateway**. That **invokes a Lambda function** which, in turn, **sends an email** using **Amazon SES**. Hey, what do you say to that, Morgan?



Morgan: Well, so, this is very similar to the architecture that we just talked about. It's a serverless architecture hosting a web backend using Lambda and API Gateway. But, instead of storing data, this makes an API call to send emails. This is a great example to show how two architectures we covered are similar in the services used but totally different in use case. That's because the code running in a **Lambda function** is just that--**code**. It can be **whatever you need it to be**.

Alan: I have another one. Have you seen this architecture using **Amazon Connect**? It's a bit different from the last use case. But, this is an architecture showing how to use Amazon Connect to provide an alternate channel and **callback option** for **customer support**. Because if there are long wait times for people calling, customers can have the **option to transition** from **voice to chat** or even **email**.



Rudy: Hey, look, as someone who has been on hold with customer support long enough to memorize the song that's playing...

[MUSIC PLAYING]

I'm extremely excited for this solution. It's a great example of how **AWS services** can be **connected** to create **complex, intelligent systems**. In this case, Amazon Connect, Lambda, and Amazon CloudFront. Connect them together, and you have a **smart customer service solution**. Customers can select callback and then hang up. Sign me up, please.

Morgan: Yeah, I really couldn't agree more. Okay, well, this has been a great chat. From serverless websites to smart customer support, we've seen how with just a few managed services, you can tackle a range of complex challenges.

▼ Module Summary

Morgan: Hey there, friends! Great news, you've made it to the end of the course!! Now that you've reached the end, there's only a few things left to say.

Alan: First off, congratulations!! You've stuck with us through a lot of learning, from regions and AZs, to EC2, database migration, and everything in between. With AWS, the possibilities are endless and there's always something new to learn.

Rudy: Yeah, the thing to remember here, is that no one expects you to be a walking AWS encyclopedia. So don't worry. You've learned the basics, and now you can continue your AWS learning journey with confidence.

Morgan: You know, I'm just excited for all of you to dive further into the world of AWS. Everyone starts somewhere, so just be positive, and try it out. Even if it's studying and building for just five minutes of your day.

Alan: Totally. And remember, folks, don't be intimidated by the vast AWS landscape. Start with the basics, and build from there. The more you start to build, the more you'll learn. And you'll figure out that with many of the rad options in the architecture, the sky's the limit.

Rudy: Exactly, and please, please, for me, take the survey at the end of the course, and let us know what you thought of it! Now, cheers Morgan, cheers Alan, and last but not least, cheers to you, cloud practitioners.



AWS Shared Responsibility Model - SRM



AWS Monitoring Services (Dashboards, Logs, Metrics)

▼ Summaries - From AWS skill builder + Links

► Module 1 - Introduction to the Cloud

▼ Module 2 - Compute in the Cloud

Resource link	Description
---------------	-------------

Compute on AWS(opens in a new tab)	This resource provides an overview of the different cloud computing services offered by AWS.
AWS Compute Blog(opens in a new tab)	This blog provides updates, tutorials, and best practices for using AWS compute services, such as Amazon EC2, AWS Lambda, Amazon ECS, and more.
AWS Compute Services(opens in a new tab)	This reference provides an in-depth introduction to the compute services available within the AWS Cloud.
Hands-On Tutorials: Compute(opens in a new tab)	This resource provides practical, step-by-step tutorials designed to help users gain hands-on experience with AWS compute services. It is ideal for beginners and those new to cloud computing.
Amazon EC2(opens in a new tab)	Amazon EC2 runs virtual servers in the cloud with flexible computing capacity.
Amazon EC2 Instance Types(opens in a new tab)	This guide provides detailed information about the different types of EC2 instances, including their specifications, capabilities, and use cases. It helps you choose the right instance type based on your workload needs, such as compute, memory, and storage requirements.
Amazon EC2 Pricing (opens in a new tab)	This guide explains the different pricing models for EC2 instances, including On-Demand, Reserved Instances, and Spot Instances, so you can choose the best option based on your usage.
Amazon EC2 Auto Scaling(opens in a new tab)	Amazon EC2 Auto Scaling automatically adjusts instance count based on demand for high availability and cost-efficiency.
Elastic Load Balancing(opens in a new tab)	Elastic Load Balancing automatically distributes incoming application traffic across multiple EC2 instances for high availability and fault tolerance.
Amazon Simple Notification Service(opens in a new tab)	Amazon SNS is a messaging service for sending notifications to users or other applications through SMS, email, or mobile push notifications.
Amazon Simple Queue Service(opens in a new tab)	Amazon SQS decouples application components through message queuing, storing and processing messages reliably.

▼ Module 3 - Exploring Compute Services

Resource link	Description
Containers on AWS(opens in a new tab)	The AWS Containers Services page provides an overview of the AWS container offerings, including services for container image storage, orchestration, and compute. These offerings are designed to streamline the deployment and management of containerized applications.

Amazon Elastic Container Registry(opens in a new tab)	The Amazon ECR is a fully managed service for storing, managing, and deploying container images securely at scale.
Amazon Elastic Container Service(opens in a new tab)	Amazon ECS is a fully managed service that streamlines the deployment, management, and scaling of containerized applications on AWS.
Amazon Elastic Kubernetes Service (opens in a new tab)	Amazon EKS is a fully managed Kubernetes service that streamlines running Kubernetes clusters on AWS and on premises. It automates infrastructure management and integrates with AWS networking, security, and storage services.
AWS Fargate(opens in a new tab)	Fargate is a serverless compute engine for running containers without managing servers. It is integrated with Amazon ECS and Amazon EKS.
AWS Elastic Beanstalk(opens in a new tab)	Elastic Beanstalk is a fully managed service for deploying and scaling web applications without managing infrastructure.
AWS Batch(opens in a new tab)	AWS Batch is a fully managed service for efficiently running large-scale batch computing jobs on AWS.
What is Amazon Lightsail?(opens in a new tab)	Lightsail is a simplified cloud platform offering VPS, containers, and databases with predictable pricing.
What is AWS Outposts?(opens in a new tab)	AWS Outposts extends AWS infrastructure and services to on-premises locations for low-latency, local data processing.
Choosing a modern application strategy(opens in a new tab)	The AWS Decision Guide for Modern Application Strategy helps organizations determine the most suitable development approach—serverless or Kubernetes—based on their operational model, team structure, and workload requirements.

▼ Module 4 - Going Global

AWS Global Infrastructure(opens in a new tab)	Learn more about the AWS Global Infrastructure.
AWS for the Edge(opens in a new tab)	Learn more about AWS edge locations and edge networking.
AWS CloudFormation(opens in a new tab)	Learn more about the infrastructure as code service, CloudFormation.

▼ Module 5 - Networking

Resource link	Description
Amazon Virtual Private Cloud(opens in a new tab)	Amazon VPC is a service to provision a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you define.

Subnet(opens in a new tab)	A subnet is a section of a VPC that can contain resources and is used to organize your resources. They can contain be either public or private.
Internet gateway(opens in a new tab)	An internet gateway is a connection between a VPC and the internet. It allows public traffic from the internet to access your VPC.
Virtual private gateway(opens in a new tab)	A virtual private gateway is the component that allows protected internet traffic to enter into the VPC. It allows a connection between your VPC and a private network only if it is coming from an approved network.
AWS Client VPN(opens in a new tab)	Amazon Client VPC is a networking service you can use to connect your remote workers and on-premises networks to the cloud. It is a fully managed, elastic VPN service that automatically scales up or down based on user demand.
AWS Site-to-Site VPN(opens in a new tab)	AWS Site-to-Site VPN creates a secure connection between your data center or branch offices and your AWS Cloud resources.
AWS PrivateLink(opens in a new tab)	AWS PrivateLink is a highly available, scalable technology that you can use to privately connect your VPC to services and resources as though they were in your VPC.
AWS Direct Connect(opens in a new tab)	AWS Direct Connect is a service that provides a dedicated private connection between your data center and a VPC.
Network Access Control List (network ACL)(opens in a new tab)	A network ACL allows or denies specific inbound or outbound traffic at the subnet level using stateless packet filtering.
Security groups(opens in a new tab)	Security groups control the inbound and outbound traffic for a resource at the instance level using stateful packet filtering.
Domain Name System (DNS)(opens in a new tab)	DNS translates human readable domain names to machine readable IP addresses (for example, 192.0.2.0).
Amazon Route 53 (opens in a new tab)	Route 53 is a scalable and reliable DNS web service that helps developers and businesses route end users to internet applications, whether they're hosted in AWS or elsewhere. It also supports domain registration, health checks, and advanced traffic routing policies.
Amazon CloudFront(opens in a new tab)	CloudFront is a web service that speeds up distribution of your web content to your users through a worldwide network of data centers called edge locations. It securely delivers content with low latency and high transfer speeds.
AWS Global Accelerator(opens in a new tab)	Global Accelerator is a networking service that helps improve the availability and performance of applications for global users by routing traffic through the AWS global network. It helps improve application availability, performance, and security.

Amazon Transit Gateway(opens in a new tab)	Amazon VPC Transit Gateways is a network transit hub used to interconnect VPCs and on-premises networks.
NAT Gateway(opens in a new tab)	Network Address Translation (NAT) gateway allows instances in a private subnet to connect with services outside your VPC. External services can't initiate a connection with those instances.
API Gateway(opens in a new tab)	The Amazon API Gateway is an AWS service for creating, publishing, maintaining, monitoring, and securing APIs at any scale. It handles all the tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls.

▼ Module 6 - Storage

Resource link	Description
Amazon EC2 Instance Store User Guide(opens in a new tab)	A temporary storage option that is directly attached to the host computer of an EC2 instance, providing high-performance but non-persistent storage.
Amazon Elastic Block Store (Amazon EBS)(opens in a new tab)	A scalable block storage service that provides persistent, high-performance volumes you can attach to your EC2 instances for data storage and applications.
Amazon Elastic Block Store (Amazon EBS) FAQ(opens in a new tab)	Frequently asked questions about Amazon EBS.
Amazon EBS Snapshots User Guide(opens in a new tab)	EBS Snapshots are point-in-time backups of your cloud storage volumes, making it possible to protect data and restore it when needed.
Amazon Data Lifecycle Manager User Guide(opens in a new tab)	A service that streamlines the creation, retention, and deletion of Amazon EBS snapshots.
Amazon Simple Storage Service (Amazon S3)(opens in a new tab)	A scalable cloud storage service that can store and retrieve any amount of data from anywhere on the web.
Amazon Simple Storage Service (Amazon S3) FAQ(opens in a new tab)	Frequently asked questions about Amazon S3.
Amazon S3 Storage Classes(opens in a new tab)	Amazon S3 offers various storage classes, from high-performance frequent access to cost-effective archival options, tailored to different data retrieval needs and budget constraints.
Amazon S3 Versioning User Guide(opens in a new tab)	Amazon S3 versioning keeps multiple variants of objects, offering recovery from unintended

	deletions or modifications by preserving every update to your files.
Amazon S3 Buckets User Guide (opens in a new tab)	S3 buckets are cloud storage containers that securely hold various types of data, allowing convenient access and management through the AWS online infrastructure.
Amazon Elastic File System (Amazon EFS) (opens in a new tab)	A scalable, fully-managed file storage service that lets multiple AWS resources access shared data simultaneously without capacity planning.
Amazon Elastic File System (Amazon EFS) FAQ (opens in a new tab)	Frequently asked questions about Amazon EFS.
Amazon FSx (opens in a new tab)	A fully managed file storage service that lets you launch and run file systems like Windows File Server, Lustre, NetApp ONTAP, and OpenZFS in the AWS cloud.
Amazon FSx for Windows File Server (opens in a new tab)	An Amazon FSx option providing reliable, high-performance file storage compatible with Windows applications in the AWS Cloud.
Amazon FSx for NetApp ONTAP (opens in a new tab)	An Amazon FSx option providing file storage with advanced data management capabilities and compatibility with both Windows and Linux workloads on AWS.
Amazon FSx for Open ZFS (opens in a new tab)	An Amazon FSx option that provides high-performance, scalable storage using the popular open-source ZFS file system.
Amazon FSx for Lustre (opens in a new tab)	An Amazon FSx option designed to accelerate workloads by providing fast data access for compute-intensive applications in AWS.
AWS Storage Gateway (opens in a new tab)	A hybrid cloud storage service that provides seamless and secure integration between on-premises environment and AWS cloud storage services.
Amazon S3 File Gateway (opens in a new tab)	A Storage Gateway configuration that provides local file access to S3 objects while caching frequently accessed data locally for faster retrieval.
Tape Gateway (opens in a new tab)	A Storage Gateway configuration used for backing up data to Amazon S3 while maintaining compatibility with existing tape-based backup applications.
Volume Gateway (opens in a new tab)	A Storage Gateway configuration that provides iSCSI block storage volumes to on-premises applications, offering both cached and stored modes.

▼ Module 7 - Databases

Resource link	Description
Amazon Relational Database Service (Amazon RDS)(opens in a new tab)	A relational database service supporting multiple engines like MySQL, PostgreSQL, and Microsoft SQL Server with automated maintenance and backups
Amazon RDS Security(opens in a new tab)	Detailed information about security configurations in Amazon RDS
Amazon Aurora(open s in a new tab)	A cloud-native database offering superior performance and availability over traditional databases while maintaining MySQL and PostgreSQL compatibility
AWS Database Migration Service (AWS DMS)(opens in a new tab)	A service that provides seamless database migration between source and target databases while keeping the source database operational
Amazon DynamoDB (opens in a new tab)	A NoSQL database service providing single-digit millisecond performance at any scale with built-in security
Amazon ElastiCache (opens in a new tab)	An in-memory caching service that supports Redis, Valkey, or Memcached to improve application performance through faster data retrieval
Amazon DocumentDB(opens in a new tab)	A MongoDB-compatible document database service designed for mission-critical workloads with automatic scaling
Amazon Backup(ope ns in a new tab)	A centralized service for automating and managing data backups across AWS services and on-premises resources
Amazon Neptune(opens in a new tab)	A graph database service optimized for storing and querying highly connected data relationships
What Is a Relational Database?(opens in a new tab)	A structured database using tables with predefined schemas, supporting complex queries and transactions through SQL for consistent data relationships
What Is a NoSQL Database?(opens in a new tab)	A nonrelational database offering flexible schemas and high scalability for varied data types, optimized for specific data models and patterns
What Is an In-Memory Caching Service?(opens in a new tab)	A high-speed data storage layer using RAM instead of disk storage, delivering microsecond latency for frequently accessed data
AWS Shared Responsibility Model(open s in a new tab)	AWS is responsible for security of the cloud (infrastructure, hardware, networking, facilities) while customers are responsible for security in the cloud (data, configuration, access management).

▼ Module 8 - AI/ML and Data Analytics

Amazon Comprehend (opens in a new tab)	Use natural language processing to extract key insights from documents.
Amazon Polly (opens in a new tab)	Convert text into lifelike speech.
Amazon Transcribe (opens in a new tab)	Convert speech into text.
Amazon Translate (opens in a new tab)	Translate text into multiple languages.
Amazon Kendra (opens in a new tab)	Use natural language processing to intelligently query enterprise content.
Amazon Rekognition (opens in a new tab)	Identify objects and activities in images and videos.
Amazon Textract (opens in a new tab)	Detect and extract typed and handwritten text in documents.
Amazon Lex (opens in a new tab)	Add voice and text conversational interfaces to applications.
Amazon Personalize (opens in a new tab)	Add personalized customer recommendations to applications.
Amazon SageMaker AI (opens in a new tab)	Build, train, and deploy your own ML models without worrying about infrastructure.
Amazon SageMaker JumpStart (opens in a new tab)	Deploy pre-trained ML solutions, like computer vision, NLP, and tabular data, with just a few clicks.
Amazon Bedrock (opens in a new tab)	Fine-tune and seamlessly integrate large FMs from Amazon and leading AI startups into your AWS applications with a single API.
Amazon Q Business (opens in a new tab)	Answer questions and solve problems using the data and expertise found in your company's information repositories.
Amazon Q Developer (opens in a new tab)	Accelerate development with code recommendations.
Amazon Kinesis Data Streams (opens in a new tab)	Ingest terabytes of streaming data in real time.
Amazon Data Firehose (opens in a new tab)	Ingest and deliver data within seconds to multiple consumers, such as data lakes and warehouses.
Amazon S3 (opens in a new tab)	Store virtually limitless amounts of all types of data using this popular data lake choice.
Amazon Redshift (opens in a new tab)	Store petabytes of structured or semistructured data in this data warehouse service. Analyze large datasets stored in the warehouse using complex SQL queries.
AWS Glue Data Catalog (opens in a new tab)	Provide metadata to various analytics services with this centralized repository.

AWS Glue (opens in a new tab)	Process data by using the AWS Glue Data Catalog as a reference.
Amazon EMR (opens in a new tab)	Process big data workloads by using popular frameworks like Apache Spark.
Amazon Athena (opens in a new tab)	Analyze various data sources hosted anywhere with a single SQL query.
Amazon QuickSight (opens in a new tab)	Visualize data by creating interactive dashboards with or without expertise.
Amazon OpenSearch Service (opens in a new tab)	Visualize and monitor real-time data analytics with keyword or NLP searches.

▼ Module 9 - Security

Resource link	Description
AWS Identity and Access Management (IAM) (opens in a new tab)	Securely manage identities and access to AWS services and resources.
AWS IAM Identity Center (opens in a new tab)	Connect your existing workforce identity source and centrally manage access to AWS with single sign-on.
AWS Secrets Manager (opens in a new tab)	Centrally store and manage credentials, API keys, and other secrets.
AWS Systems Manager (opens in a new tab)	Manage nodes, or connection points, at scale on AWS and in multi-cloud and hybrid environments.
AWS Shield (opens in a new tab)	Protect your network and applications from the most common, frequently occurring types of DDoS attacks.
AWS WAF (opens in a new tab)	Protect your network and applications from blocked IP addresses defined by a web ACL.
AWS Key Management Service (AWS KMS) (opens in a new tab)	Create and manage cryptographic keys to encrypt and decrypt your data.
Amazon Macie (opens in a new tab)	Certify that sensitive data is discovered and protected in Amazon S3.
AWS Certificate Manager (ACM) (opens in a new tab)	Create and manage SSL/TLS certificates that provide data encryption in transit.
Amazon Inspector (opens in a new tab)	Check applications for security vulnerabilities and deviations from security best practices.
Amazon GuardDuty (opens in a new tab)	Continuously monitor the AWS environment with intelligent threat detection.

Amazon Detective (opens in a new tab)	Analyze threats with interactive visualizations contained in a unified view.
AWS Security Hub (opens in a new tab)	Aggregate security findings and organize them into actionable insights.

▼ Module 10 - Monitoring, Compliance, and Governance in the AWS Cloud

Resource link	Description
Amazon CloudWatch (opens in a new tab)	CloudWatch monitors your AWS resources and the applications you run on AWS in real time. With CloudWatch, you gain system-wide visibility into resource utilization, application performance, and operational health.
AWS CloudTrail (opens in a new tab)	CloudTrail enables auditing, security monitoring, and operational troubleshooting. CloudTrail records user activity and API calls across AWS services as events. CloudTrail events help you answer the question of "Who did what, where, and when?"
AWS Artifact (opens in a new tab)	AWS Artifact a self-service portal that provides on-demand access to AWS security and compliance documentation, including reports, certifications, and agreements.
AWS Config (opens in a new tab)	AWS Config is a service to assess, audit, and evaluate the configurations of your AWS resources.
AWS Audit Manager (opens in a new tab)	Audit Manager is a service to continually audit your AWS usage to streamline risk and compliance assessment.
AWS Organizations (opens in a new tab)	Organizations helps you centrally manage and govern your environment as you grow and scale your AWS resources. It helps you manage policies for groups of accounts and automate account creation.
AWS Control Tower (opens in a new tab)	With AWS Control Tower, you can enforce and manage governance rules for security, operations, and compliance at scale across all your organizations and accounts in the AWS Cloud.
AWS Service Catalog (opens in a new tab)	With Service Catalog, you can create, share, and organize from a curated catalog of AWS resources. You can deploy baseline networking resources and security tools for new AWS accounts so you can govern consistently.
AWS License Manager (opens in a new tab)	License Manager is a service that helps you manage your software licenses and fine-tune your licensing costs.
AWS Trusted Advisor (opens in a new tab)	Trusted Advisor helps you optimize costs, increase performance, improve security and resilience, and operate at scale in the cloud. It continuously evaluates your AWS environment using best practice checks

	across those categories and recommends actions to remediate deviations from best practices.
AWS Health(opens in a new tab)	AWS Health is the data source for events and changes affecting your AWS Cloud resources. AWS Health notifies you about service events, planned changes, and account notifications to help you manage and take actions.
AWS Identity and Access Management Access Analyzer(opens in a new tab)	IAM Access Analyzer provides capabilities to set, verify, and refine security permissions to achieve least privilege security standards.

▼ Module 11 - Pricing and Support

Resource link	Description
Amazon S3 Pricing(opens in a new tab)	The Amazon S3 Pricing page outlines costs for storage, requests, data transfer, and additional features, along with details on AWS Free Tier for new customers.
Amazon EC2 On-Demand Pricing(opens in a new tab)	The Amazon EC2 On-Demand Pricing page details the costs for EC2 instances, including pricing for instance types, data transfer, and additional features with no long-term commitments.
What is AWS Organizations?(opens in a new tab)	The AWS Organizations User Guide explains how to centrally manage multiple AWS accounts, automate account creation, apply governance policies, and simplify billing and resource sharing.
AWS Billing Console (opens in a new tab)	The AWS Billing Console provides a centralized platform for managing and understanding AWS charges, offering tools to view and download invoices, monitor discounts and credits, and analyze spending trends across multiple accounts.
AWS Budgets(opens in a new tab)	AWS Budgets provides tools to create custom budgets for tracking and managing AWS costs, usage, and Reserved Instance or Savings Plan use. It also helps prevent overspending by offering alerts and automated actions.
AWS Cost Explorer(opens in a new tab)	AWS Cost Explorer provides tools to visualize, analyze, and manage AWS costs and usage, with customizable reports and cost forecasting to optimize cloud spending.
AWS Pricing Calculator Documentation(opens in a new tab)	The AWS Pricing Calculator is a free, web-based tool that helps customers create detailed cost estimates for AWS services, so they can plan and manage their cloud expenditures effectively.
AWS re:Post(opens in a new tab)	AWS re:Post is a community-driven, question-and-answer platform provided where users can

	seek help, share knowledge, and find solutions related to AWS services and technologies.
AWS Trust and Safety Center(opens in a new tab)	The AWS Trust and Safety Center offers guidance on reporting abuse, protecting applications, and following best practices for digital messaging on AWS.
AWS Professional Services(opens in a new tab)	The AWS Professional Services page highlights how AWS experts help organizations accelerate cloud adoption with tailored guidance across industries and technologies.
Welcome to AWS Documentation(opens in a new tab)	The AWS Documentation page provides comprehensive technical resources, including guides, API references, tutorials, and best practices across AWS services.
AWS Marketplace(opens in a new tab)	The AWS Marketplace is a digital catalog where customers can discover, purchase, and manage third-party software, data, and services with flexible pricing options.
Engage with AWS Partners(opens in a new tab)	The AWS Partners page helps you discover how AWS Partners can help you accelerate your cloud journey.
Funding Benefits for AWS Partners(opens in a new tab)	The AWS Partner Funding page outlines funding benefits, including credits and discounts, to support AWS partners in developing, marketing, and selling solutions based in AWS.
AWS Partner Events (opens in a new tab)	The AWS Partner Events page lists upcoming webinars, workshops, and in-person events for IT professionals, developers, and business leaders, with options to filter by date, location, and topic.
AWS Partner Training and Certification(opens in a new tab)	The AWS Partner Training page offers courses, certifications, and resources to help AWS Partners build cloud skills across various roles and enhance their expertise.

▼ Module 12 - Migrating to the AWS Cloud

Resource link	Description
Three Phases of Migration(opens in a new tab)	In performing a large-scale migration, there are typically three phases: assess, mobilize, and the last phase, migrate and modernize.
AWS Cloud Adoption Framework(opens in a new tab)	AWS CAF is a framework that brings AWS experience and best practices to companies preparing to migrate to the AWS Cloud. The framework provides tools to help accelerate the migration journey, organize resources, and align management during the transition.
Seven Migration Strategies (7)	When migrating to the cloud, there are seven common migration strategies that customers can choose

Rs)(opens in a new tab)	to implement. They are relocate, rehost, replatform, refactor, repurchase, retain, and retire.
Migration Evaluator(opens in a new tab)	The Migration Evaluator is a migration assessment service that helps you create a business case for AWS Cloud planning and migration. It does this with a data-driven approach, analyzing your current state and target state, and developing a migration readiness plan with projected cloud costs.
AWS Application Discovery Service(opens in a new tab)	Application Discovery Service discovers on-premises server inventory and connections. It gathers configuration, performance, and connection details for both servers and databases to create a detailed migration plan.
AWS Application Migration Service(opens in a new tab)	Application Migration Service is a tool to migrate and improve your on-premises and cloud-based applications. It helps customers simplify, expedite, and reduce the cost of migrating and modernizing applications.
AWS Migration Hub(opens in a new tab)	Migration Hub is a centralized hub to take you from discovery to assessment, planning, and implementation of your migration. It provides tools, guidance, and automated recommendations to collaborate with your team and track your migration.
AWS Database Migration Service (AWS DMS)(opens in a new tab)	AWS DMS makes it possible to quickly and securely migrate databases to the AWS Cloud. It provides a way to plan, assess, convert, and migrate databases even with data warehouses in one central tool.
AWS Schema Conversion Tool (AWS SCT)(opens in a new tab)	AWS SCT is a service that makes it possible to convert database schemas and code objects (like stored procedures, views, and functions) from one database engine to another.
AWS DataSync (opens in a new tab)	DataSync is specifically designed for automating and accelerating secure data migrations. DataSync manages data movement workloads with bandwidth throttling, migration scheduling, task filtering, and task reporting. It also provides rapid data replication.
AWS Transfer Family(opens in a new tab)	Transfer family makes it possible to seamlessly manage and share data with simple, secure, and scalable file transfers. This service provides fully managed support for secure file transfers over SFTP, FTPS, FTP, and others. It helps you transfer files directly into and out of AWS storage services like Amazon S3 and Amazon EFS.
AWS Direct Connect(opens in a new tab)	Direct Connect is a cloud service that you can use to establish a private, dedicated network connection between your on-premises network and AWS.
AWS Snow Family(opens in a new tab)	AWS Snowball Edge Storage Optimized devices are a great solution for offline data migration where

<a data-bbox="265 145 462 314" href="#">new tab)	connecting to the internet might not be an option. Snowball Edge Storage Optimized devices deliver high performance storage, and make it possible to simplify multi-petabyte data migrations from on-premises locations to AWS.
------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

▼ Module 13 - Well-Architected Solution

Resource link	Description
<a data-bbox="265 530 558 597" href="#">AWS CodeBuild (opens in a new tab)	Build scripts for compiling, testing, and packaging your code.
<a data-bbox="265 619 558 687" href="#">AWS CodePipeline (opens in a new tab)	Automate and monitor CI/CD pipelines.
<a data-bbox="265 709 558 777" href="#">AWS X-Ray (opens in a new tab)	Monitor and debug applications.
<a data-bbox="265 799 558 866" href="#">AWS AppSync (opens in a new tab)	Connect applications to multiple data sources using a single GraphQL API request.
<a data-bbox="265 889 558 956" href="#">AWS Amplify (opens in a new tab)	Quickly develop, deploy, and manage full-stack applications on AWS.
<a data-bbox="265 979 558 1046" href="#">Amazon Connect (opens in a new tab)	Manage customer service operations with artificial intelligence.
<a data-bbox="265 1069 558 1136" href="#">Amazon SES (opens in a new tab)	Use automation to send large volumes of transactional and marketing emails.
<a data-bbox="265 1158 558 1226" href="#">Amazon AppStream 2.0 (opens in a new tab)	Stream cloud-based and desktop applications to users on compatible devices.
<a data-bbox="265 1248 558 1316" href="#">Amazon WorkSpaces (opens in a new tab)	Provide employees with secure virtual desktops.
<a data-bbox="265 1338 558 1439" href="#">Amazon WorkSpaces Secure Browser (opens in a new tab)	Provide employees with secure access to web-based applications and sites.
<a data-bbox="265 1462 558 1529" href="#">AWS IoT Core (opens in a new tab)	Securely connect devices with cloud applications to create IoT solutions.
<a data-bbox="265 1551 558 1619" href="#">AWS Well-Architected (opens in a new tab)	The AWS Well-Architected Framework provides a structured approach to designing and operating secure, high performing, resilient, and efficient infrastructure for applications and workloads on AWS, guided by six key pillars.
<a data-bbox="265 1731 558 1832" href="#">AWS Well-Architected Framework (opens in a new tab)	This website is the official documentation for the AWS Well-Architected Framework.
<a data-bbox="265 1855 558 1945" href="#">What is AWS Well-Architected Tool? (opens in a new tab)	The AWS Well-Architected Tool provides a consistent process for measuring your architecture using AWS best practices.
<a data-bbox="265 1967 558 2046" href="#">AWS Architecture Center (opens in a new tab)	The AWS Architecture Center provides reference architectures and best practices for building secure, efficient, and scalable solutions on AWS.

[AWS Architecture Blog
\(opens in a new tab\)](#)

The AWS Architecture Blog shares technical insights and best practices for building effective solutions on AWS.

▼ Useful Links:

195 flashcards

<https://quizlet.com/392359849/aws-cloud-practitioner-exam-questions-flash-cards/?i=2u6uqa&x=1jqY>

Free content

<https://www.w3schools.com/aws/index.php>

A summary

<https://sathittham.medium.com/aws-cloud-practitioner-certification-cheat-sheet-1191b36137a8>

Exam samples Github

<https://github.com/kananinirav/AWS-Certified-Cloud-Practitioner-Notes/tree/master#>

Architecture Diagramming

<https://icepanel.io/software-architecture-diagramming>