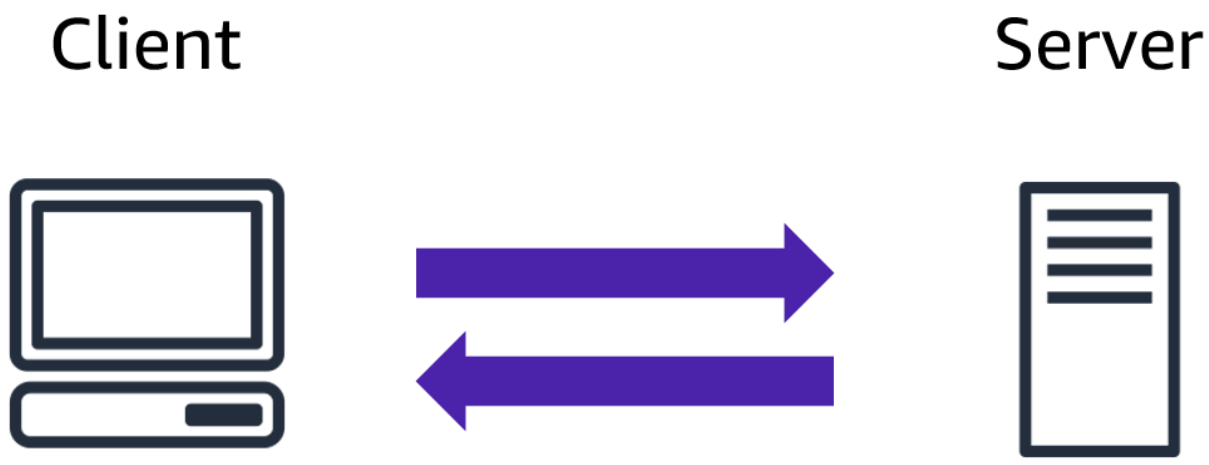
What is a Client-Server Model?

In computing, a client can be a web browser or desktop application that a person interacts with to make requests to computer servers. A server can be services such as Amazon Elastic Compute Cloud (Amazon EC2), a type of virtual server.

For example, suppose that a client makes a request for a news article, the score in an online game, or a funny video. The server evaluates the details of this request and fulfills it by returning the information to the client.



What is cloud computing?

Cloud computing is the delivery of computing services—including servers, storage, databases, networking, software, analytics, and intelligence—over the internet ("the cloud") to offer faster innovation, flexible resources, and economies of scale.

You typically pay only for cloud services you use, helping you lower your operating costs, run your infrastructure more efficiently, and scale as your business needs change.

Deployment Models for Cloud Computing

When selecting a cloud strategy, a company must consider factors such as required cloud application components, preferred resource management tools, and any legacy IT infrastructure requirements.

The three cloud computing deployment models

- Cloud-based Deployment
- On-premises Deployment
- Hybrid Deployment

Cloud-based Deployment -

Simple words - Imagine you have a company, but instead of keeping all your computer equipment in your building, you rent space in a giant data center (like a huge warehouse for computers) run by another company. That's a cloud-based deployment.

You access your applications and data over the internet, and the cloud provider takes care of all the hardware, software updates, security, and maintenance. It's like renting an apartment instead of owning a house – you don't have to worry about fixing the roof or mowing the lawn.

- Run all parts of the application in the cloud.
- Migrate existing applications to the cloud.
- Design and build new applications in the cloud.

In a cloud-based deployment model, you can migrate existing applications to the cloud, or you can design and build new applications in the cloud. You can build those applications on low-level infrastructure that requires your IT staff to manage them. Alternatively, you can build them using higher-level services that reduce the management, architecting, and scaling requirements of the core infrastructure.

For example, a company might create an application consisting of virtual servers, databases, and networking components[different parts of your application to talk to each other] that are fully based in the cloud.

On-premises Deployment

Simple Words - Imagine you have a company. An on-premises deployment means you keep all the computer equipment (servers, networking gear, etc.) and software needed to run your company's applications inside your own building, in a dedicated room (often called a server room). You're responsible for everything: buying the equipment, setting it up, maintaining it, keeping it cool, and making sure it's secure. It's like owning and running your own power plant for your house.

- Deploy resources by using virtualization and resource management tools.
- Increase resource utilization by using application management and virtualization technologies.

On-premises deployment is also known as a private cloud deployment. In this model, resources are deployed on premises by using virtualization and resource management tools.

For example, you might have applications that run on technology that is fully kept in your on-premises data center. Though this model is much like legacy IT infrastructure, its incorporation of application management and virtualization technologies helps to increase resource utilization.

Hybrid Deployment

Simple Words - A hybrid deployment is like having a mix of both worlds: some of your company's computer systems and applications run in your own building (on-premises), while others run in the cloud.

A hybrid approach lets companies choose the best place for different parts of their systems. For example, they might keep sensitive data on-premises for security reasons, but use the cloud for things like website hosting or email. It's a way to balance control and flexibility.

- Connect cloud-based resources to on-premises infrastructure.
- Integrate cloud-based resources with legacy IT applications.

In a hybrid deployment, cloud-based resources are connected to on-premises infrastructure. You might want to use this approach in a number of situations. For example, you have legacy applications that are better maintained on premises, or government regulations require your business to keep certain records on premises.

Cloud computing offers several key advantages for businesses:

Pay as you go: Instead of investing heavily in servers and data centers upfront, you only pay for the computing resources you use. This saves money and allows for flexible spending.

Focus on your work, not IT: Cloud computing handles the maintenance of servers and infrastructure, freeing you to concentrate on your applications and customers.

No more guessing: You don't need to predict how much computing power you'll need. Cloud computing lets you easily adjust resources up or down as demand changes, so you're never paying for unused capacity or running out of resources.

Cost savings: Cloud providers achieve economies of scale, meaning they can offer lower prices than you could get on your own.

Faster and more agile: Cloud computing makes it quicker to develop and deploy applications. You can get new resources in minutes, not weeks, allowing for faster innovation.

Global reach: You can easily deploy applications to customers worldwide, providing them with fast access and low latency, no matter where they are.

Cloud Computing Models

Imagine renting a house. Cloud computing has three main ways to "rent" IT resources, just like renting a house, but for computers:

IaaS (Infrastructure as a Service) / CaaS (Compute as a Service) This is like renting an empty house. You get the walls, roof, and basic utilities (like electricity and water), but you have to furnish it yourself. You control everything inside: what furniture you buy, how you arrange it, etc. In cloud terms, you get the servers, storage, and network, but you install and manage the operating systems, software, and everything else. It gives you the most control, but also the most responsibility.

What it provides: Think of it as the foundation. You get access to:

- **Compute:** Virtual machines (like computers in the cloud) or dedicated physical servers.
- **Storage:** Space to store your data, whether it's files, databases, or applications.
- **Networking:** Tools to connect your resources, like virtual networks, firewalls, and load balancers.

What you manage: You're responsible for:

- **Operating systems:** Installing and managing the software that runs your virtual machines.
- **Middleware:** Software that connects applications, like databases and web servers.
- **Applications:** Installing and managing your own software.
- **Data:** Protecting and managing your data.

Who uses it:

- **Businesses with IT expertise:** Companies that need a high degree of control over their infrastructure.
- **Startups:** Companies that need to quickly scale their resources.
- **Developers:** Those who need a flexible environment to build and test applications.

Examples: Amazon Web Services (AWS) EC2, Microsoft Azure Virtual Machines, Google Compute Engine

PaaS (Platform as a Service): This is like renting a furnished house. The furniture (operating systems and software) is already there, and you can't really change it much, but you can decorate and personalize the space (develop and deploy your applications). You don't have to worry about maintaining the furniture (managing the underlying infrastructure). It's less control than IaaS, but much less work.

- **What it provides:** A complete environment for developing, deploying, and managing applications. You get:
    - **Everything in IaaS, plus:**
    - **Operating systems:** Managed for you.
    - **Middleware:** Like databases and web servers, ready to use.
    - **Development tools:** Frameworks, libraries, and tools to build applications.
- **What you manage:** You focus on:
    - **Applications:** Developing, deploying, and managing your applications.
    - **Data:** Protecting and managing your data.
- **Who uses it:**
    - **Developers:** Those who want to focus on coding and not infrastructure.
    - **Businesses with development teams:** Companies that want to streamline their development process.
- **Examples:** AWS Elastic Beanstalk, Google App Engine, Heroku

SaaS (Software as a Service): This is like renting an apartment. You get a ready-to-use living space. You don't worry about the building's plumbing, electricity, or even the furniture. You just use what's there (the completed application). Think of Gmail or Netflix – you just use the service; you don't worry about how it works behind the scenes. It's the least control, but also the least maintenance.

What it provides: Ready-to-use software applications delivered over the internet.

You get: A complete application: Accessible from anywhere with an internet connection.

What you manage: Nothing! The provider handles everything.

Who uses it: End-users: Anyone who needs to use a specific application.
Businesses of all sizes: Companies that want to use software without managing it.
Examples: Gmail, Salesforce, Dropbox

What is Virtualization?

Imagine you have a powerful computer. Traditionally, you'd install one operating system (like Windows or macOS) on it, and run your applications within that system. Virtualization is like having a superpower that lets you split that single computer into multiple virtual computers, all running at the same time. Each of these virtual computers can have its own operating system and applications, completely separate from the others.

Why is it Important in Cloud Computing?

Cloud providers have massive data centers filled with computers. Virtualization lets them use these computers very efficiently. Instead of one customer getting an entire physical machine, they can get a virtual machine that shares the resources of a physical machine with others. This saves energy, space, and money, making cloud services more affordable.

Layers of Virtualization

1. **Physical Hardware:** This is the actual computer with its CPU, memory, storage, etc.
2. **Hypervisor:** This is the key component. It's a special piece of software that sits between the physical hardware and the virtual machines. It's like a traffic controller, managing how each virtual machine gets access to the hardware's resources.
3. **Virtual Machines (VMs):** These are the virtual computers created by the hypervisor. Each VM acts like a real computer, with its own operating system and applications.

The Hypervisor: Your Traffic Controller

The hypervisor is crucial for virtualization. Here's what it does:

- Resource Allocation: It divides the physical computer's resources (CPU, memory, etc.) among the virtual machines.
- Isolation: It keeps the virtual machines separate from each other, so if one crashes, it doesn't affect the others.
- Abstraction: It hides the complexity of the underlying hardware from the virtual machines, making them easier to manage.

Multitenancy is a core concept in cloud computing that allows multiple customers (tenants) to share the same computing resources, such as servers, storage, and applications. It's like an apartment building where many families live in separate apartments but share the same building structure, plumbing, and electrical systems.

How it Works

- **Shared Resources:** In a multitenant environment, multiple customers share the same underlying infrastructure, including hardware, software, and network resources.
- **Logical Isolation:** Even though they share resources, each tenant's data and applications are logically isolated from others. This means that one tenant cannot access or interfere with another tenant's data.
- **Centralized Management:** The cloud provider manages the infrastructure and ensures that each tenant has a secure and reliable experience.

Benefits of Multitenancy

- **Cost Efficiency:** By sharing resources, cloud providers can reduce costs, which are then passed on to customers in the form of lower prices.
- **Scalability:** Multitenancy enables easy scaling of resources. If a tenant needs more resources, they can quickly access them without affecting other tenants.
- **Efficiency:** Cloud providers can optimize resource utilization by dynamically allocating resources to tenants based on their needs.
- **Simplified Management:** Cloud providers handle the maintenance and updates of the infrastructure, freeing up tenants to focus on their core business.

Examples of Multitenancy

- **Email Services:** Services like Gmail and Outlook.com are multitenant. Millions of users share the same infrastructure, but each user's emails are kept separate and secure.
- **Streaming Services:** Netflix and Amazon Prime Video use multitenancy to deliver content to millions of users simultaneously.

Security Considerations

- **Data Isolation:** Cloud providers must ensure that tenant data is completely isolated from each other to prevent unauthorized access.
- **Security Measures:** Robust security measures, such as access controls, encryption, and intrusion detection systems, are essential to protect tenant data.

AmAmazon EC2 (Elastic Compute Cloud) gives you virtual computers in the cloud, with different types designed for different needs. Let me break this down:

EC2 instances are like renting different types of computers in Amazon's data centers. You can choose the right "computer" based on what your software needs.

Hardware Options

- Different processor brands (Intel, AMD, NVIDIA, and Amazon's own chips) power these instances
- You can choose what works best for your software and budget
- Some options cost less, others perform better for specific tasks

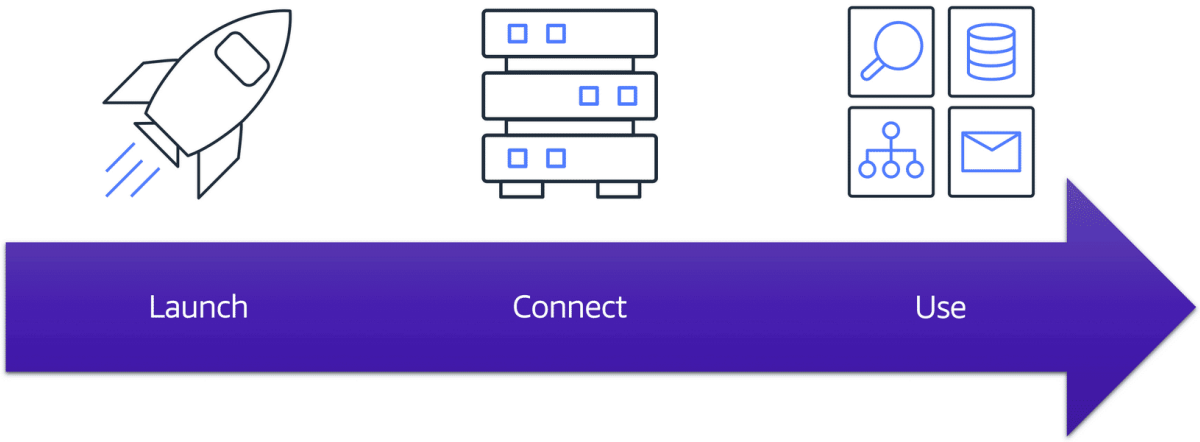
Extra Performance Features

- Local Storage Options: Some instances come with built-in storage disks for faster data access
- Enhanced Networking: Special network connections that are faster for data-intensive applications
- Bare Metal Instances (Own hypervisor): Special instances where you get the whole physical server to yourself without any virtualization layer in between

The bare metal option lets your software talk directly to the actual computer hardware. This is useful if:

- You want to run your own virtualization software
- Your application needs direct access to the physical hardware
- You need every bit of performance without any overhead

How Amazon EC2 works



Launch

First, you launch an instance. Begin by selecting a template with basic configurations for your instance. These configurations include the operating system, application server, or applications. You also select the instance type, which is the specific hardware configuration of your instance.

As you are preparing to launch an instance, you specify security settings to control the network traffic that can flow into and out of your instance. Later in this course, we will explore Amazon EC2 security features in greater detail.

Connect

Next, connect to the instance. You can connect to the instance in several ways. Your programs and applications have multiple different methods to connect directly to the instance and exchange data. Users can also connect to the instance by logging in and accessing the computer desktop.

Use

After you have connected to the instance, you can begin using it. You can run commands to install software, add storage, copy and organize files, and more.

Amazon EC2 instance types

General Purpose Instances

Think of these as versatile family sedans that handle everyday tasks well. They balance processing power, memory, and networking capabilities.

- Running your company's website that handles moderate traffic
- Hosting a small business database
- Running game servers for casual gaming communities
- Managing backend systems for typical business applications

Compute Optimized Instances

These are like sports cars - built for speed and performance. They excel at processing tasks quickly.

- High-traffic website servers during sales events
- Video encoding services that process many videos simultaneously
- Dedicated servers for competitive online gaming
- Batch processing systems like payroll processing that needs to crunch lots of calculations at once

Memory Optimized Instances

Imagine these as trucks with enormous cargo space. They're designed to hold and process large amounts of data at once.

- Large databases that need to keep information readily available
- Real-time analytics platforms analyzing customer behavior
- In-memory caching systems for faster application responses
- Big data processing applications that analyze large datasets

Accelerated Computing Instances

These are specialized vehicles with custom modifications for specific tasks. They have special hardware to speed up particular operations.

- Graphics rendering for animation studios
- Machine learning systems for image recognition
- Game streaming services where games run on the server
- Scientific simulations that require complex calculations

Storage Optimized Instances

These are like warehouse facilities with high-speed conveyor systems. They're built to store and retrieve large amounts of data quickly.

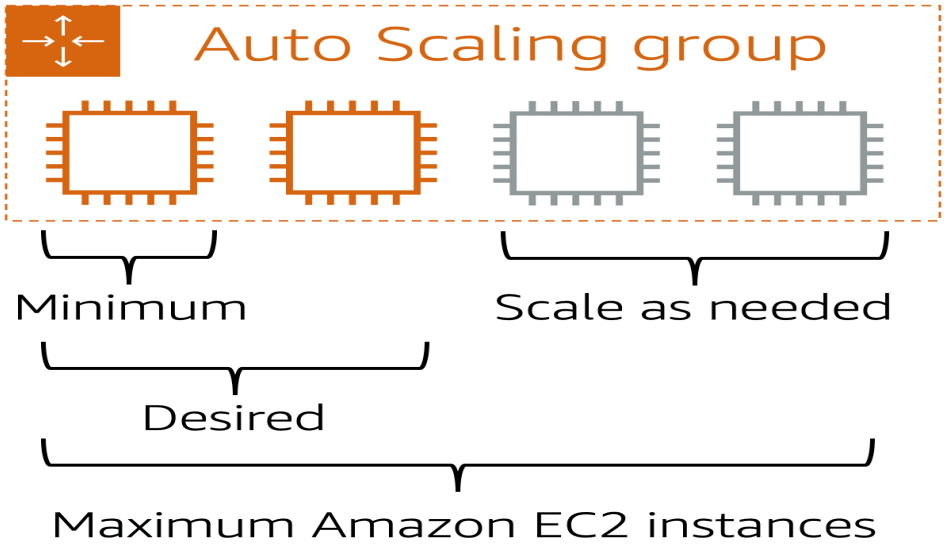
- Large file storage systems like Dropbox or Google Drive
- Database systems that handle many simultaneous transactions (like banking)
- Data warehousing for business intelligence
- E-commerce platforms that need to process many orders simultaneously

The key is matching your needs to the right instance type, just like you'd choose different vehicles for different purposes - a moving truck for relocating, a sports car for a race, or a family sedan for daily commuting.

Scaling Amazon EC2

In the cloud, computing power is a programmatic resource, so you can take a more flexible approach to the issue of scaling. By adding Amazon EC2 Auto Scaling to an application, you can add new instances to the application when necessary and terminate them when no longer needed.

Suppose that you are preparing to launch an application on Amazon EC2 instances. When configuring the size of your Auto Scaling group, you might set the minimum number of Amazon EC2 instances at one. This means that at all times, there must be at least one Amazon EC2 instance running.



When you create an Auto Scaling group, you can set the minimum number of Amazon EC2 instances. The minimum capacity is the number of Amazon EC2 instances that launch immediately after you have created the Auto Scaling group. In this example, the Auto Scaling group has a minimum capacity of one Amazon EC2 instance.

Next, you can set the desired capacity at two Amazon EC2 instances even though your application needs a minimum of a single Amazon EC2 instance to run.

Note: If you do not specify the desired number of Amazon EC2 instances in an Auto Scaling group, the desired capacity defaults to your minimum capacity.

The third configuration that you can set in an Auto Scaling group is the maximum capacity. For example, you might configure the Auto Scaling group to scale out in response to increased demand, but only to a maximum of four Amazon EC2 instances.

Because Amazon EC2 Auto Scaling uses Amazon EC2 instances, you pay for only the instances you use, when you use them. You now have a cost-effective architecture that provides the best customer experience while reducing expenses.

Vertical Scaling in AWS (Upgrading your EC2 instance):

- **What it is:** Imagine you have a small EC2 instance (your virtual server in the cloud) running your website. If it starts to get slow, you can vertically scale it by simply changing it to a larger instance type. This is like going from a "t2.micro" (small) to a "t2.medium" (medium) instance. You're essentially giving your server more CPU power, memory, and sometimes even faster network capabilities.

- **How it works:** AWS makes this easy. You can do it through the AWS Management Console, or even automate it with tools like AWS CLI (command line interface). You essentially stop your instance, choose the new, larger instance type, and start it again.

- **AWS Services involved:**
    - **Amazon EC2:** This is the core service for virtual servers. You choose your instance type here.
    - **AWS CLI/SDKs:** These tools let you automate the scaling process.

Horizontal Scaling in AWS (Adding more EC2 instances):

- **What it is:** Now, imagine your website gets *really* popular. One big server might not be enough. With horizontal scaling, you launch more EC2 instances, all running the same application. They work together to handle the increased traffic. This is like adding more "t2.micro" instances instead of just making one huge "t2.large".

- **How it works:** This is where things get a bit more sophisticated. You'll typically use these AWS services:
    - **Amazon EC2:** Again, the core for your servers.
    - **Auto Scaling:** This service automatically launches or terminates EC2 instances based on demand. You set rules like "if CPU usage goes above 80%, add another instance".
    - **Elastic Load Balancing (ELB):** This distributes traffic across all your EC2 instances, so no single server gets overloaded. It's like a traffic director for your website.

- **AWS Services involved:**
    - **Amazon EC2:** Your virtual servers.
    - **Auto Scaling:** To automate adding/removing instances.
    - **Elastic Load Balancing:** To distribute traffic.

Elastic Load Balancer

Elastic Load Balancing is the AWS service that automatically distributes incoming application traffic across multiple resources, such as Amazon EC2 instances.

A load balancer acts as a single point of contact for all incoming web traffic to your Auto Scaling group. This means that as you add or remove Amazon EC2 instances in response to the amount of incoming traffic, these requests route to the load balancer first. Then, the requests spread across multiple resources that will handle them. For example, if you have multiple Amazon EC2 instances, Elastic Load Balancing distributes the workload across the multiple instances so that no single instance has to carry the bulk of it.

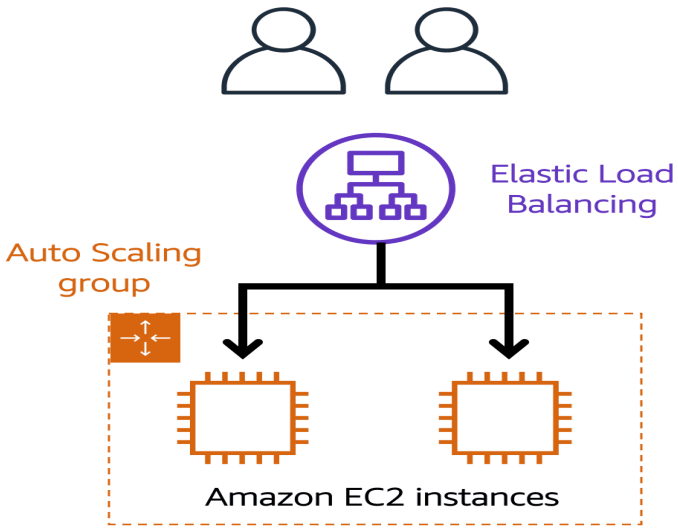
Although Elastic Load Balancing and Amazon EC2 Auto Scaling are separate services, they work together to help ensure that applications running in Amazon EC2 can provide high performance and availability.

Example: Elastic Load Balancing

Low-demand period

Here's an example of how Elastic Load Balancing works. Suppose that a few customers have come to the coffee shop and are ready to place their orders.

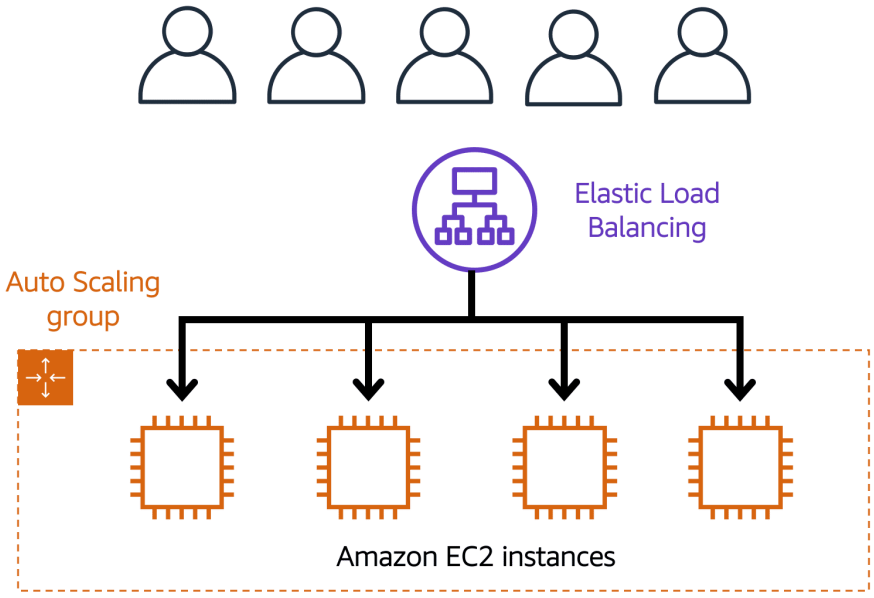
If only a few registers are open, this matches the demand of customers who need service. The coffee shop is less likely to have open registers with no customers. In this example, you can think of the registers as Amazon EC2 instances.



High-demand period

Throughout the day, as the number of customers increases, the coffee shop opens more registers to accommodate them. In the diagram, the Auto Scaling group represents this.

Additionally, a coffee shop employee directs customers to the most appropriate register so that the number of requests can evenly distribute across the open registers. You can think of this coffee shop employee as a load balancer.



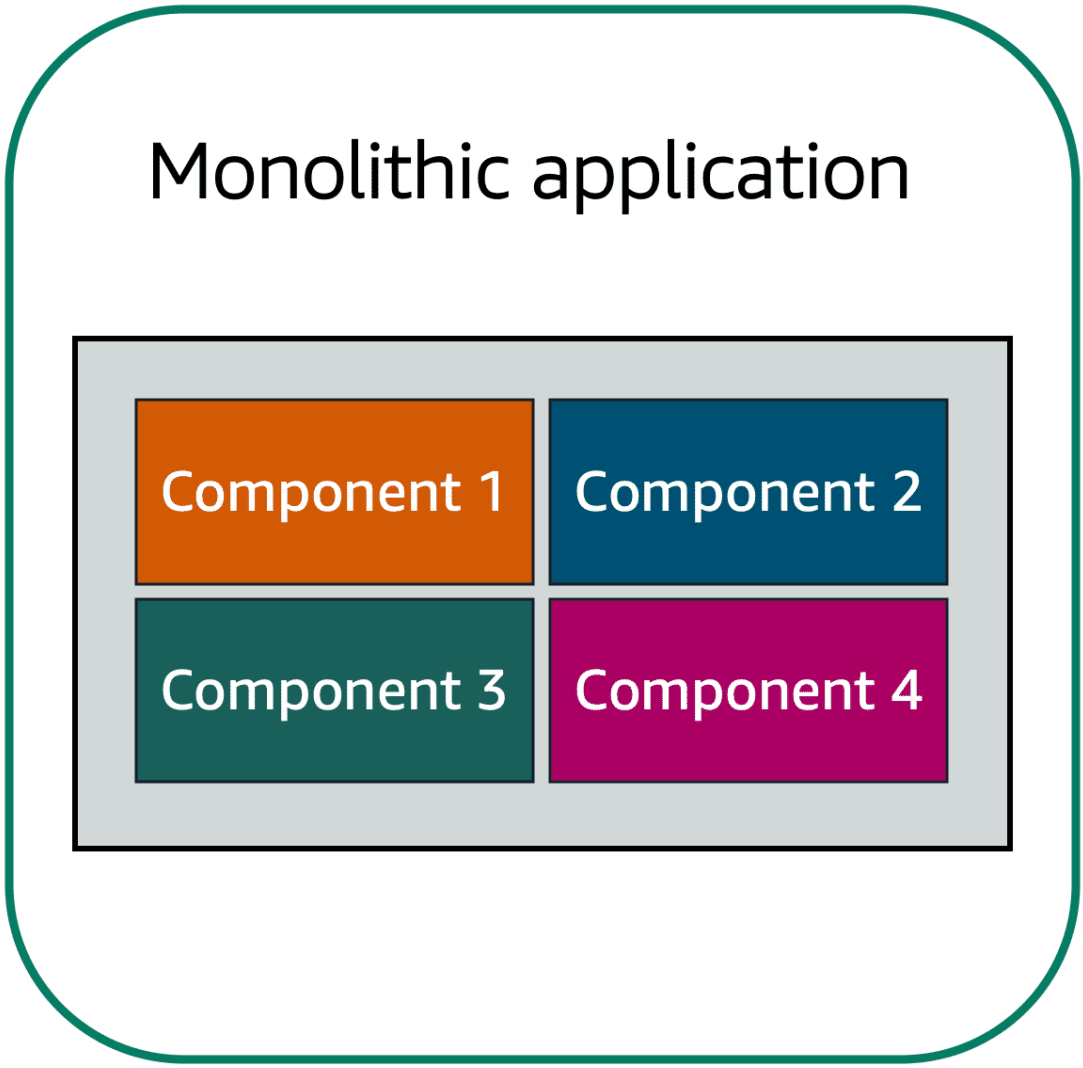
Monolithic Applications and Microservices

Monolithic Applications

Applications are made of multiple components. The components communicate with each other to transmit data, fulfill requests, and keep the application running.

Suppose that you have an application with tightly coupled components. These components might include databases, servers, the user interface, business logic, and so on. This type of architecture can be considered a monolithic application.

In this approach to application architecture, if a single component fails, other components fail, and possibly the entire application fails.



Microservices

To help maintain application availability when a single component fails, you can design your application through a microservices approach.

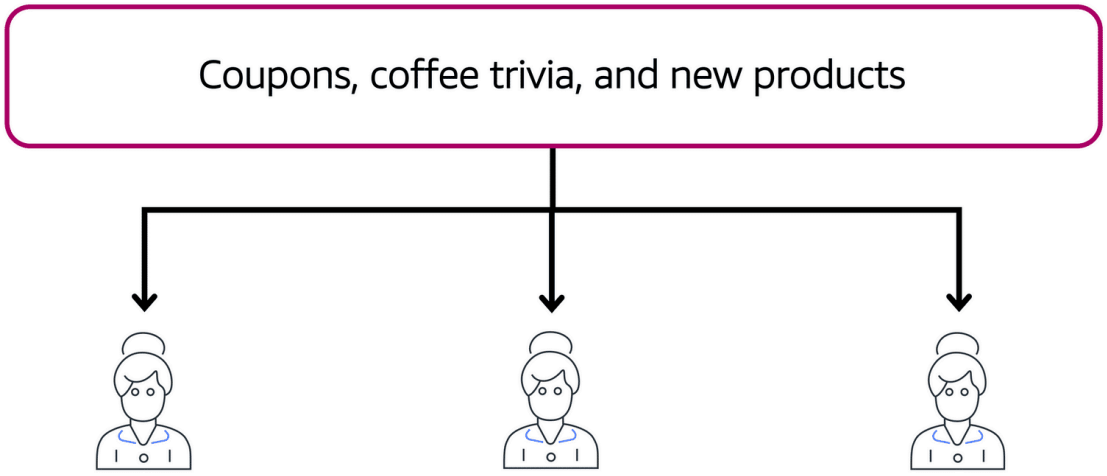
In a microservices approach, application components are loosely coupled. In this case, if a single component fails, the other components continue to work because they are communicating with each other. The loose coupling prevents the entire application from failing.

When designing applications on AWS, you can take a microservices approach with services and components that fulfill different functions. Two services facilitate application integration: Amazon Simple Notification Service (Amazon SNS) and Amazon Simple Queue Service (Amazon SQS).

Amazon Simple Notification Service (Amazon SNS)

Amazon Simple Notification Service (Amazon SNS) is a publish/subscribe service. Using Amazon SNS topics, a publisher publishes messages to subscribers. This is similar to the coffee shop; the cashier provides coffee orders to the barista who makes the drinks. In Amazon SNS, subscribers can be web servers, email addresses, AWS Lambda functions, or several other options.

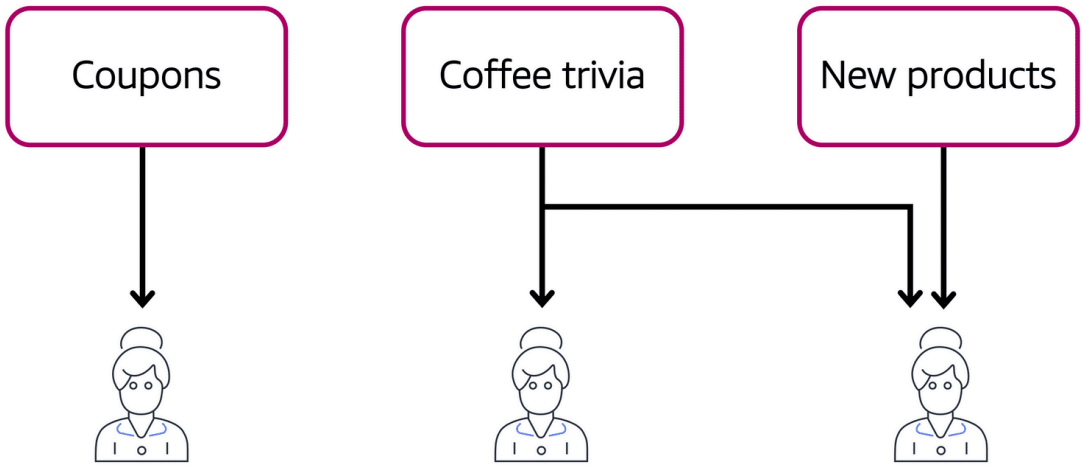
Publishing updates from a single topic



Suppose that the coffee shop has a single newsletter that includes updates from all areas of its business. It includes topics such as coupons, coffee trivia, and new products. All of these topics are grouped because this is a single newsletter. All customers who subscribe to the newsletter receive updates about coupons, coffee trivia, and new products.

After a while, some customers express that they would prefer to receive separate newsletters for only the specific topics that interest them. The coffee shop owners decide to try this approach.

Publishing updates from multiple topics



Now, instead of having a single newsletter for all topics, the coffee shop has broken it up into three separate newsletters. Each newsletter is devoted to a specific topic: coupons, coffee trivia, and new products.

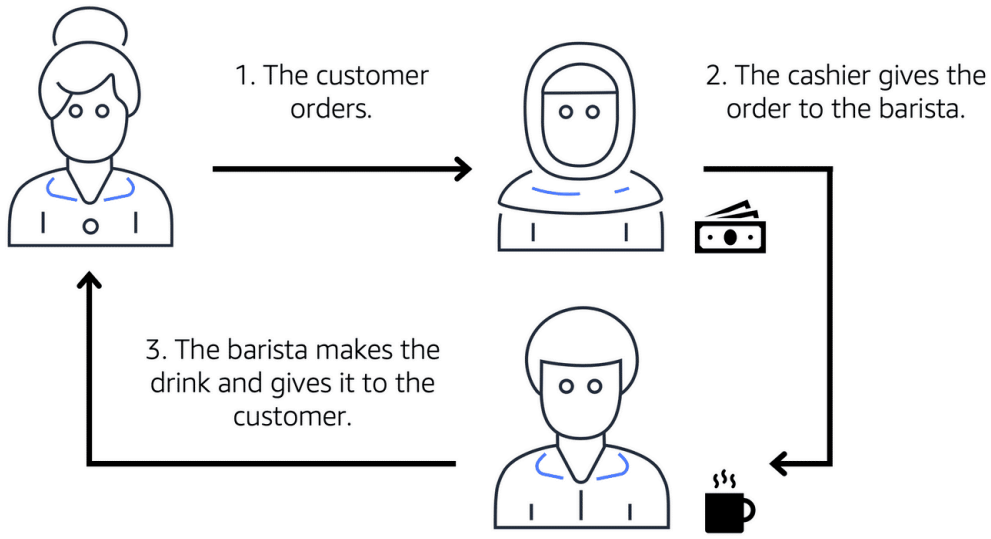
Subscribers will now receive updates immediately for only the specific topics to which they have subscribed.

It is possible for subscribers to subscribe to a single topic or to multiple topics. For example, the first customer subscribes to only the coupons topic, and the second subscriber subscribes to only the coffee trivia topic. The third customer subscribes to both the coffee trivia and new products topics.

Amazon Simple Queue Service (Amazon SQS)

Amazon Simple Queue Service (Amazon SQS) is a message queuing service. Using **Amazon SQS, you can send, store, and receive messages between software components, without losing messages or requiring other services to be available**. In Amazon SQS, an application sends messages into a queue. A user or service retrieves a message from the queue, processes it, and then deletes it from the queue.

Example: Fulfilling an order



Suppose that the coffee shop has an ordering process in which a cashier takes orders, and a barista makes the orders. Think of the cashier and the barista as two separate components of an application.

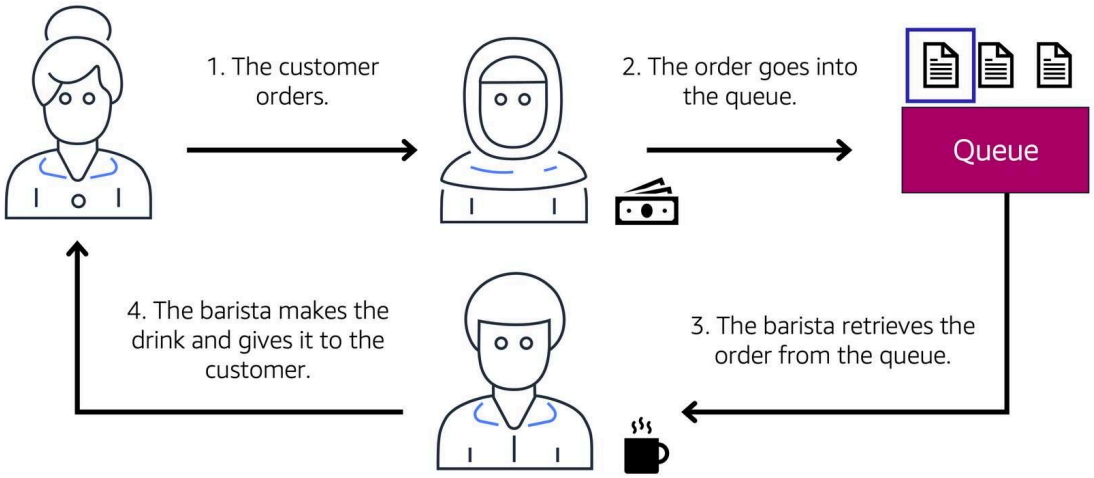
First, the cashier takes an order and writes it down on a piece of paper. Next, the cashier delivers the paper to the barista. Finally, the barista makes the drink and gives it to the customer.

When the next order comes in, the process repeats. This process runs smoothly as long as both the cashier and the barista are coordinated.

What might happen if the cashier took an order and went to deliver it to the barista, but the barista was out on a break or busy with another order? The cashier would need to wait until the barista is ready to accept the order. This would cause delays in the ordering process and require customers to wait longer to receive their orders.

As the coffee shop has become more popular and the ordering line is moving more slowly, the owners notice that the current ordering process is time consuming and inefficient. They decide to try a different approach that uses a queue.

Example: Orders in a queue



Recall that the cashier and the barista are two separate components of an application. A message queuing service such as Amazon SQS enables messages between decoupled application components.

In this example, the first step in the process remains the same as before: a customer places an order with the cashier.

The cashier puts the order into a queue. You can think of this as an order board that serves as a buffer between the cashier and the barista. Even if the barista is out on a break or busy with another order, the cashier can continue placing new orders into the queue.

Next, the barista checks the queue and retrieves the order.

The barista prepares the drink and gives it to the customer.

The barista then removes the completed order from the queue.

While the barista is preparing the drink, the cashier is able to continue taking new orders and add them to the queue.

Tight Coupling: In a tightly coupled architecture, components are highly dependent on each other. They communicate directly and are strongly interconnected. While this can offer faster communication, it has several disadvantages:

1. Lower fault tolerance - If one component fails, it often causes cascading failures
2. Limited scalability - Components must scale together, even if only one needs more resources
3. Harder to modify - Changes to one component often require changes to others
4. Less flexibility - Components are bound to specific implementations

Example of tight coupling in AWS:

- An EC2 instance directly calling another EC2 instance using its IP address
- Applications directly communicating with each other without any intermediary
- Hard-coded dependencies between services

Loose Coupling: In a loosely coupled architecture, components are more independent and interact through well-defined interfaces. AWS promotes loose coupling through various services:

1. Simple Queue Service (SQS)
- Acts as a buffer between components
- Enables asynchronous communication
- Components don't need to be available simultaneously

2. Simple Notification Service (SNS)
- Pub/sub messaging for one-to-many communication
- Services can subscribe to relevant topics without knowing publishers

3. API Gateway
- Creates standardized interfaces between services
- Handles authentication and traffic management
- Services can evolve independently

4. Elastic Load Balancer (ELB)
- Distributes traffic across multiple instances
- Services don't need to know instance details
- Instances can be added/removed without affecting clients

Decoupling:

- Refers to the complete separation of components
- Components are entirely independent and unaware of each other
- Communication happens through intermediary systems like message queues
- Example: Two services communicating through SQS, where the sender doesn't know or care about the receiver

What is pub/sub messaging?

Publish-subscribe messaging, or pub/sub messaging, is an asynchronous communication model that makes it easy for developers to build highly functional and architecturally complex applications in the cloud. In modern cloud architecture, applications are decoupled into smaller, independent building blocks called services. Pub/sub messaging provides instant event notifications for these distributed systems. It supports scalable and reliable communication between independent software modules.

How does pub/sub messaging work?

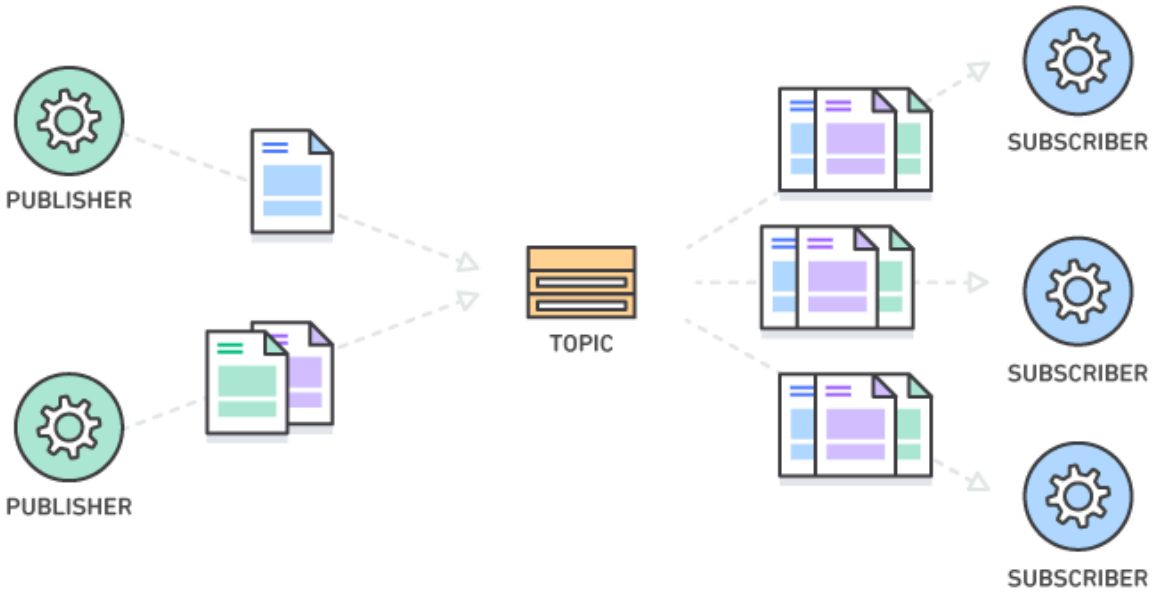
The publish-subscribe (pub/sub) system has four key components.

Messages - A message is communication data sent from sender to receiver. Message data types can be anything from strings to complex objects representing text, video, sensor data, audio, or other digital content.

Topics - Every message has a topic associated with it. The topic acts like an intermediary channel between senders and receivers. It maintains a list of receivers who are interested in messages about that topic.

Subscribers - A subscriber is the message recipient. Subscribers have to register (or subscribe) to topics of interest. They can perform different functions or do something different with the message in parallel.

Publishers - The publisher is the component that sends messages. It creates messages about a topic and sends them once only to all subscribers of that topic. This interaction between the publisher and subscribers is a one-to-many relationship. The publisher doesn’t need to know who is using the information it is broadcasting, and the subscribers don’t need to know where the message comes from.



What is the difference between message queues and pub/sub messaging?

A message queue is another form of asynchronous communication used in serverless and microservices architectures. Messages are stored in the queue until they are processed and deleted. Message queues require the sender to know who they are exchanging messages with. Message ordering may also cause bottlenecks in the system.

In contrast, the publish-subscribe (pub/sub) pattern allows for more flexibility. Several interested subscribers can receive messages simultaneously and asynchronously. Publishers don't need to know who the subscribers are. Message handling is more scalable and reliable, and it gives better performance.

WebHook

A webhook is a way for apps to automatically share information with each other in real time.

Imagine this:

- You have a weather app and a smart umbrella.
- Instead of the weather app constantly asking "Is it raining?" (like calling an API), it can send a message to the umbrella saying "It's going to rain!" when the forecast changes.
- The umbrella can then automatically open itself.

That message is a webhook. It's a way for one app to "hook" into another and get notified when something interesting happens.

Here's why it's useful:

- Saves time and resources: Apps don't have to constantly check for updates.
- Real-time updates: Get information as soon as it happens.
- Automation: Trigger actions in other apps automatically.

Common examples:

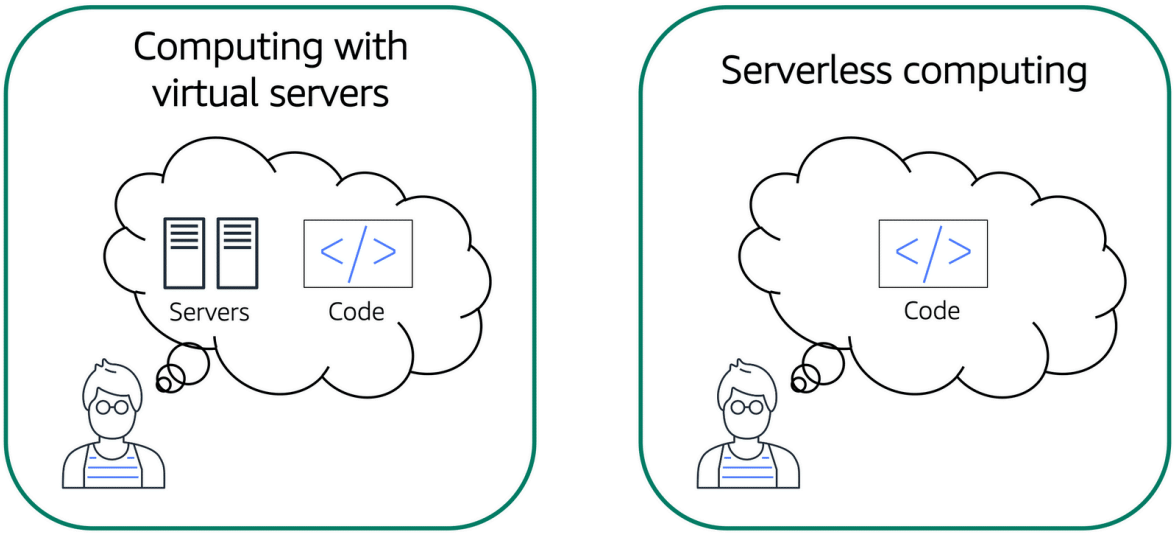
- A payment app notifying your accounting software when a transaction is made.
- A social media platform sending a notification to your phone when you get a new message.
- A project management tool updating your team's chat channel when a task is completed.

In short, webhooks are like automated messengers that help apps communicate and work together more efficiently.

Serverless Computing

Earlier in this module, you learned about Amazon EC2, a service that lets you run virtual servers in the cloud. If you have applications that you want to run in Amazon EC2, you must do the following:

1. Provision instances (virtual servers).
2. Upload your code.
3. Continue to manage the instances while your application is running.



The term “serverless” means that your code runs on servers, but you do not need to provision or manage these servers. With serverless computing, you can focus more on innovating new products and features instead of maintaining servers.

Another benefit of serverless computing is the flexibility to scale serverless applications automatically. Serverless computing can adjust the applications' capacity by modifying the units of consumptions, such as throughput and memory.

An AWS service for serverless computing is AWS Lambda.

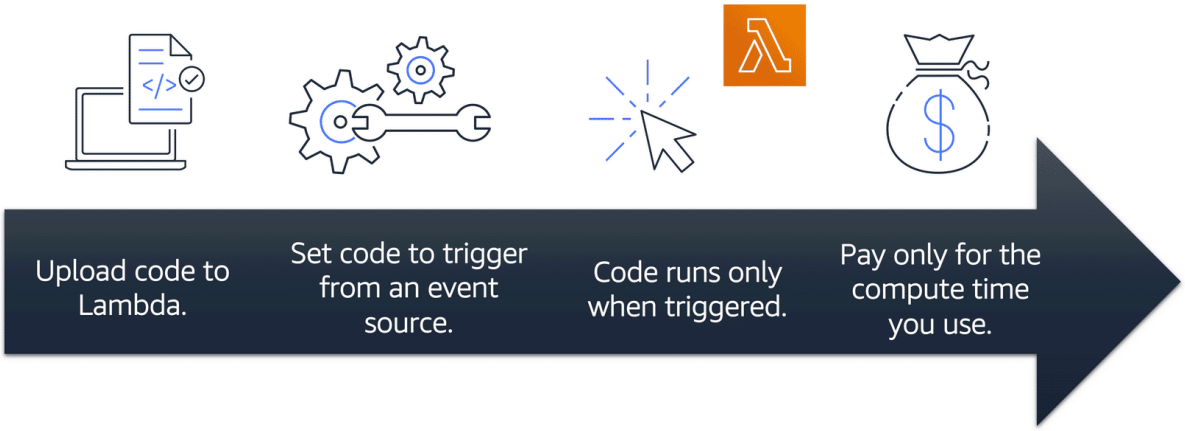
AWS Lambda

AWS Lambda is a service that lets you run code without needing to provision or manage servers.

While using AWS Lambda, you pay only for the compute time that you consume. Charges apply only when your code is running. You can also run code for virtually any type of application or backend service, all with zero administration.

For example, a simple Lambda function might involve automatically resizing uploaded images to the AWS Cloud. In this case, the function triggers when uploading a new image.

How AWS Lambda works



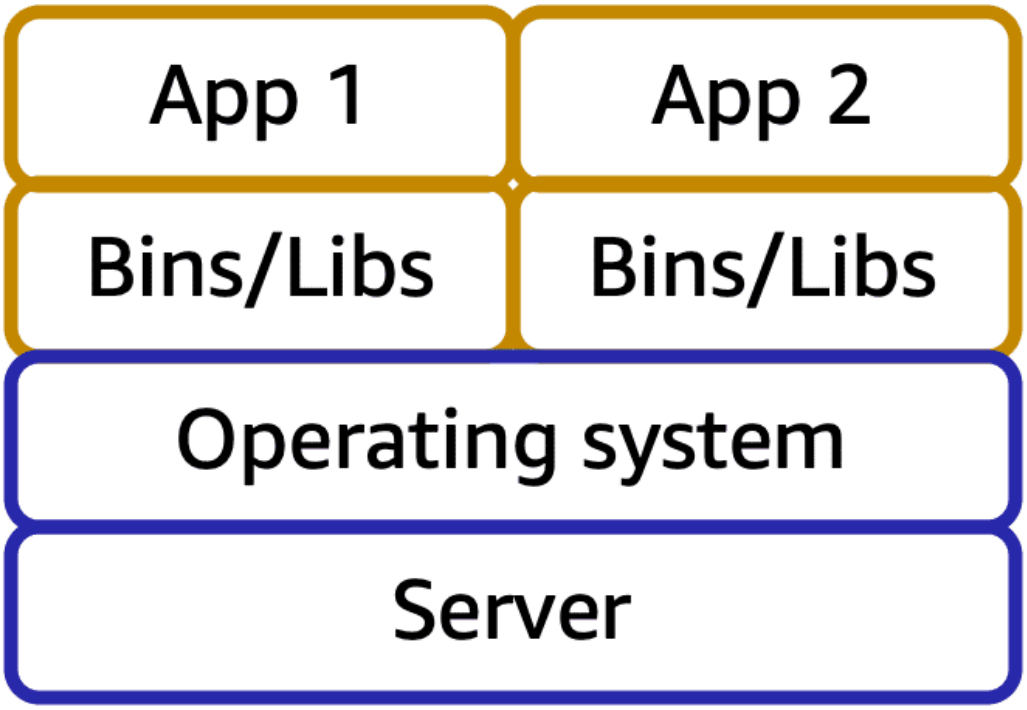
1. You upload your code to Lambda.
2. You set your code to trigger from an event source, such as AWS services, mobile applications, or HTTP endpoints.
3. Lambda runs your code only when triggered.
4. You pay only for the compute time that you use. In the previous example of resizing images, you would pay only for the compute time that you use when uploading new images. Uploading the images triggers Lambda to run code for the image resizing function.

Containers

In AWS, you can also build and run containerized applications.

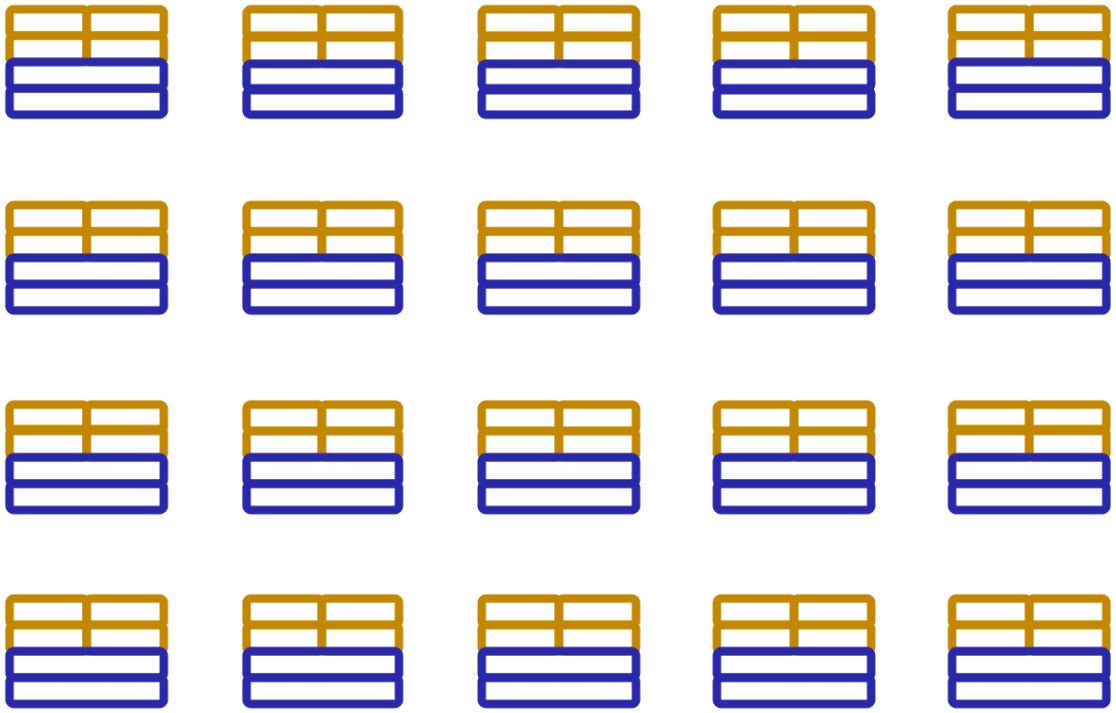
Containers provide you with a standard way to package your application's code and dependencies into a single object. You can also use containers for processes and workflows in which there are essential requirements for security, reliability, and scalability.

Examples: One host with multiple containers



Suppose that a company’s application developer has an environment on their computer that is different from the environment on the computers used by the IT operations staff. The developer wants to ensure that the application’s environment remains consistent regardless of deployment, so they use a containerized approach. This helps to reduce time spent debugging applications and diagnosing differences in computing environments.

Tens of hosts with hundreds of containers



When running containerized applications, it's important to consider scalability. Suppose that instead of a single host with multiple containers, you have to manage tens of hosts with hundreds of containers. Alternatively, you have to manage possibly hundreds of hosts with thousands of containers. At a large scale, imagine how much time it might take for you to monitor memory usage, security, logging, and so on.
Amazon Elastic Container Service (Amazon ECS)

Amazon Elastic Container Service (Amazon ECS) is a highly scalable, high-performance container management system that enables you to run and scale containerized applications on AWS.

Amazon ECS supports Docker containers. Docker is a software platform that enables you to build, test, and deploy applications quickly. AWS supports the use of open-source Docker Community Edition and subscription-based Docker Enterprise Edition. With Amazon ECS, you can use API calls to launch and stop Docker-enabled applications.

Selecting a Region

When determining the right Region for your services, data, and applications, consider the following four business factors.

Compliance with data governance and legal requirements

Depending on your company and location, you might need to run your data out of specific areas. For example, if your company requires all of its data to reside within the boundaries of the UK, you would choose the London Region.

Not all companies have location-specific data regulations, so you might need to focus more on the other three factors.

Proximity to your customers

Selecting a Region that is close to your customers will help you to get content to them faster. For example, your company is based in Washington, DC, and many of your customers live in Singapore. You might consider running your infrastructure in the Northern Virginia Region to be close to company headquarters, and run your applications from the Singapore Region.

Available services within a Region

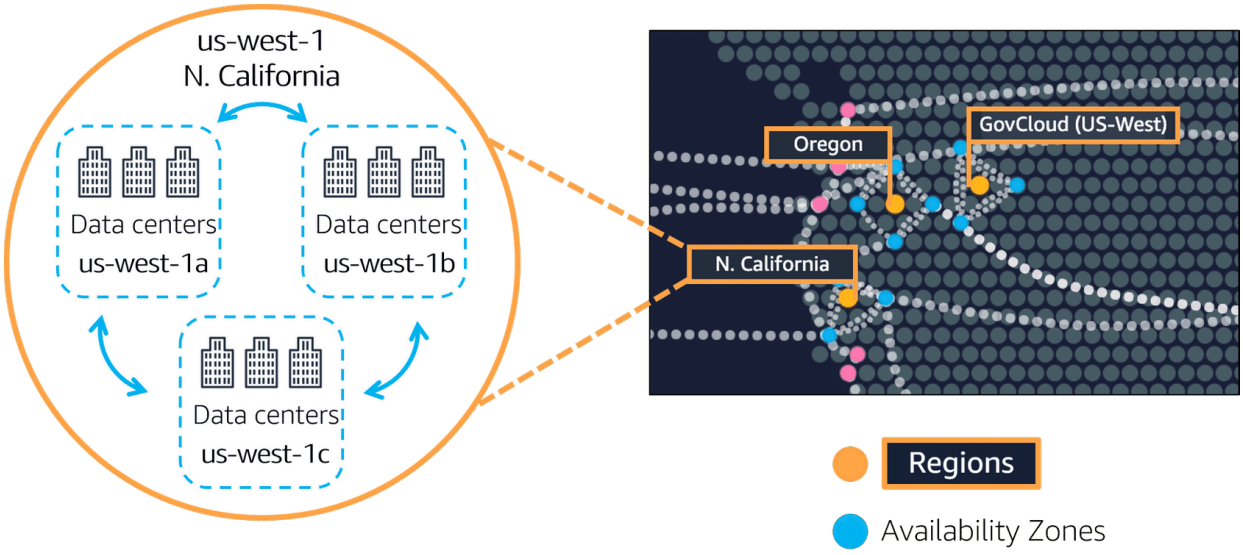
Sometimes, the closest Region might not have all the features that you want to offer to customers. AWS is frequently innovating by creating new services and expanding on features within existing services. However, making new services available around the world sometimes requires AWS to build out physical hardware one Region at a time.

Suppose that your developers want to build an application that uses Amazon Braket (AWS quantum computing platform). As of this course, Amazon Braket is not yet available in every AWS Region around the world, so your developers would have to run it in one of the Regions that already offers it.

Pricing

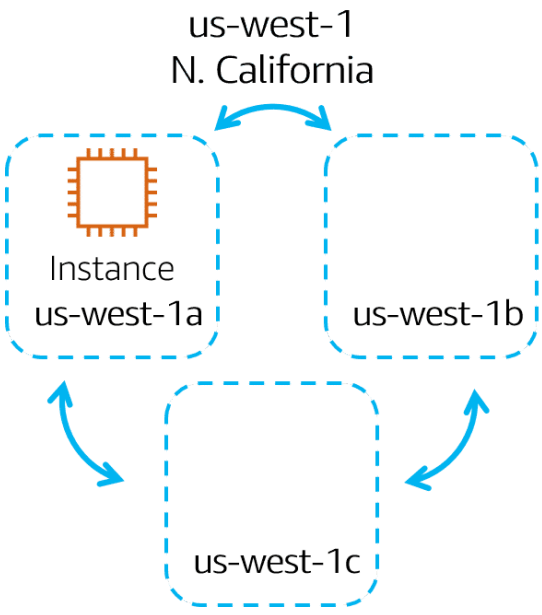
Suppose that you are considering running applications in both the United States and Brazil. The way Brazil's tax structure is set up, it might cost 50% more to run the same workload out of the São Paulo Region compared to the Oregon Region. You will learn in more detail that several factors determine pricing, but for now know that the cost of services can vary from Region to Region.

Availability Zones



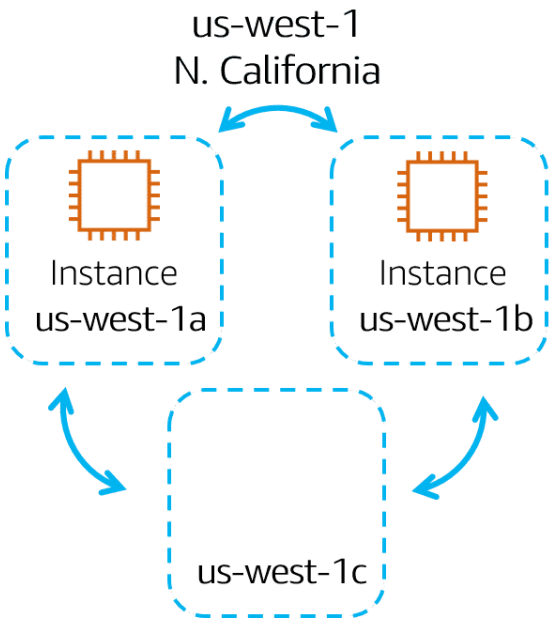
An Availability Zone is a single data center or a group of data centers within a Region. Availability Zones are located tens of miles apart from each other. This is close enough to have low latency (the time between when content requested and received) between Availability Zones. However, if a disaster occurs in one part of the Region, they are distant enough to reduce the chance that multiple Availability Zones are affected. Running Amazon EC2 instances in multiple Availability Zones

Amazon EC2 instance in a single Availability Zone

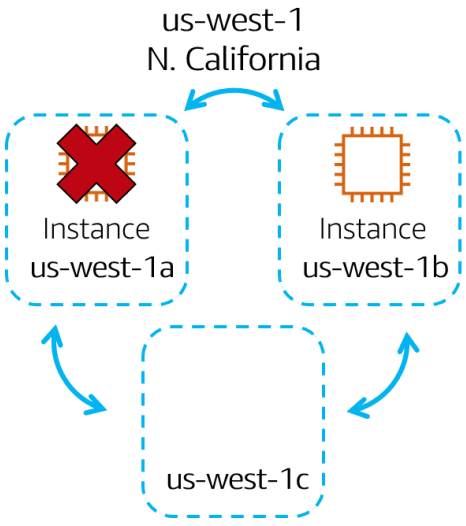


Suppose that you're running an application on a single Amazon EC2 instance in the Northern California Region. The instance is running in the us-west-1a Availability Zone. If us-west-1a were to fail, you would lose your instance.

Amazon EC2 instances in multiple Availability Zones



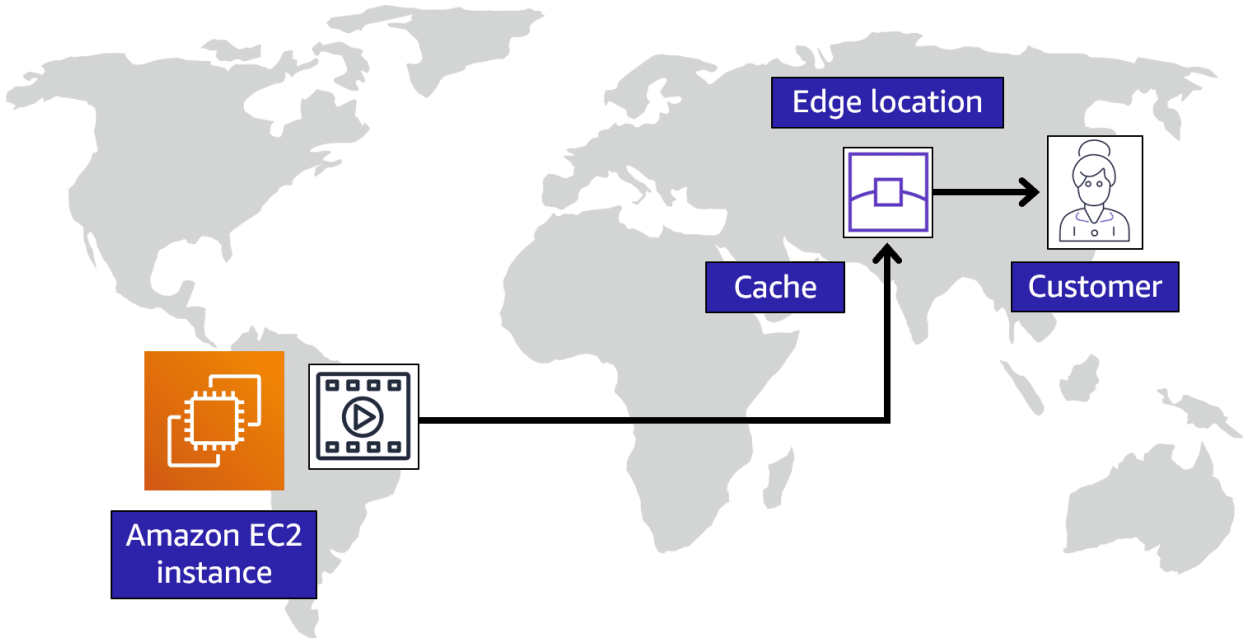
A best practice is to run applications across at least two Availability Zones in a Region. In this example, you might choose to run a second Amazon EC2 instance in us-west-1b. Availability Zone failure



If us-west-1a were to fail, your application would still be running in us-west-1b.

Edge location

An edge location is a site that Amazon CloudFront uses to store cached copies of your content closer to your customers for faster delivery.



Origin

Suppose that your company's data is stored in Brazil, and you have customers who live in China. To provide content to these customers, you don't need to move all the content to one of the Chinese Regions.

Edge Location

Instead of requiring your customers to get their data from Brazil, you can cache a copy locally at an edge location that is close to your customers in China.

Customer

When a customer in China requests one of your files, Amazon CloudFront retrieves the file from the cache in the edge location and delivers the file to the customer. The file is delivered to the customer faster because it came from the edge location near China instead of the original source in Brazil.
Mark as completed

Content Delivery Network (CDN)

A Content Delivery Network (CDN) is a network of servers that delivers web content to users based on their geographic location.[1] A CDN can improve website performance and reduce latency by serving content from a server that is closer to the user.

A CDN works by caching content on servers around the world. When a user requests content from a website, the CDN will serve the content from the server that is closest to the user. This can significantly reduce the time it takes for the content to load.

CDNs can also be used to improve website security. By distributing content across multiple servers, CDNs can make it more difficult for attackers to take down a website.

There are many different CDN providers. **Some of the most popular include Amazon CloudFront, Google Cloud CDN, and Microsoft Azure CDN.**

Here are some of the benefits of using a CDN:

- **Improved website performance**: CDNs can significantly reduce the time it takes for web pages to load.
- **Reduced latency:** CDNs can reduce the delay between when a user requests content and when the content is delivered.
- **Increased website security:** CDNs can help to protect websites from attacks.
- **Improved SEO:** CDNs can help to improve a website's search engine ranking.

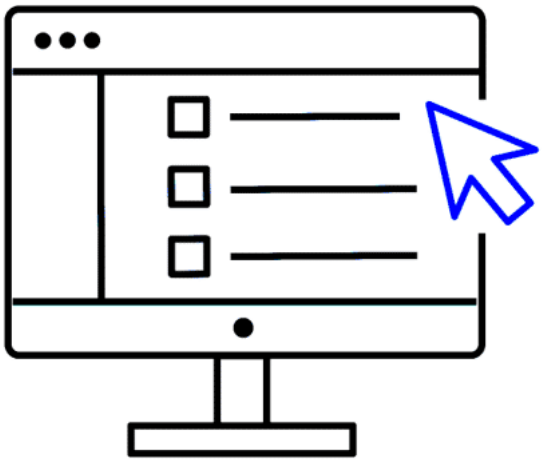
If you are looking to improve the performance and security of your website, a CDN is a good option to consider.

You may be wondering, how do I actually interact with these services? The answer is APIs.

In AWS, everything is an API call. An API is an application programming interface. What that means is, there are predetermined ways for you to interact with AWS services. You can invoke or call these APIs to provision, configure, and manage your AWS resources. For example, you can launch an EC2 instance, or you can create an AWS Lambda function. Each of those would be different requests and different API calls to AWS.

Ways to Interact with AWS Services

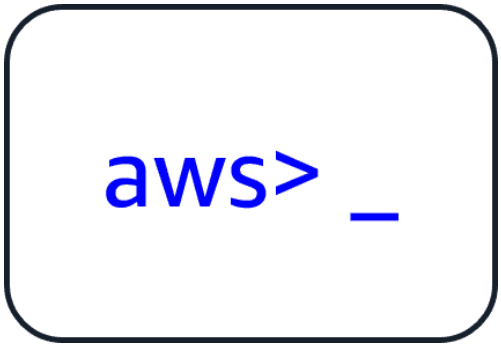
AWS Management Console



The AWS Management Console is a web-based interface for accessing and managing AWS services. You can quickly access recently used services and search for other services by name, keyword, or acronym. The console includes wizards and automated workflows that can simplify the process of completing tasks.

You can also use the AWS Console mobile application to perform tasks such as monitoring resources, viewing alarms, and accessing billing information. Multiple identities can stay logged into the AWS Console mobile app at the same time.

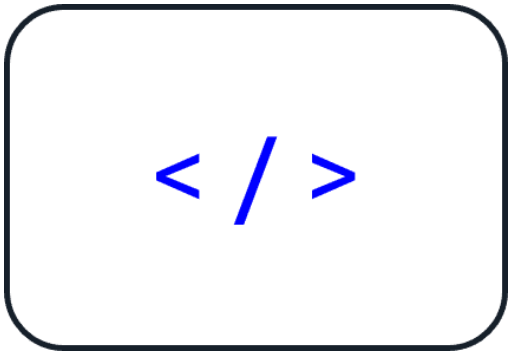
AWS Command Line Interface



To save time when making API requests, you can use the AWS Command Line Interface (AWS CLI). AWS CLI enables you to control multiple AWS services directly from the command line within one tool. AWS CLI is available for users on Windows, macOS, and Linux.

By using AWS CLI, you can automate the actions that your services and applications perform through scripts. For example, you can use commands to launch an Amazon EC2 instance, connect an Amazon EC2 instance to a specific Auto Scaling group, and more.

Software development kits (SDKs)



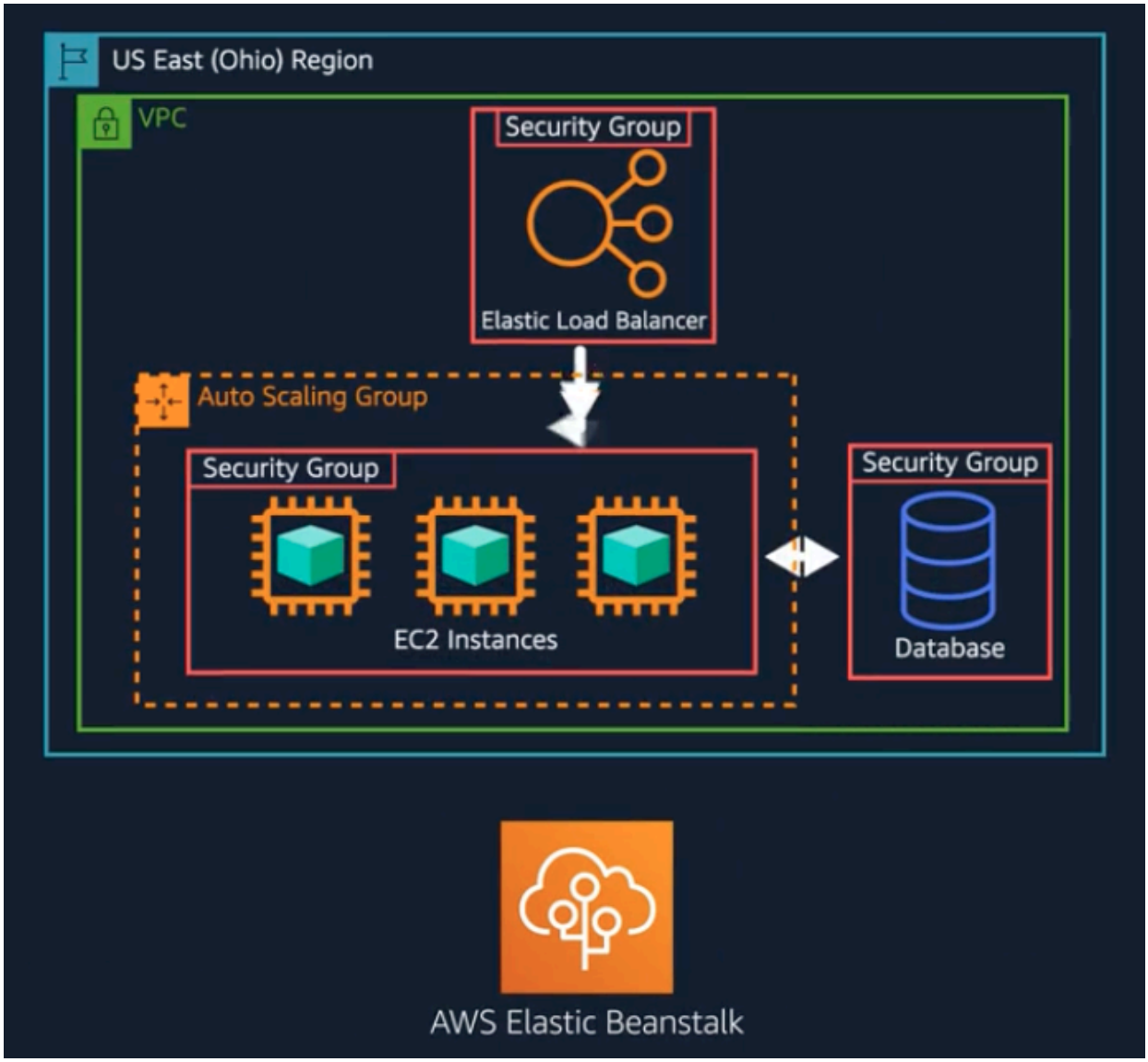
Another option for accessing and managing AWS services is the software development kits (SDKs). SDKs make it easier for you to use AWS services through an API designed for your programming language or platform. SDKs enable you to use AWS services with your existing applications or create entirely new applications that will run on AWS.

To help you get started with using SDKs, AWS provides documentation and sample code for each supported programming language. Supported programming languages include C++, Java, .NET, and more.

AWS Elastic Beanstalk

With AWS Elastic Beanstalk, you provide code and configuration settings, and Elastic Beanstalk deploys the resources necessary to perform the following tasks:

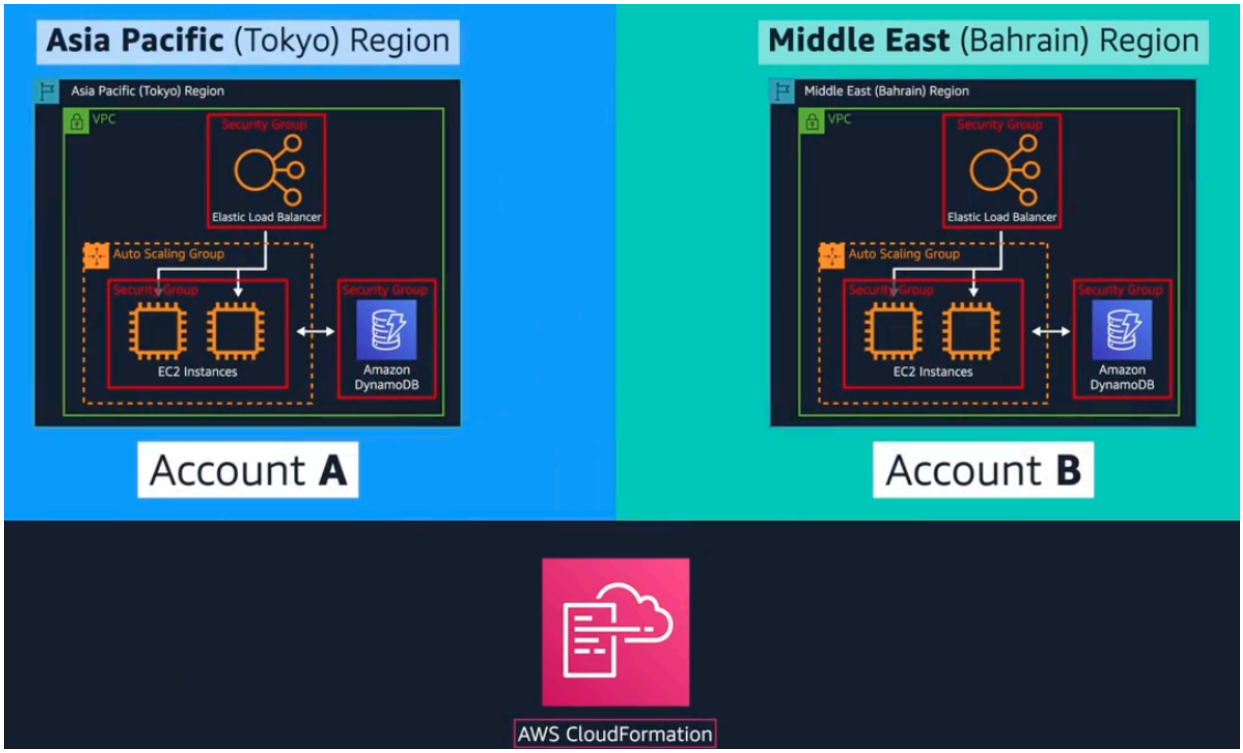
- Adjust capacity
- Load balancing
- Automatic scaling
- Application health monitoring



AWS CloudFormation

With AWS CloudFormation, you can treat your infrastructure as code. This means that you can build an environment by writing lines of code instead of using the AWS Management Console to individually provision resources.

AWS CloudFormation provisions your resources in a safe, repeatable manner, enabling you to frequently build your infrastructure and applications without having to perform manual actions or write custom scripts. It determines the right operations to perform when managing your stack and rolls back changes automatically if it detects errors.



Connectivity to AWS

Amazon Virtual Private Cloud (Amazon VPC)

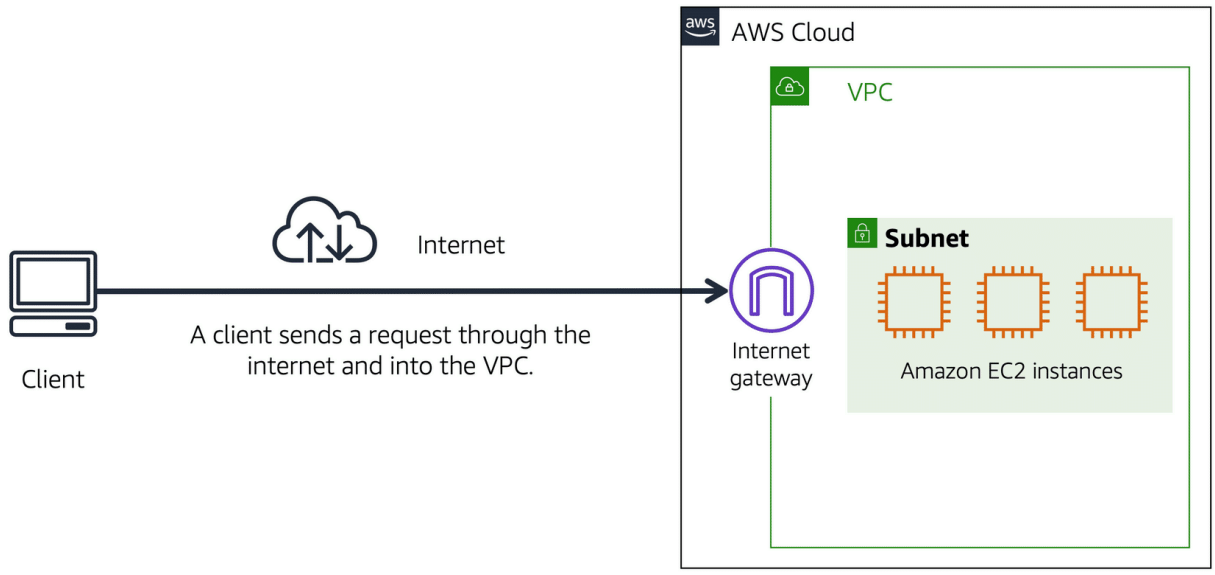
Imagine the millions of customers who use AWS services. Also, imagine the millions of resources that these customers have created, such as Amazon EC2 instances. Without boundaries around all of these resources, network traffic would be able to flow between them unrestricted.

A networking service that you can use to establish boundaries around your AWS resources is Amazon Virtual Private Cloud (Amazon VPC).

Amazon VPC enables you to provision an isolated section of the AWS Cloud. In this isolated section, you can launch resources in a virtual network that you define. Within a virtual private cloud (VPC), you can organize your resources into subnets. A subnet is a section of a VPC that can contain resources such as Amazon EC2 instances.

Internet gateway

To allow public traffic from the internet to access your VPC, you attach an internet gateway to the VPC.



An internet gateway is a connection between a VPC and the internet. You can think of an internet gateway as being similar to a doorway that customers use to enter the coffee shop. Without an internet gateway, no one can access the resources within your VPC.

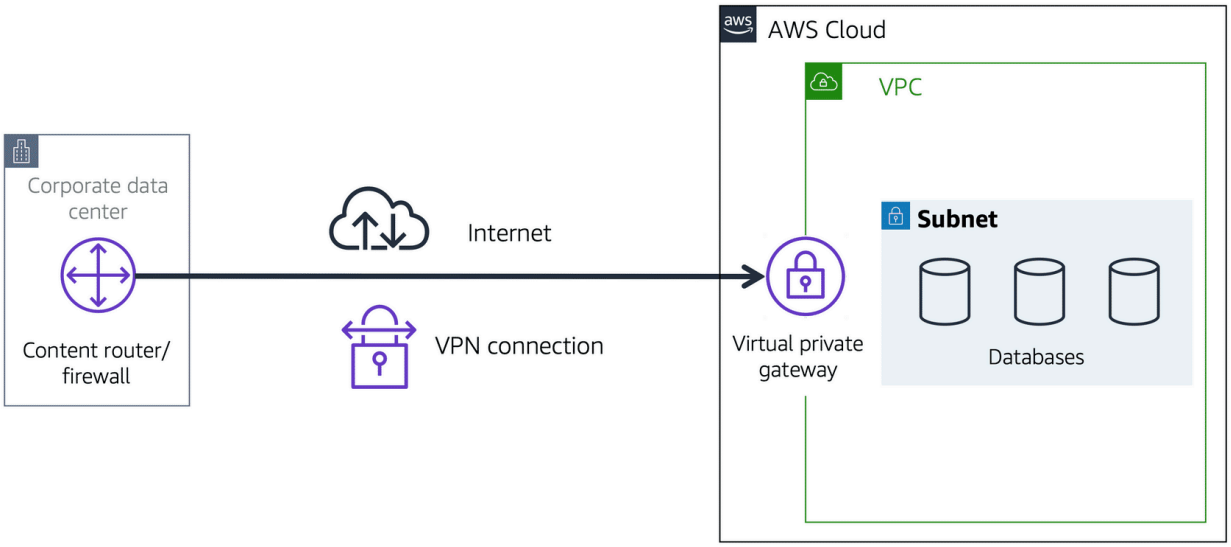
What if you have a VPC that includes only private resources?

Virtual private gateway

Here's an example of how a virtual private gateway works. You can think of the internet as the road between your home and the coffee shop. Suppose that you are traveling on this road with a bodyguard to protect you. You are still using the same road as other customers, but with an extra layer of protection.

The bodyguard is like a virtual private network (VPN) connection that encrypts (or protects) your internet traffic from all the other requests around it.

The virtual private gateway is the component that allows protected internet traffic to enter into the VPC. Even though your connection to the coffee shop has extra protection, traffic jams are possible because you're using the same road as other customers.



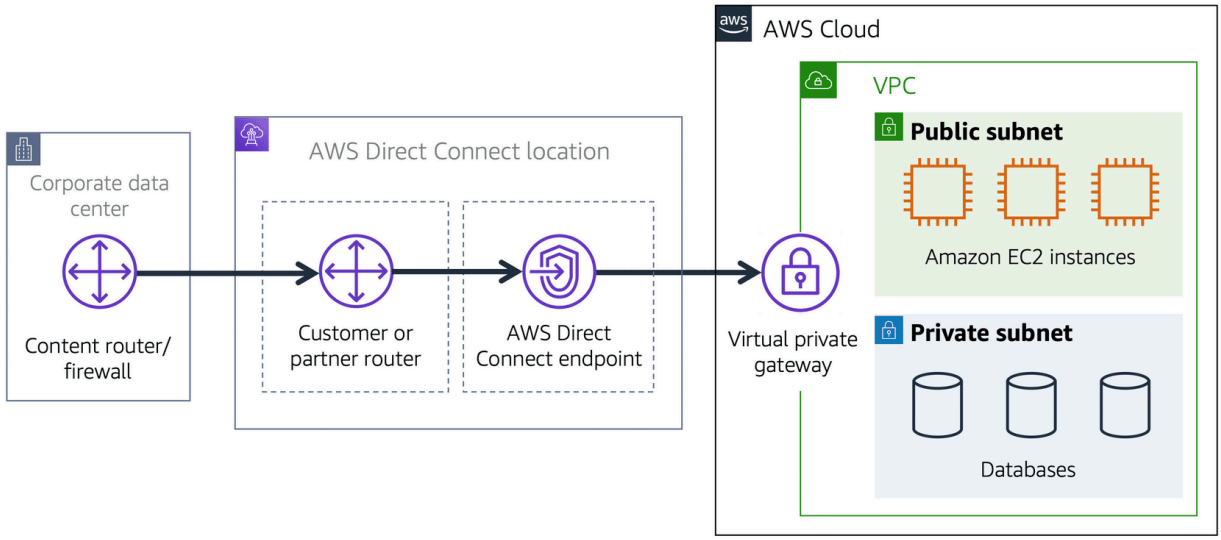
A virtual private gateway enables you to establish a virtual private network (VPN) connection between your VPC and a private network, such as an on-premises data center or internal corporate network. A virtual private gateway allows traffic into the VPC only if it is coming from an approved network.

AWS Direct Connect

AWS Direct Connect is a service that enables you to establish a dedicated private connection between your data center and a VPC.

Suppose that there is an apartment building with a hallway directly linking the building to the coffee shop. Only the residents of the apartment building can travel through this hallway.

This private hallway provides the same type of dedicated connection as AWS Direct Connect. Residents are able to get into the coffee shop without needing to use the public road shared with other customers.



The private connection that AWS Direct Connect provides helps you to reduce network costs and increase the amount of bandwidth that can travel through your network.

A VPN, or Virtual Private Network, is like a secret tunnel for your internet traffic. It creates a secure connection between your device and the internet, protecting your data and privacy.

Here's a simple diagram to illustrate:

[Your Device] --> [Encrypted Tunnel] --> [VPN Server] --> [Internet]

Here's how it works:

1. **You connect to a VPN server:** When you use a VPN, your internet traffic is routed through a server operated by the VPN provider.
2. **Your data is encrypted:** Before leaving your device, your data is encrypted, making it unreadable to anyone who might try to intercept it.
3. **Your IP address is hidden:** Your real IP address, which identifies your device and location, is replaced with the IP address of the VPN server. This makes it appear as if you are browsing from a different location.
4. **Your online activity is protected:** With your data encrypted and your IP address hidden, your online activity becomes much more private and secure.

Benefits of using a VPN:

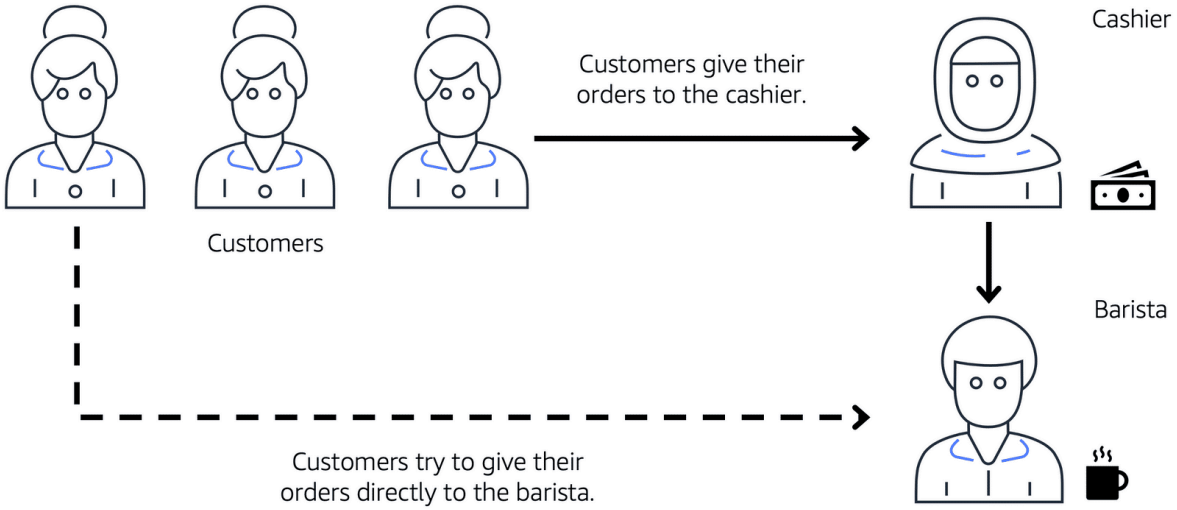
- **Enhanced privacy:** Protect your online activity from prying eyes, including your internet service provider (ISP), government agencies, and hackers.
- **Increased security:** Secure your data on public Wi-Fi networks, which are often vulnerable to hacking.
- **Access to restricted content:** Bypass geographical restrictions and access content that may be blocked in your region.
- **Anonymity:** Hide your IP address and browse the internet anonymously.

Subnets and Network Access Control Lists

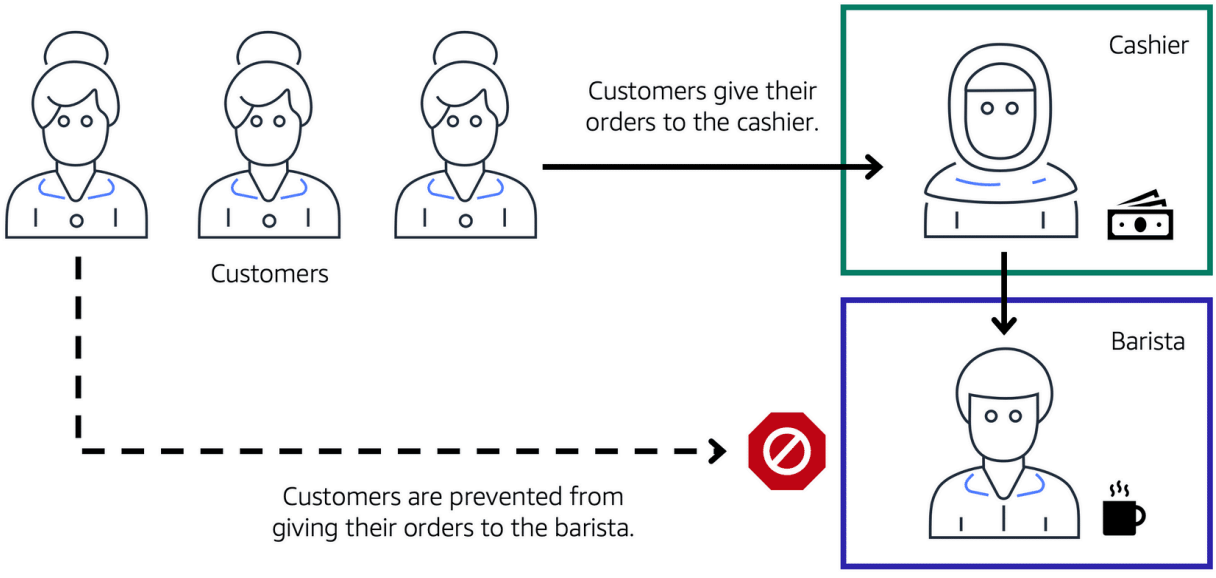
To learn more about the role of subnets within a VPC, review the following example from the coffee shop.

First, customers give their orders to the cashier. The cashier then passes the orders to the barista. This process allows the line to keep running smoothly as more customers come in.

Suppose that some customers try to skip the cashier line and give their orders directly to the barista. This disrupts the flow of traffic and results in customers accessing a part of the coffee shop that is restricted to them.



To fix this, the owners of the coffee shop divide the counter area by placing the cashier and the barista in separate workstations. The cashier's workstation is public facing and designed to receive customers. The barista's area is private. The barista can still receive orders from the cashier but not directly from customers.

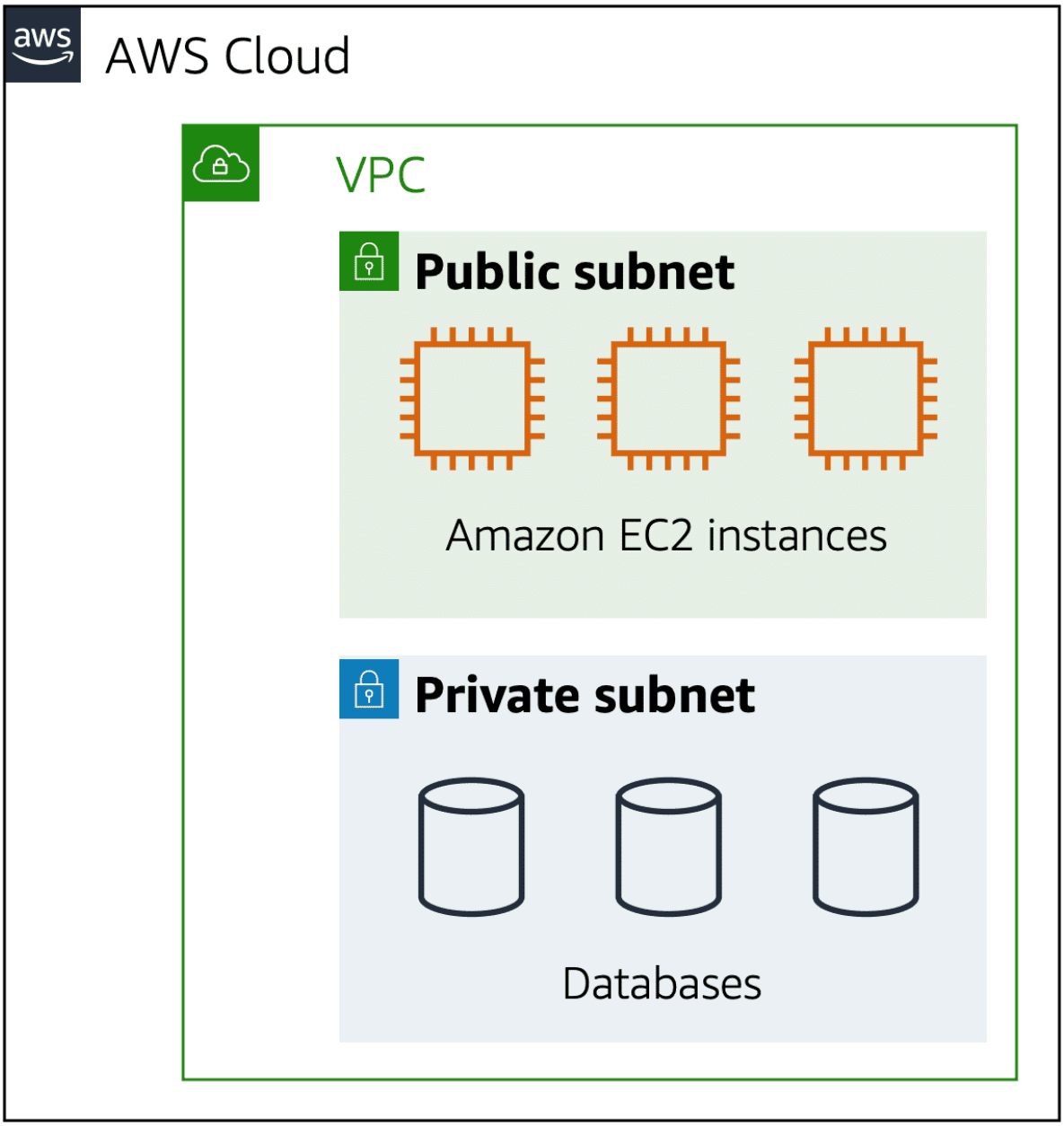


This is similar to how you can use AWS networking services to isolate resources and determine exactly how network traffic flows.

In the coffee shop, you can think of the counter area as a VPC. The counter area divides into two separate areas for the cashier's workstation and the barista's workstation. In a VPC, subnets are separate areas that are used to group together resources.

Subnets

A subnet is a section of a VPC in which you can group resources based on security or operational needs. Subnets can be public or private.



Public subnets contain resources that need to be accessible by the public, such as an online store's website.

Private subnets contain resources that should be accessible only through your private network, such as a database that contains customers' personal information and order histories.

In a VPC, subnets can communicate with each other. For example, you might have an application that involves Amazon EC2 instances in a public subnet communicating with databases that are located in a private subnet.

Network Traffic in a VPC

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.

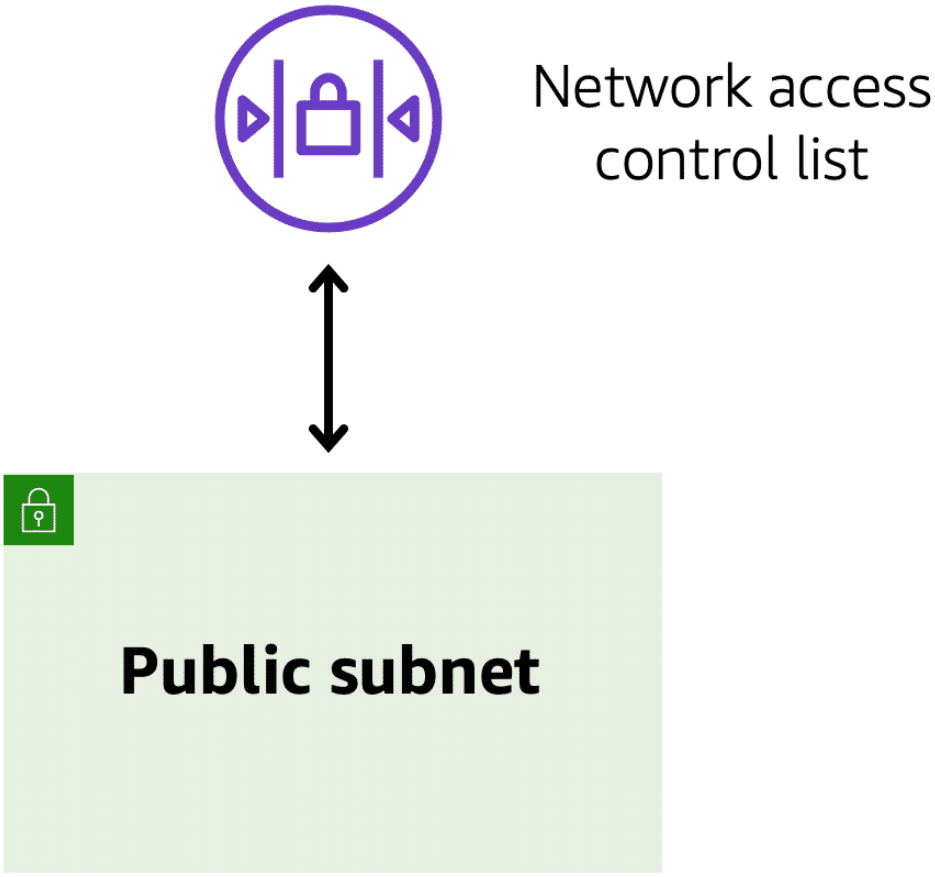
It enters into a VPC through an internet gateway. Before a packet can enter into a subnet or exit from a subnet, it checks for permissions. These permissions indicate who sent the packet and how the packet is trying to communicate with the resources in a subnet.

The VPC component that checks packet permissions for subnets is a network access control list (ACL).

Network Access Control Lists (ACLs)

A network access control list (ACL) is a virtual firewall that controls inbound and outbound traffic at the subnet level.

For example, step outside of the coffee shop and imagine that you are in an airport. In 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 out of the country. If a traveler is on an approved list, they are able to get through. However, if they are not on the approved list or are explicitly on a list of banned travelers, they cannot come in.



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.

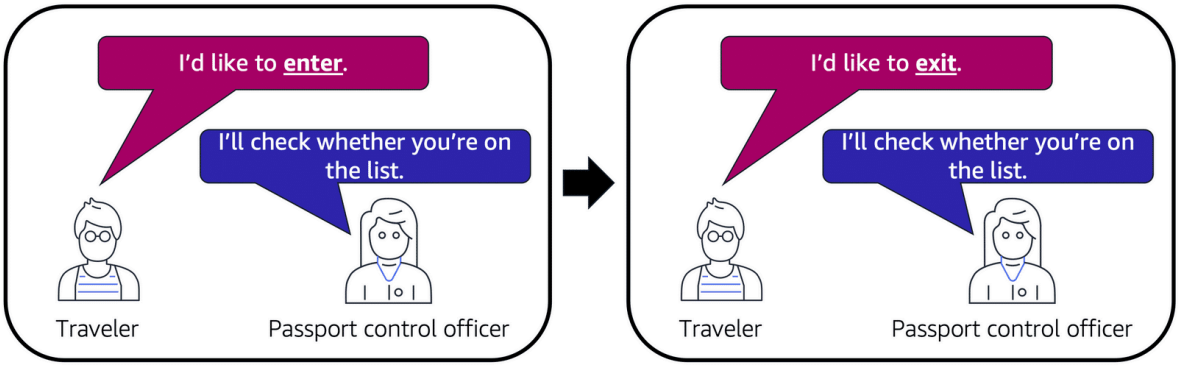
By default, your account's default network ACL allows all inbound and outbound traffic, but you can modify it by adding your own rules. For custom network ACLs, all inbound and outbound traffic is denied until you add rules to specify which traffic to allow. Additionally, all network ACLs have an explicit deny rule. This rule ensures that if a packet doesn't match any of the other rules on the list, the packet is denied.

Stateless Packet Filtering

Network ACLs perform stateless packet filtering. They remember nothing and check packets that cross the subnet border each way: inbound and outbound.

Recall the previous example of a traveler who wants to enter into a different country. This is similar to sending a request out from an Amazon EC2 instance and to the internet.

When a packet response for that request comes back to the subnet, the network ACL does not remember your previous request. The network ACL checks the packet response against its list of rules to determine whether to allow or deny.

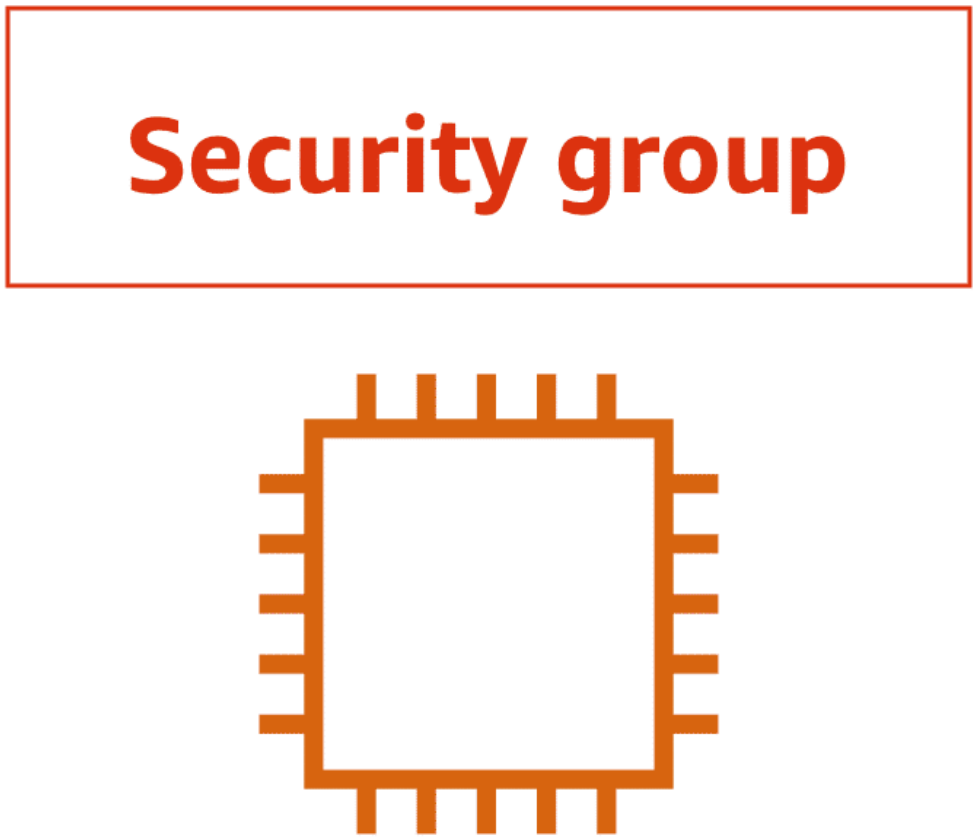


After a packet has entered a subnet, it must have its permissions evaluated for resources within the subnet, such as Amazon EC2 instances.

The VPC component that checks packet permissions for an Amazon EC2 instance is a security group.

Security Groups

A security group is a virtual firewall that controls inbound and outbound traffic for an Amazon EC2 instance.



By default, a security group denies all inbound traffic and allows all outbound traffic. You can add custom rules to configure which traffic to allow or deny.

For this example, suppose that you are in an apartment building with a door attendant who greets guests in the lobby. You can think of the guests as packets and the door attendant as a security group. As guests arrive, the door attendant checks a list to ensure they can enter the building. However, the door attendant does not check the list again when guests are exiting the building

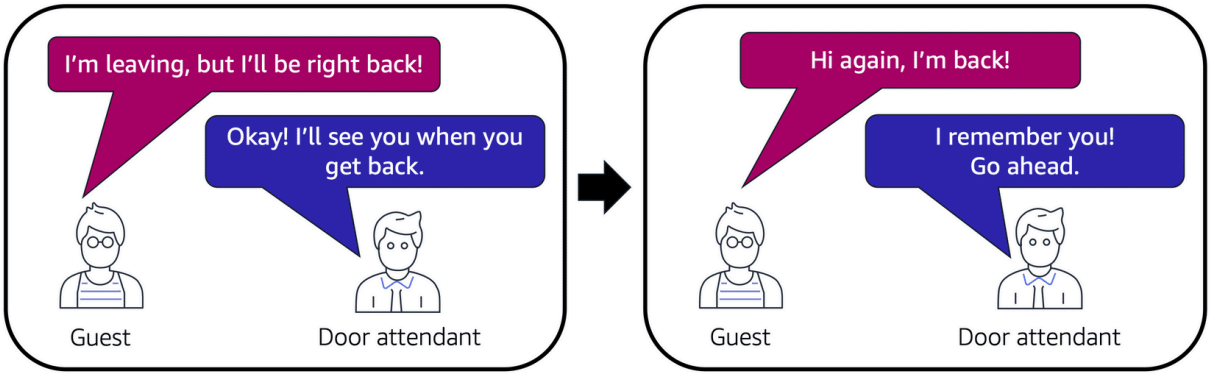
If you have multiple Amazon EC2 instances within a subnet, you can associate them with the same security group or use different security groups for each instance.

Stateful Packet Filtering

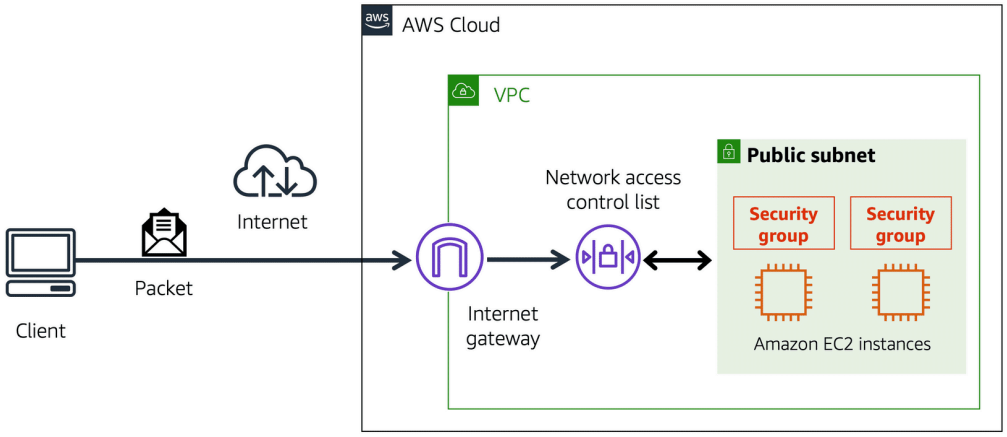
Security groups perform stateful packet filtering. They remember previous decisions made for incoming packets.

Consider the same example of sending a request out from an Amazon EC2 instance to the internet.

When a packet response for that request returns to the instance, the security group remembers your previous request. The security group allows the response to proceed, regardless of inbound security group rules.



Both network ACLs and security groups enable you to configure custom rules for the traffic in your VPC. As you continue to learn more about AWS security and networking, make sure to understand the differences between network ACLs and security groups.

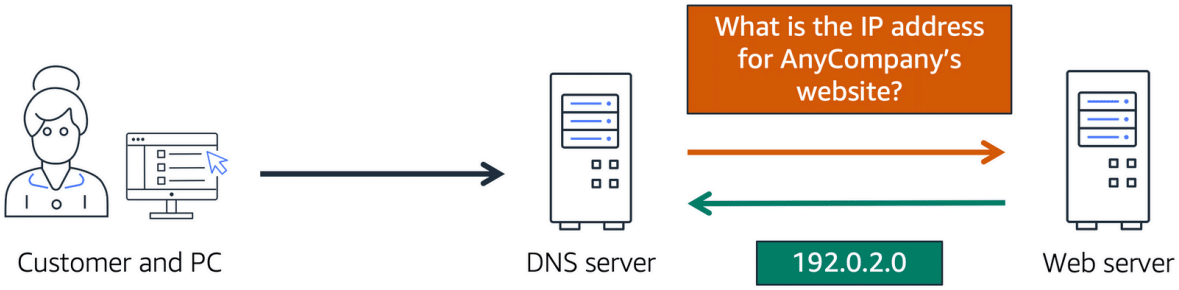


Global Networking

Domain Name System (DNS)

Suppose that AnyCompany has a website hosted in the AWS Cloud. Customers enter the web address into their browser, and they are able to access the website. This happens because of Domain Name System (DNS) resolution. DNS resolution involves a DNS server communicating with a web server.

You can think of DNS as being the phone book of the internet. DNS resolution is the process of translating a domain name to an IP address.



For example, suppose that you want to visit AnyCompany's website.

1. When you enter the domain name into your browser, this request is sent to a DNS server.
2. The DNS server asks the web server for the IP address that corresponds to AnyCompany's website.
3. The web server responds by providing the IP address for AnyCompany's website, 192.0.2.0.

Amazon Route 53

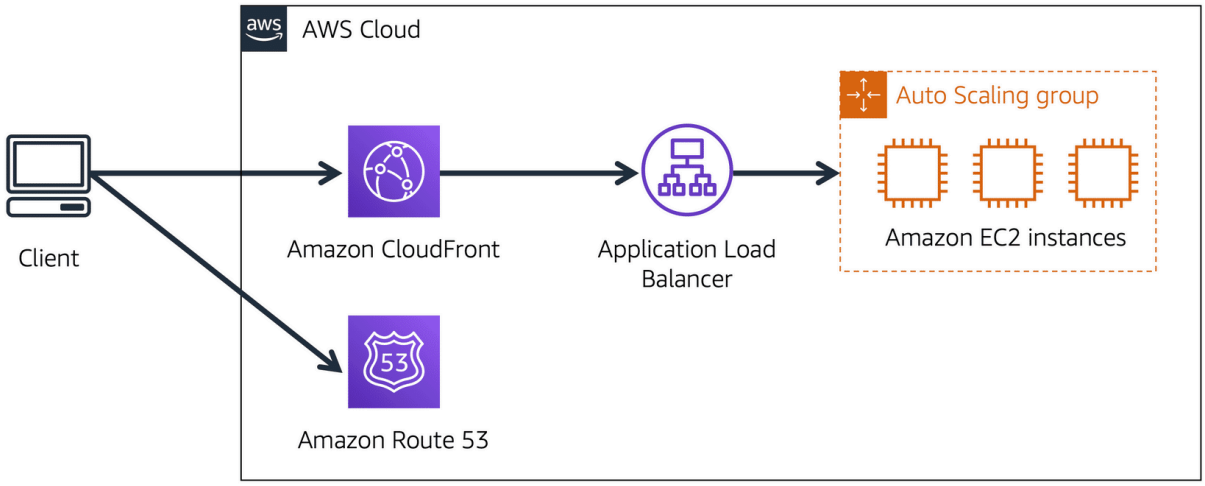
Amazon Route 53 is a DNS web service. It gives developers and businesses a reliable way to route end users to internet applications hosted in AWS.

Amazon Route 53 connects user requests to infrastructure running in AWS (such as Amazon EC2 instances and load balancers). It can route users to infrastructure outside of AWS.

Another feature of Route 53 is the ability to manage the DNS records for domain names. You can register new domain names directly in Route 53. You can also transfer DNS records for existing domain names managed by other domain registrars. This enables you to manage all of your domain names within a single location.

In the previous module, you learned about Amazon CloudFront, a content delivery service. The following example describes how Route 53 and Amazon CloudFront work together to deliver content to customers.

Example: How Amazon Route 53 and Amazon CloudFront deliver content



Suppose that AnyCompany's application is running on several Amazon EC2 instances. These instances are in an Auto Scaling group that attaches to an Application Load Balancer.

1. A customer requests data from the application by going to AnyCompany's website.
2. Amazon Route 53 uses DNS resolution to identify AnyCompany.com's corresponding IP address, 192.0.2.0. This information is sent back to the customer.
3. The customer's request is sent to the nearest edge location through Amazon CloudFront.
4. Amazon CloudFront connects to the Application Load Balancer, which sends the incoming packet to an Amazon EC2 instance.

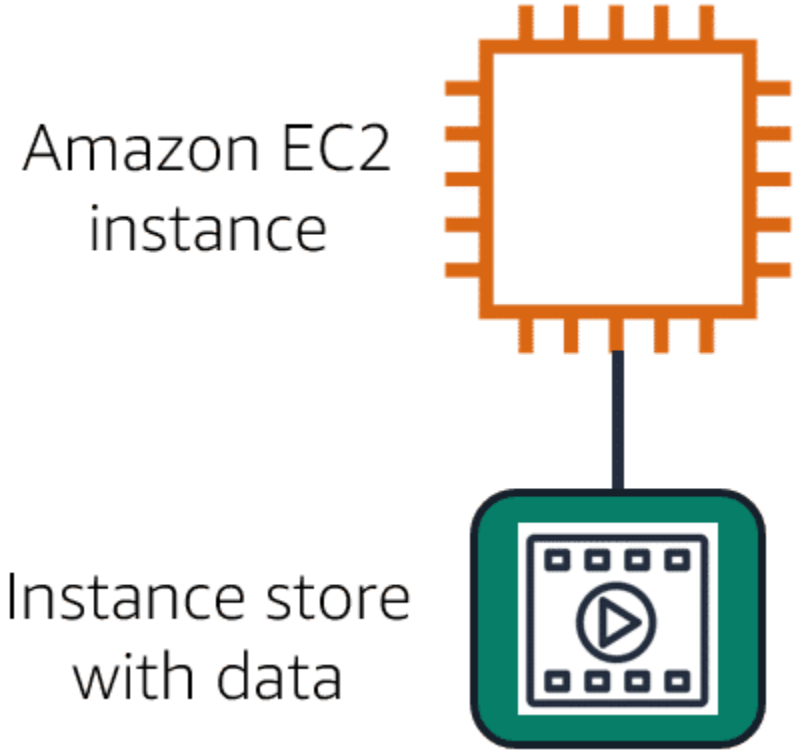
Instance Stores and Amazon Elastic Block Store (Amazon EBS)

Instance stores

Block-level storage volumes behave like physical hard drives.

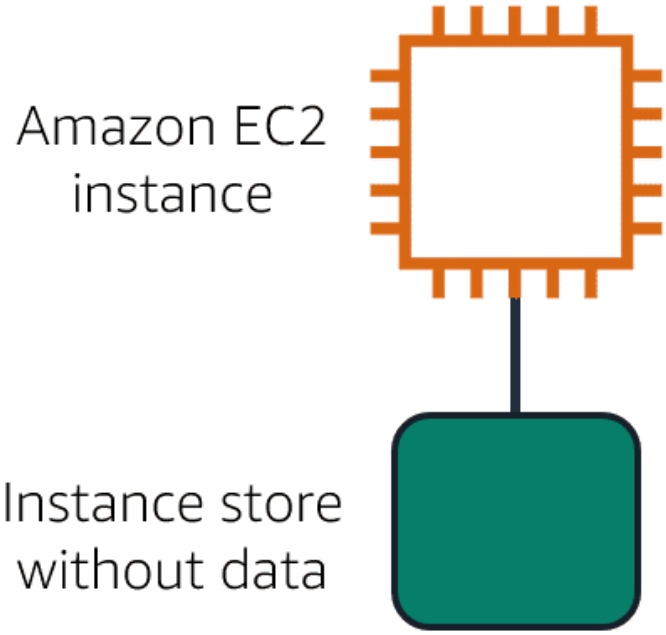
An instance store provides temporary block-level storage for an Amazon EC2 instance. An instance store is disk storage that is physically attached to the host computer for an EC2 instance, and therefore has the same lifespan as the instance. When the instance is terminated, you lose any data in the instance store.

An Amazon EC2 instance with an attached instance store is running.



The instance is stopped or terminated.

All data on the attached instance store is deleted.

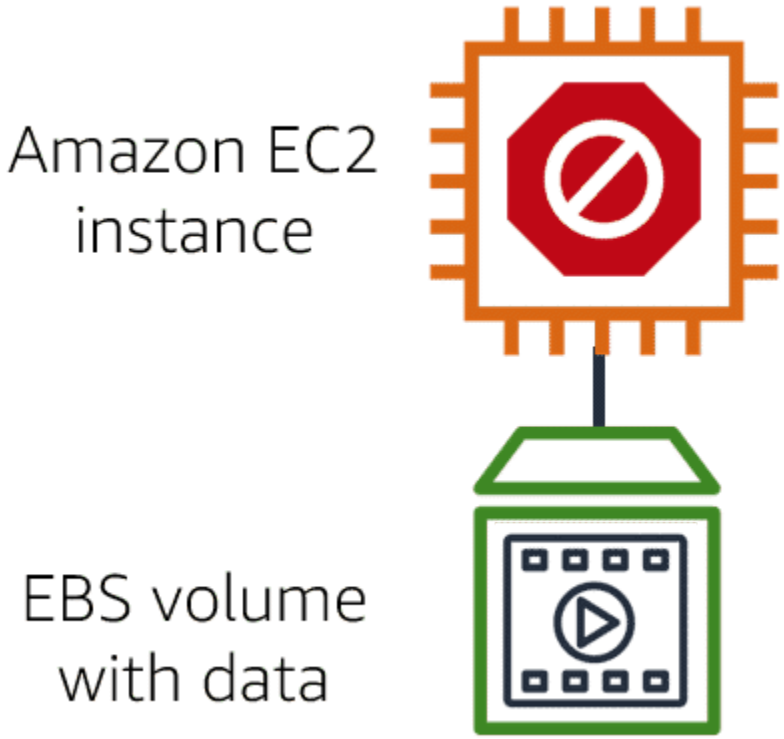


Amazon Elastic Block Storage (Amazon EBS)

Amazon Elastic Block Store (Amazon EBS) is a service that 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.

To create an EBS volume, you define the configuration (such as volume size and type) and provision it. After you create an EBS volume, it can attach to an Amazon EC2 instance.

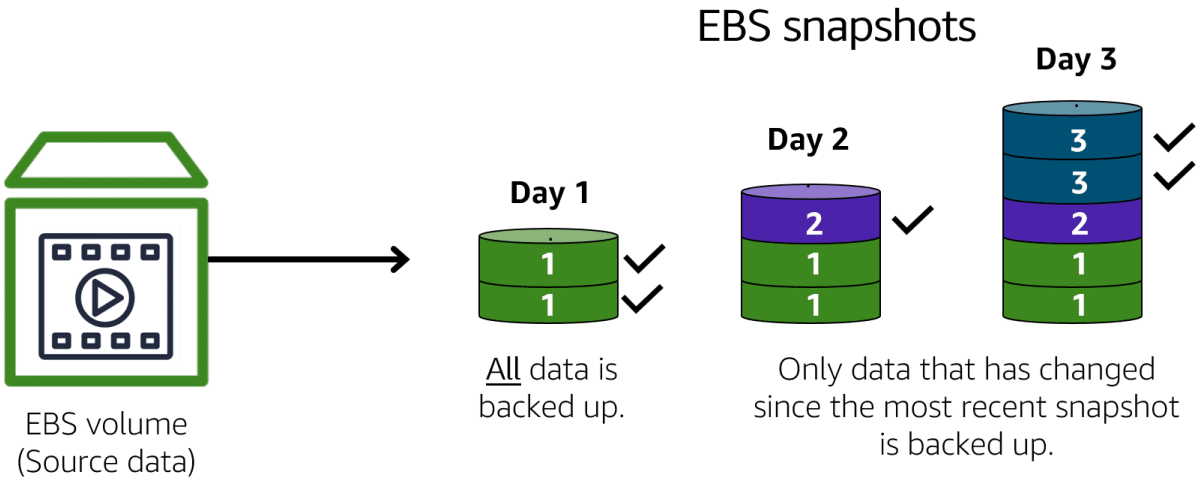
Because EBS volumes are for data that needs to persist, it's important to back up the data. You can take incremental backups of EBS volumes by creating Amazon EBS snapshots.



Amazon EBS Snapshots

An EBS snapshot is an incremental backup. This means that the first backup taken of a volume copies all the data. For subsequent backups, only the blocks of data that have changed since the most recent snapshot are saved.

Incremental backups are different from full backups, in which all the data in a storage volume copies each time a backup occurs. The full backup includes data that has not changed since the most recent backup.

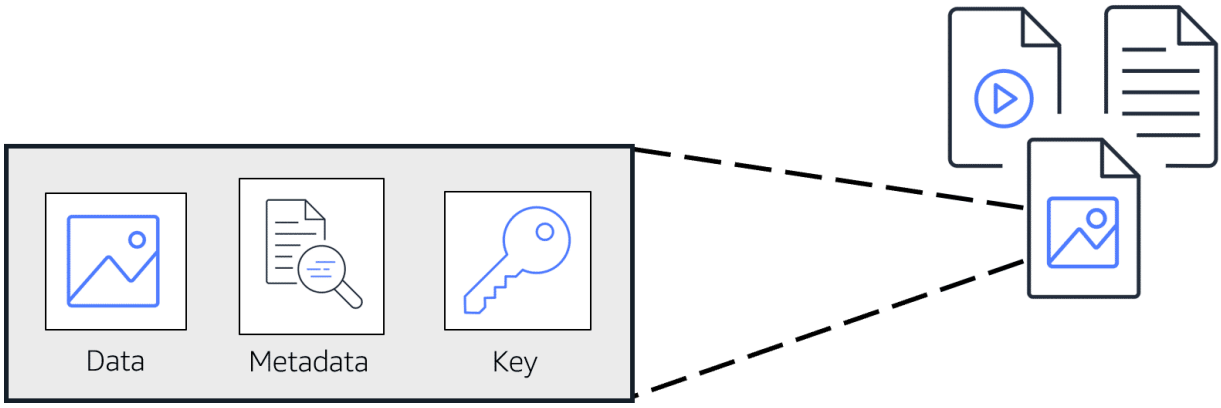


Amazon Simple Storage Service (Amazon S3)

Object Storage

In object storage, each object consists of data, metadata, and a key.

The data might be an image, video, text document, or any other type of file. Metadata contains information about what the data is, how it is used, the object size, and so on. An object's key is its unique identifier.



Recall that when you modify a file in block storage, only the pieces that are changed are updated. When a file in object storage is modified, the entire object is updated.

Amazon Simple Storage Service (Amazon S3)

Amazon Simple Storage Service (Amazon S3) is a service that provides object-level storage. Amazon S3 stores data as objects in buckets.

You can upload any type of file to Amazon S3, such as images, videos, text files, and so on. For example, you might use Amazon S3 to store backup files, media files for a website, or archived documents. Amazon S3 offers unlimited storage space. The maximum file size for an object in Amazon S3 is 5 TB.

When you upload a file to Amazon S3, you can set permissions to control visibility and access to it. You can also use the Amazon S3 versioning feature to track changes to your objects over time.

Amazon S3 Storage Classes

With Amazon S3, you pay only for what you use. You can choose from a range of storage classes to select a fit for your business and cost needs. When selecting an Amazon S3 storage class, consider these two factors:

- How often you plan to retrieve your data
- How available you need your data to be

Amazon S3 Standard

- Designed for frequently accessed data
- Stores data in a minimum of three Availability Zones

Amazon S3 Standard provides high availability for objects. This makes it a good choice for a wide range of use cases, such as websites, content distribution, and data analytics. Amazon S3 Standard has a higher cost than other storage classes intended for infrequently accessed data and archival storage.

Amazon S3 Standard-Infrequent Access (S3 Standard-IA)

- Ideal for infrequently accessed data
- Similar to Amazon S3 Standard but has a lower storage price and higher retrieval price

Amazon S3 Standard-IA is ideal for data infrequently accessed but requires high availability when needed. Both Amazon S3 Standard and Amazon S3 Standard-IA store data in a minimum of three Availability Zones. S3 Standard-IA provides the same level of availability as Amazon S3 Standard but with a lower storage price and a higher retrieval price.

Amazon S3 One Zone-Infrequent Access (S3 One Zone-IA)

- Stores data in a single Availability Zone
- Has a lower storage price than Amazon S3 Standard-IA

Compared to Amazon S3 Standard and Amazon S3 Standard-IA, which store data in a minimum of three Availability Zones, Amazon S3 One Zone-IA stores data in a single Availability Zone. This makes it a good storage class to consider if the following conditions apply:

- You want to save costs on storage.
- You can easily reproduce your data in the event of an Availability Zone failure.

Amazon S3 Intelligent-Tiering

- Ideal for data with unknown or changing access patterns
- Requires a small monthly monitoring and automation fee per object

In the Amazon S3 Intelligent-Tiering storage class, Amazon S3 monitors objects' access patterns. If you haven't accessed an object for 30 consecutive days, Amazon S3 automatically moves it to the infrequent access tier, Amazon S3 Standard-IA. If you access an object in the infrequent access tier, Amazon S3 automatically moves it to the frequent access tier, Amazon S3 Standard.

Amazon S3 Glacier Instant Retrieval

- Works well for archived data that requires immediate access
- Can retrieve objects within a few milliseconds

When you decide between the options for archival storage, consider how quickly you must retrieve the archived objects. You can retrieve objects stored in the Amazon S3 Glacier Instant Retrieval storage class within milliseconds, with the same performance as Amazon S3 Standard.

Amazon S3 Glacier Flexible Retrieval

- Low-cost storage designed for data archiving
- Able to retrieve objects within a few minutes to hours

Amazon S3 Glacier Flexible Retrieval is a low-cost storage class that is ideal for data archiving. For example, you might use this storage class to store archived customer records or older photos and video files.

Amazon S3 Glacier Deep Archive

- Lowest-cost object storage class ideal for archiving
- Able to retrieve objects within 12 hours

Amazon S3 Deep Archive supports long-term retention and digital preservation for data that might be accessed once or twice in a year. This storage class is the lowest-cost storage in the AWS Cloud, with data retrieval from 12 to 48 hours. All objects from this storage class are replicated and stored across at least three geographically dispersed Availability Zones.

Amazon S3 Outposts

- Creates S3 buckets on Amazon S3 Outposts
- Makes it easier to retrieve, store, and access data on AWS Outposts

Amazon S3 Outposts delivers object storage to your on-premises AWS Outposts environment. Amazon S3 Outposts is designed to store data durably and redundantly across multiple devices and servers on your Outposts. It works well for workloads with local data residency requirements that must satisfy demanding performance needs by keeping data close to on-premises applications.

Amazon Elastic File System (Amazon EFS)
File Storage

In file storage, multiple clients (such as users, applications, servers, and so on) can access data that is stored in shared file folders. In this approach, a storage server uses block storage with a local file system to organize files. Clients access data through file paths.

Compared to block storage and object storage, file storage is ideal for use cases in which a large number of services and resources need to access the same data at the same time.

Amazon Elastic File System (Amazon EFS) is a scalable file system used with AWS Cloud services and on-premises resources. As you add and remove files, Amazon EFS grows and shrinks automatically. It can scale on demand to petabytes without disrupting applications.

Comparing Amazon EBS and Amazon EFS

Amazon EBS

- An Amazon EBS volume stores data in a single Availability Zone.
- To attach an Amazon EC2 instance to an EBS volume, both the Amazon EC2 instance and the EBS volume must reside within the same Availability Zone.

Amazon EFS

- Amazon EFS is a regional service. It stores data in and across multiple Availability Zones.
- The duplicate storage enables you to access data concurrently from all the Availability Zones in the Region where a file system is located. Additionally, on-premises servers can access Amazon EFS using AWS Direct Connect.

Amazon Relational Database Service (Amazon RDS)

Relational databases

In a relational database, data is stored in a way that relates it to other pieces of data.

An example of a relational database might be the coffee shop's inventory management system. Each record in the database would include data for a single item, such as product name, size, price, and so on.

Relational databases use structured query language (SQL) to store and query data. This approach allows data to be stored in an easily understandable, consistent, and scalable way. For example, the coffee shop owners can write a SQL query to identify all the customers whose most frequently purchased drink is a medium latte.

Example of data in a relational database:

ID	Product Name	Size	Price
1	Medium roast ground coffee	12 oz.	$5.30
2	Dark roast ground coffee	20 oz.	$9.27

Amazon Relational Database Service

Amazon Relational Database Service (Amazon RDS) is a service that enables you to run relational databases in the AWS Cloud.

Amazon RDS is a managed service that automates tasks such as hardware provisioning, database setup, patching, and backups. With these capabilities, you can spend less time completing administrative tasks and more time using data to innovate your applications. You can integrate Amazon RDS with other services to fulfill your business and operational needs, such as using AWS Lambda to query your database from a serverless application.

Amazon RDS provides a number of different security options. Many Amazon RDS database engines offer encryption at rest (protecting data while it is stored) and encryption in transit (protecting data while it is being sent and received).

Amazon RDS database engines

Amazon RDS is available on six database engines, which optimize for memory, performance, or input/output (I/O). Supported database engines include:

- Amazon Aurora
- PostgreSQL
- MySQL
- MariaDB
- Oracle Database
- Microsoft SQL Server

Amazon Aurora

Amazon Aurora is an enterprise-class relational database. It is compatible with MySQL and PostgreSQL relational databases. It is up to five times faster than standard MySQL databases and up to three times faster than standard PostgreSQL databases.

Amazon Aurora helps to reduce your database costs by reducing unnecessary input/output (I/O) operations, while ensuring that your database resources remain reliable and available.

Consider Amazon Aurora if your workloads require high availability. It replicates six copies of your data across three Availability Zones and continuously backs up your data to Amazon S3.

Amazon DynamoDB

Nonrelational Databases

In a nonrelational database, you create tables. A table is a place where you can store and query data.

Nonrelational databases are sometimes referred to as “NoSQL databases” because they use structures other than rows and columns to organize data. One type of structural approach for nonrelational databases is key-value pairs. With key-value pairs, data is organized into items (keys), and items have attributes (values). You can think of attributes as being different features of your data.

In a key-value database, you can add or remove attributes from items in the table at any time. Additionally, not every item in the table has to have the same attributes.

Example of data in a nonrelational database:Amazon DynamoDB

Key	Value
1	**Name**: John Doe **Address**: 123 Any Street **Favorite drink**: Medium latte
2	**Name**: Mary Major **Address**: 100 Main Street **Birthday**: July 5, 1994

Amazon DynamoDB is a key-value database service. It delivers single-digit millisecond performance at any scale.

Serverless

- DynamoDB is serverless, which means that you do not have to provision, patch, or manage servers.
- You also do not have to install, maintain, or operate software.

Automatic Scaling

- As the size of your database shrinks or grows, DynamoDB automatically scales to adjust for changes in capacity while maintaining consistent performance.
- This makes it a suitable choice for use cases that require high performance while scaling.

Relational Databases (like traditional tables):

- **Think of a spreadsheet:**
    - Data is organized into tables with rows and columns.
    - Relationships between tables are defined (e.g., a "customers" table and an "orders" table can be linked by a customer ID).
    - They're very structured, ensuring data consistency and accuracy.
    - They use SQL (Structured Query Language) to manage and retrieve data.
    - Best for: Applications where data integrity and complex relationships are crucial (like banking or financial systems).

Non-Relational Databases (NoSQL):

- **Think of a flexible storage container:**
    - Data can be stored in various formats (documents, key-value pairs, graphs, etc.).
    - They're more flexible and can handle unstructured or rapidly changing data.
    - They're designed to scale easily, handling large volumes of data and high traffic.
    - Best for: Applications that need flexibility and scalability, like social media, big data, or real-time web applications.

- **Key points:**
    - They are often called NoSQL, which means "Not Only SQL".
    - They are very good at scaling out, meaning that you can add more servers to handle more data.

AWS Database Migration Service (AWS DMS)

AWS Database Migration Service (AWS DMS) enables you to migrate relational databases, nonrelational databases, and other types of data stores.

With AWS DMS, you move data between a source database and a target database. The source and target databases can be of the same type or different types. During the migration, your source database remains operational, reducing downtime for any applications that rely on the database.

For example, suppose that you have a MySQL database that is stored on premises in an Amazon EC2 instance or in Amazon RDS. Consider the MySQL database to be your source database. Using AWS DMS, you could migrate your data to a target database, such as an Amazon Aurora database.

Other use cases for AWS DMS

Development and test database migrations

- Enabling developers to test applications against production data without affecting production users

Database consolidation

- Combining several databases into a single database

Continuous replication

- Sending ongoing copies of your data to other target sources instead of doing a one-time migration

Amazon DocumentDB

Amazon DocumentDB is a document database service that supports MongoDB workloads. (MongoDB is a document database program.)

Amazon Neptune

Amazon Neptune is a graph database service.

You can use Amazon Neptune to build and run applications that work with highly connected datasets, such as recommendation engines, fraud detection, and knowledge graphs.

Amazon Quantum Ledger Database (Amazon QLDB)

Amazon Quantum Ledger Database (Amazon QLDB) is a ledger database service.

You can use Amazon QLDB to review a complete history of all the changes that have been made to your application data.

Amazon Managed Blockchain

Amazon Managed Blockchain is a service that you can use to create and manage blockchain networks with open-source frameworks.

Blockchain is a distributed ledger system that lets multiple parties run transactions and share data without a central authority.

Amazon ElastiCache

Amazon ElastiCache is a service that adds caching layers on top of your databases to help improve the read times of common requests.

It supports two types of data stores: Redis and Memcached.

Amazon DynamoDB Accelerator

Amazon DynamoDB Accelerator (DAX) is an in-memory cache for DynamoDB.

It helps improve response times from single-digit milliseconds to microseconds.

EC2 Instance layers

Physical Layer:

- Imagine the actual hardware – the servers in Amazon's data centers. This is the foundation. It's the real machines with processors, memory, and storage.
- Think of it as the concrete and steel of a building.

Hypervisor Layer:

- This is the software that allows Amazon to run multiple virtual machines (your EC2 instances) on that single physical server.
- It's like a manager that divides the physical server's resources into separate, isolated compartments.
- Think of it as the walls that divide a large office space into individual cubicles.

Network Layer:

- This is how your EC2 instance connects to the internet and other resources.
- It handles things like IP addresses, routing, and security groups (firewalls).
- Think of it as the roads and traffic signals that allow cars to move around.

Operating System (OS) Layer:

- This is the software that controls the basic functions of your virtual machine, like Windows or Linux.
- It's the platform on which you run your applications.
- Think of it as the operating system on your computer.

Application Layer:

- This is where your software lives – the programs you use to perform specific tasks, like a website, a game server, or a data processing tool.
- This is the actual software that the user interacts with. Example, a web page.

Data and Database Layer:

- This is where your information is stored and managed. Databases hold structured data, and filesystems store other kinds of data.
- Think of it as the filing cabinets and storage rooms where you keep all your important documents.
- The database is a more organized way of storing and retrieving specific pieces of data.

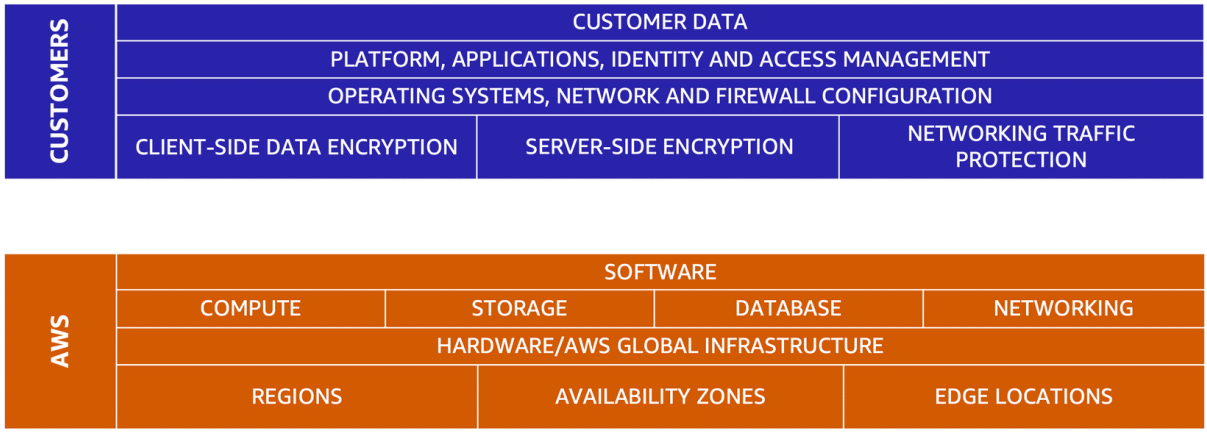
Shared Responsibility Model

The AWS Shared Responsibility Model

Throughout this course, you have learned about a variety of resources that you can create in the AWS Cloud. These resources include Amazon EC2 instances, Amazon S3 buckets, and Amazon RDS databases. Who is responsible for keeping these resources secure: you (the customer) or AWS?

The answer is both. The reason is that you do not treat your AWS environment as a single object. Rather, you treat the environment as a collection of parts that build upon each other. AWS is responsible for some parts of your environment and you (the customer) are responsible for other parts. This concept is known as the shared responsibility model.

The shared responsibility model divides into customer responsibilities (commonly referred to as "security in the cloud") and AWS responsibilities (commonly referred to as "security of the cloud").



You can think of this model as being similar to the division of responsibilities between a homeowner and a homebuilder. The builder (AWS) is responsible for constructing your house and ensuring that it is solidly built. As the homeowner (the customer), it is your responsibility to secure everything in the house by ensuring that the doors are closed and locked.

Customers: Security in the Cloud

Customers are responsible for the security of everything that they create and put *in* the AWS Cloud.

When using AWS services, you, the customer, maintain complete control over your content. You are responsible for managing security requirements for your content, including which content you choose to store on AWS, which AWS services you use, and who has access to that content. You also control how access rights are granted, managed, and revoked.

The security steps that you take will depend on factors such as the services that you use, the complexity of your systems, and your company's specific operational and security needs. Steps include selecting, configuring, and patching the operating systems that will run on Amazon EC2 instances, configuring security groups, and managing user accounts.

AWS: Security of the Cloud

AWS is responsible for security *of* the cloud.

AWS operates, manages, and controls the components at all layers of infrastructure. This includes areas such as the host operating system, the virtualization layer, and even the physical security of the data centers from which services operate.

AWS is responsible for protecting the global infrastructure that runs all of the services offered in the AWS Cloud. This infrastructure includes AWS Regions, Availability Zones, and edge locations.

AWS manages the security of the cloud, specifically the physical infrastructure that hosts your resources, which include:

- Physical security of data centers
- Hardware and software infrastructure
- Network infrastructure
- Virtualization infrastructure

Although you cannot visit AWS data centers to see this protection firsthand, AWS provides several reports from third-party auditors. These auditors have verified its compliance with a variety of computer security standards and regulations.

User Permission and Access

AWS Identity and Access Management (IAM)

AWS Identity and Access Management (IAM) enables you to manage access to AWS services and resources securely.

IAM gives you the flexibility to configure access based on your company's specific operational and security needs. You do this by using a combination of IAM features, which are explored in detail in this lesson:

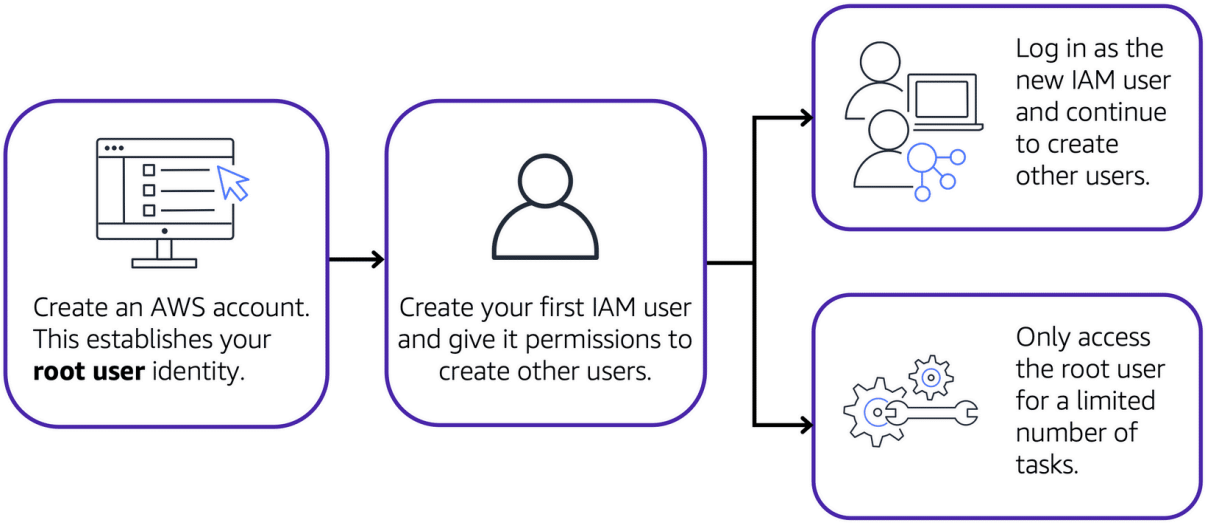
- IAM users, groups, and roles
- IAM policies
- Multi-factor authentication

You will also learn best practices for each of these features.

AWS Account Root User

When you first create an AWS account, you begin with an identity known as the root user.

The root user is accessed by signing in with the email address and password that you used to create your AWS account. You can think of the root user as being similar to the owner of the coffee shop. It has complete access to all the AWS services and resources in the account.



Best practice:

Do not use the root user for everyday tasks.

Instead, use the root user to create your first IAM user and assign it permissions to create other users.

Then, continue to create other IAM users, and access those identities for performing regular tasks throughout AWS. Only use the root user when you need to perform a limited number of tasks that are only available to the root user. Examples of these tasks include changing your root user email address and changing your AWS support plan.

IAM users

An IAM user is an identity that you create in AWS. It represents the person or application that interacts with AWS services and resources. It consists of a name and credentials.

By default, when you create a new IAM user in AWS, it has no permissions associated with it. To allow the IAM user to perform specific actions in AWS, such as launching an Amazon EC2 instance or creating an Amazon S3 bucket, you must grant the IAM user the necessary permissions.

Best practice:

We recommend that you create individual IAM users for each person who needs to access AWS.

Even if you have multiple employees who require the same level of access, you should create individual IAM users for each of them. This provides additional security by allowing each IAM user to have a unique set of security credentials.

IAM policies

An IAM policy is a document that allows or denies permissions to AWS services and resources.

IAM policies enable you to customize users' levels of access to resources. For example, you can allow users to access all of the Amazon S3 buckets within your AWS account, or only a specific bucket.

Best practice:

Follow the security principle of least privilege when granting permissions.

By following this principle, you help to prevent users or roles from having more permissions than needed to perform their tasks.

For example, if an employee needs access to only a specific bucket, specify the bucket in the IAM policy. Do this instead of granting the employee access to all of the buckets in your AWS account.

Example: IAM policy

Here's an example of how IAM policies work. Suppose that the coffee shop owner has to create an IAM user for a newly hired cashier. The cashier needs access to the receipts kept in an Amazon S3 bucket with the ID: AWSDOC-EXAMPLE-BUCKET.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:ListObject",
    "Resource": "arn:aws:s3:::
AWSDOC-EXAMPLE-BUCKET"
  }
}
```

In this example, the IAM policy is allowing a specific action within Amazon S3: ListObject. The policy also mentions a specific bucket ID: AWSDOC-EXAMPLE-BUCKET. When the owner attaches this policy to the cashier's IAM user, it will allow the cashier to view all of the objects in the AWSDOC-EXAMPLE-BUCKET bucket.

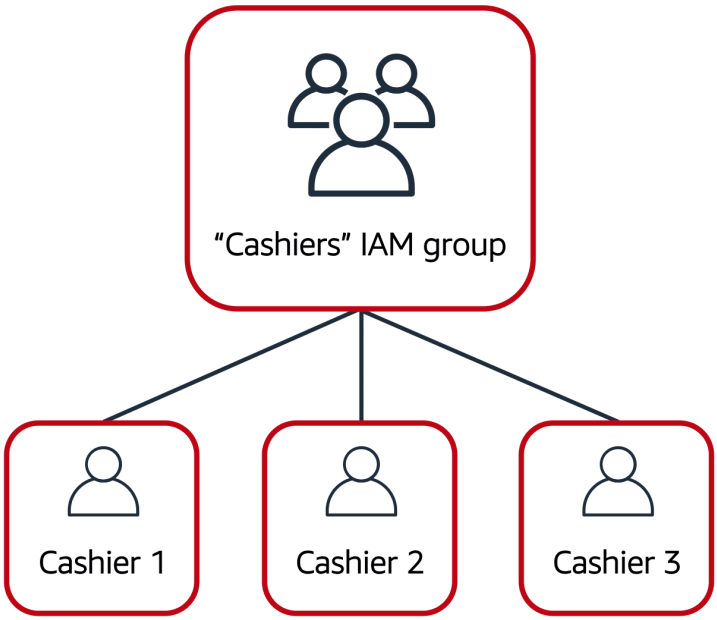
If the owner wants the cashier to be able to access other services and perform other actions in AWS, the owner must attach additional policies to specify these services and actions.

Now, suppose that the coffee shop has hired a few more cashiers. Instead of assigning permissions to each individual IAM user, the owner places the users into an IAM group.

IAM Groups

An IAM group is a collection of IAM users. When you assign an IAM policy to a group, all users in the group are granted permissions specified by the policy.

Here's an example of how this might work in the coffee shop. Instead of assigning permissions to cashiers one at a time, the owner can create a "Cashiers" IAM group. The owner can then add IAM users to the group and then attach permissions at the group level.



Assigning IAM policies at the group level also makes it easier to adjust permissions when an employee transfers to a different job. For example, if a cashier becomes an inventory specialist, the coffee shop owner removes them from the “Cashiers” IAM group and adds them into the “Inventory Specialists” IAM group. This ensures that employees have only the permissions that are required for their current role.

What if a coffee shop employee hasn’t switched jobs permanently, but instead, rotates to different workstations throughout the day? This employee can get the access they need through IAM roles.

IAM Roles

In the coffee shop, an employee rotates to different workstations throughout the day. Depending on the staffing of the coffee shop, this employee might perform several duties: work at the cash register, update the inventory system, process online orders, and so on.

When the employee needs to switch to a different task, they give up their access to one workstation and gain access to the next workstation. The employee can easily switch between workstations, but at any given point in time, they can have access to only a single workstation. This same concept exists in AWS with IAM roles.

An IAM role is an identity that you can assume to gain temporary access to permissions.

Before an IAM user, application, or service can assume an IAM role, they must be granted permissions to switch to the role. When someone assumes an IAM role, they abandon all previous permissions that they had under a previous role and assume the permissions of the new role.

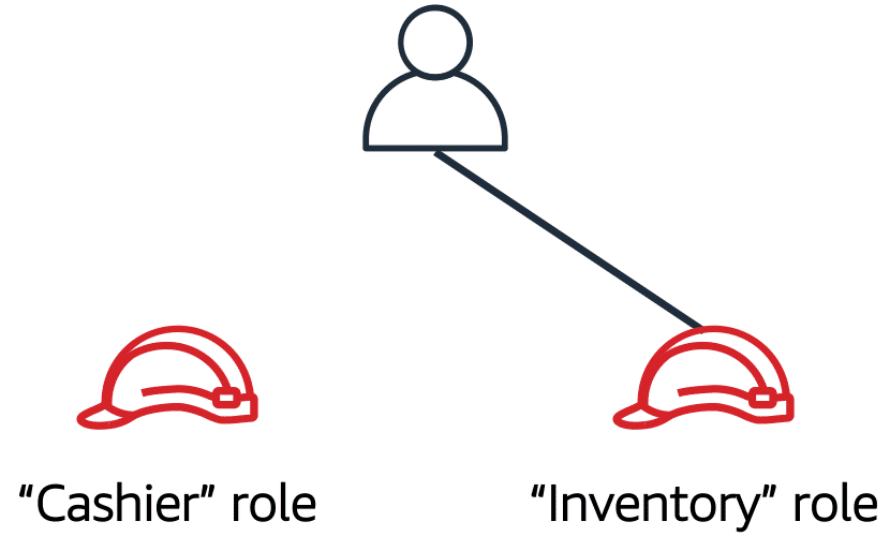
Best practice:

IAM roles are ideal for situations in which access to services or resources needs to be granted temporarily, instead of long-term.

Example: IAM Roles

Review an example of how IAM roles could be used in the coffee shop:

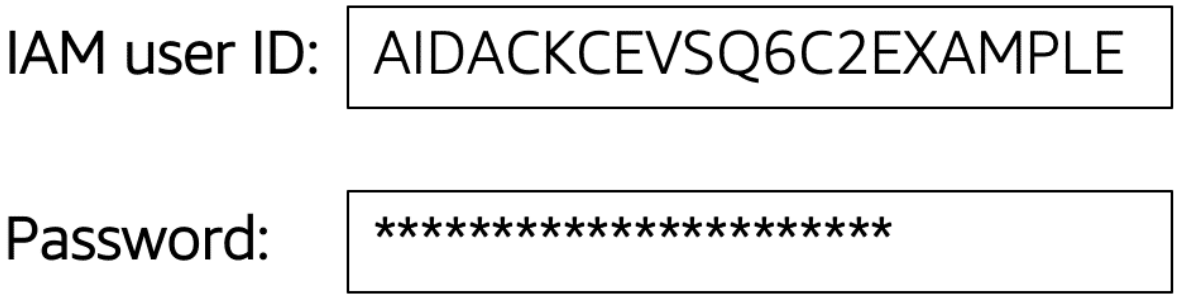
1. First, the owner grants the employee permissions to switch to a role for each workstation in the coffee shop.
2. Next, the employee begins their day by assuming the “Cashier” role. This grants them access to the cash register system.
3. Later in the day, the employee needs to update the inventory system. They assume the “Inventory” role. This grants the employee access to the inventory system and also revokes their access to the cash register system.



Multi-factor Authentication

Have you ever signed in to a website that required you to provide multiple pieces of information to verify your identity? You might have needed to provide your password and then a second form of authentication, such as a random code sent to your phone. This is an example of multi-factor authentication.

In IAM, multi-factor authentication (MFA) provides an extra layer of security for your AWS account.



1. First, when a user signs in to an AWS website, they enter their IAM user ID and password.
2. Next, the user is prompted for an authentication response from their AWS MFA device. This device could be a hardware security key, a hardware device, or an MFA application on a device such as a smartphone.
3. When the user has been successfully authenticated, they are able to access the requested AWS services or resources.

You can enable MFA for the root user and IAM users. As a best practice, enable MFA for the root user and all IAM users in your account. By doing this, you can keep your AWS account safe from unauthorized access.

AWS Organizations

Suppose that your company has multiple AWS accounts. You can use AWS Organizations to consolidate and manage multiple AWS accounts within a central location.

When you create an organization, AWS Organizations automatically creates a root, which is the parent container for all the accounts in your organization.

In AWS Organizations, you can centrally control permissions for the accounts in your organization by using service control policies (SCPs). SCPs enable you to place restrictions on the AWS services, resources, and individual API actions that users and roles in each account can access.

Consolidated billing is another feature of AWS Organizations. You will learn about consolidated billing in a later module.

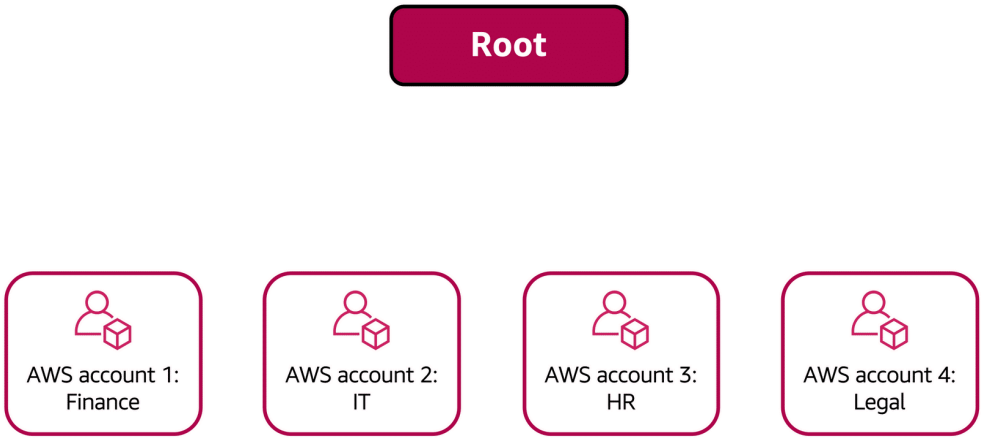
Organizational Units

In AWS Organizations, you can group accounts into organizational units (OUs) to make it easier to manage accounts with similar business or security requirements. When you apply a policy to an OU, all the accounts in the OU automatically inherit the permissions specified in the policy.

By organizing separate accounts into OUs, you can more easily isolate workloads or applications that have specific security requirements. For instance, if your company has accounts that can access only the AWS services that meet certain regulatory requirements, you can put these accounts into one OU. Then, you can attach a policy to the OU that blocks access to all other AWS services that do not meet the regulatory requirements.

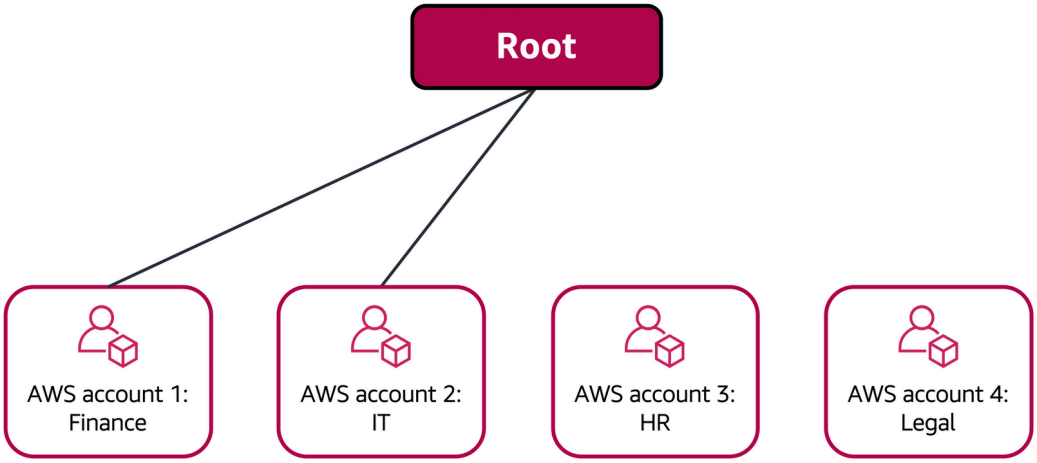
Example: AWS Organizations

Review an example of how a company might use AWS Organizations:

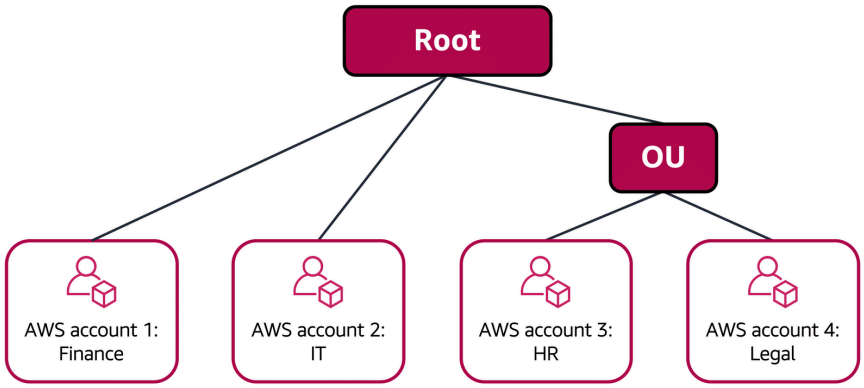


Imagine that your company has separate AWS accounts for the finance, information technology (IT), human resources (HR), and legal departments. You decide to consolidate these accounts into a single organization so that you can administer them from a central location. When you create the organization, this establishes the root.

In designing your organization, you consider the business, security, and regulatory needs of each department. You use this information to decide which departments group together in OUs.



The finance and IT departments have requirements that do not overlap with those of any other department. You bring these accounts into your organization to take advantage of benefits such as consolidated billing, but you do not place them into any OUs.



The HR and legal departments need to access the same AWS services and resources, so you place them into an OU together. Placing them into an OU enables you to attach policies that apply to both the HR and legal departments' AWS accounts.

Even though you have placed these accounts into OUs, you can continue to provide access for users, groups, and roles through IAM.

By grouping your accounts into OUs, you can more easily give them access to the services and resources that they need. You also prevent them from accessing any services or resources that they do not need.

Compliance

AWS Artifact

Depending on your company's industry, you may need to uphold specific standards. An audit or inspection will ensure that the company has met those standards.

AWS Artifact is a service that provides on-demand access to AWS security and compliance reports and select online agreements. AWS Artifact consists of two main sections: 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.

The following are some of the compliance reports and regulations that you can find within AWS Artifact. Each report includes a description of its contents and the reporting period for which the document is valid.

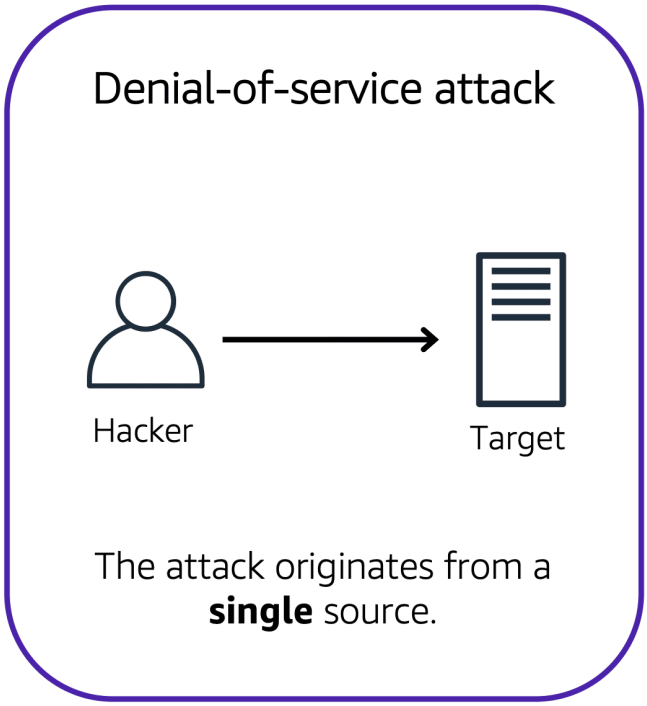
Denial-of-Service Attacks

Customers can call the coffee shop to place their orders. After answering each call, a cashier takes the order and gives it to the barista.

However, suppose that a prankster is calling in multiple times to place orders but is never picking up their drinks. This causes the cashier to be unavailable to take other customers' calls. The coffee shop can attempt to stop the false requests by blocking the phone number that the prankster is using.

In this scenario, the prankster's actions are similar to a denial-of-service attack.

Denial-of-Service Attacks - A denial-of-service (DoS) attack is a deliberate attempt to make a website or application unavailable to users.

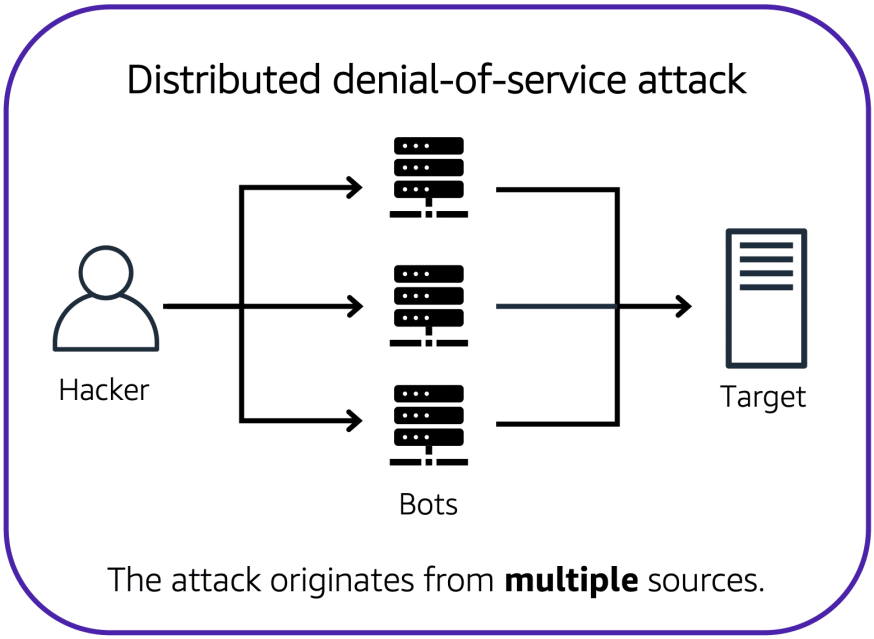


For example, an attacker might flood a website or application with excessive network traffic until the targeted website or application becomes overloaded and is no longer able to respond. If the website or application becomes unavailable, this denies service to users who are trying to make legitimate requests.

Distributed Denial-of-Service Attacks

Now, suppose that the prankster has enlisted the help of friends.

The prankster and their friends repeatedly call the coffee shop with requests to place orders, even though they do not intend to pick them up. These requests are coming in from different phone numbers, and it's impossible for the coffee shop to block them all. Additionally, the influx of calls has made it increasingly difficult for customers to be able to get their calls through. This is similar to a distributed denial-of-service attack.



In a distributed denial-of-service (DDoS) attack, multiple sources are used to start an attack that aims to make a website or application unavailable. This can come from a group of attackers, or even a single attacker. The single attacker can use multiple infected computers (also known as "bots") to send excessive traffic to a website or application.

To help minimize the effect of DoS and DDoS attacks on your applications, you can use AWS Shield.

AWS Shield

AWS Shield is a service that protects applications against DDoS attacks. AWS Shield provides two levels of protection: Standard and Advanced.

AWS Shield Standard

AWS Shield Standard automatically protects all AWS customers at no cost. It protects your AWS resources from the most common, frequently occurring types of DDoS attack.

As network traffic comes into your applications, AWS Shield Standard uses a variety of analysis techniques to detect malicious traffic in real time and automatically mitigates it.

AWS Shield Advanced

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 Elastic Load Balancing. Additionally, you can integrate AWS Shield with AWS WAF by writing custom rules to mitigate complex DDoS attacks.

Additional Security Services
AWS Key Management Service (AWS KMS)

The coffee shop has many items, such as coffee machines, pastries, money in the cash registers, and so on. You can think of these items as data. The coffee shop owners want to ensure that all of these items are secure, whether they're sitting in the storage room or being transported between shop locations.

In the same way, you must ensure that your applications' data is secure while in storage (encryption at rest) and while it is transmitted, known as encryption in transit.

AWS Key Management Service (AWS KMS) enables you to perform encryption operations through the use of cryptographic keys. A cryptographic key is a random string of digits used for locking (encrypting) and unlocking (decrypting) data. You can use AWS KMS to create, manage, and use cryptographic keys. You can also control the use of keys across a wide range of services and in your applications.

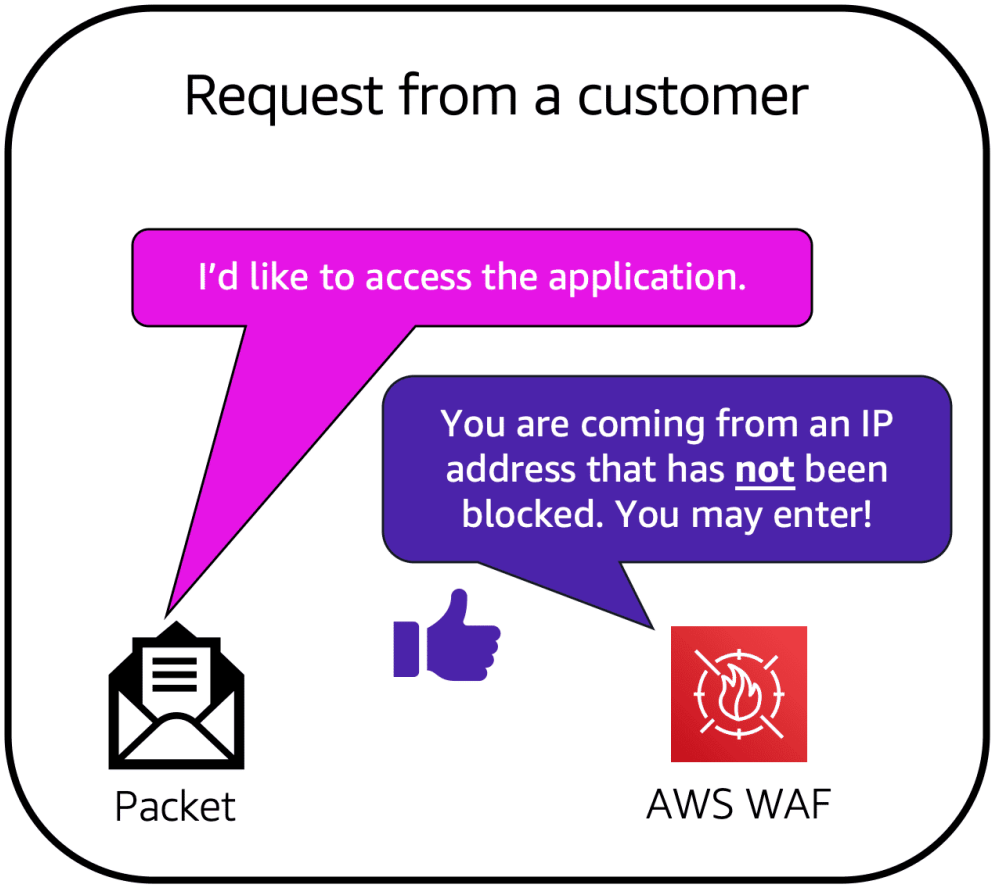
With AWS KMS, you can choose the specific levels of access control that you need for your keys. For example, you can specify which IAM users and roles are able to manage keys. Alternatively, you can temporarily disable keys so that they are no longer in use by anyone. Your keys never leave AWS KMS, and you are always in control of them.

AWS WAF

AWS WAF is a web application firewall that lets you monitor network requests that come into your web applications.

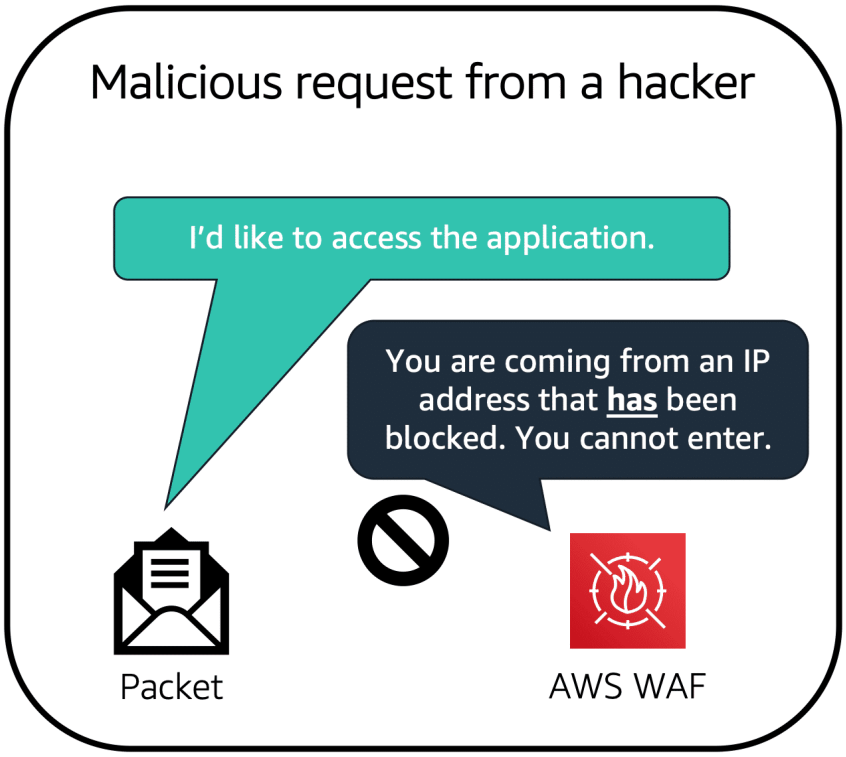
AWS WAF works together with Amazon CloudFront and an Application Load Balancer. Recall the network access control lists that you learned about in an earlier module. AWS WAF works in a similar way to block or allow traffic. However, it does this by using a web access control list (ACL) to protect your AWS resources.

Here's an example of how you can use AWS WAF to allow and block specific requests.



Suppose that your application has been receiving malicious network requests from several IP addresses. You want to prevent these requests from continuing to access your application, but you also want to ensure that legitimate users can still access it. You configure the web ACL to allow all requests except those from the IP addresses that you have specified.

When a request comes into AWS WAF, it checks against the list of rules that you have configured in the web ACL. If a request did not come from one of the blocked IP addresses, it allows access to the application.



However, if a request came from one of the blocked IP addresses that you have specified in the web ACL, it is denied access.

Amazon Inspector

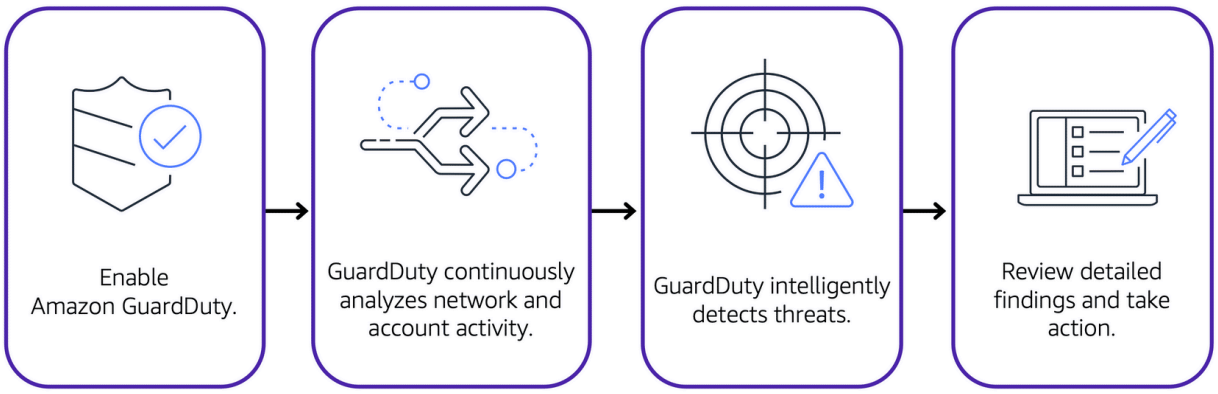
Suppose that the developers at the coffee shop are developing and testing a new ordering application. They want to make sure that they are designing the application in accordance with security best practices. However, they have several other applications to develop, so they cannot spend much time conducting manual assessments. To perform automated security assessments, they decide to use Amazon Inspector.

Amazon Inspector helps to improve the security and compliance of applications by running automated security assessments. It checks applications for security vulnerabilities and deviations from security best practices, such as open access to Amazon EC2 instances and installations of vulnerable software versions.

After Amazon Inspector has performed an assessment, it provides you with a list of security findings. The list prioritizes by severity level, including a detailed description of each security issue and a recommendation for how to fix it. However, AWS does not guarantee that following the provided recommendations resolves every potential security issue. Under the shared responsibility model, customers are responsible for the security of their applications, processes, and tools that run on AWS services.

Amazon GuardDuty

Amazon GuardDuty is a service that provides intelligent threat detection for your AWS infrastructure and resources. It identifies threats by continuously monitoring the network activity and account behavior within your AWS environment.



After you have enabled GuardDuty for your AWS account, GuardDuty begins monitoring your network and account activity. You do not have to deploy or manage any additional security software. GuardDuty then continuously analyzes data from multiple AWS sources, including VPC Flow Logs and DNS logs.

If GuardDuty detects any threats, you can review detailed findings about them from the AWS Management Console. Findings include recommended steps for remediation. You can also configure AWS Lambda functions to take remediation steps automatically in response to GuardDuty's security findings.

Amazon CloudWatch

Amazon CloudWatch is a web service that enables you to monitor and manage various metrics and configure alarm actions based on data from those metrics.

CloudWatch uses metrics to represent the data points for your resources. AWS services send metrics to CloudWatch. CloudWatch then uses these metrics to create graphs automatically that show how performance has changed over time.

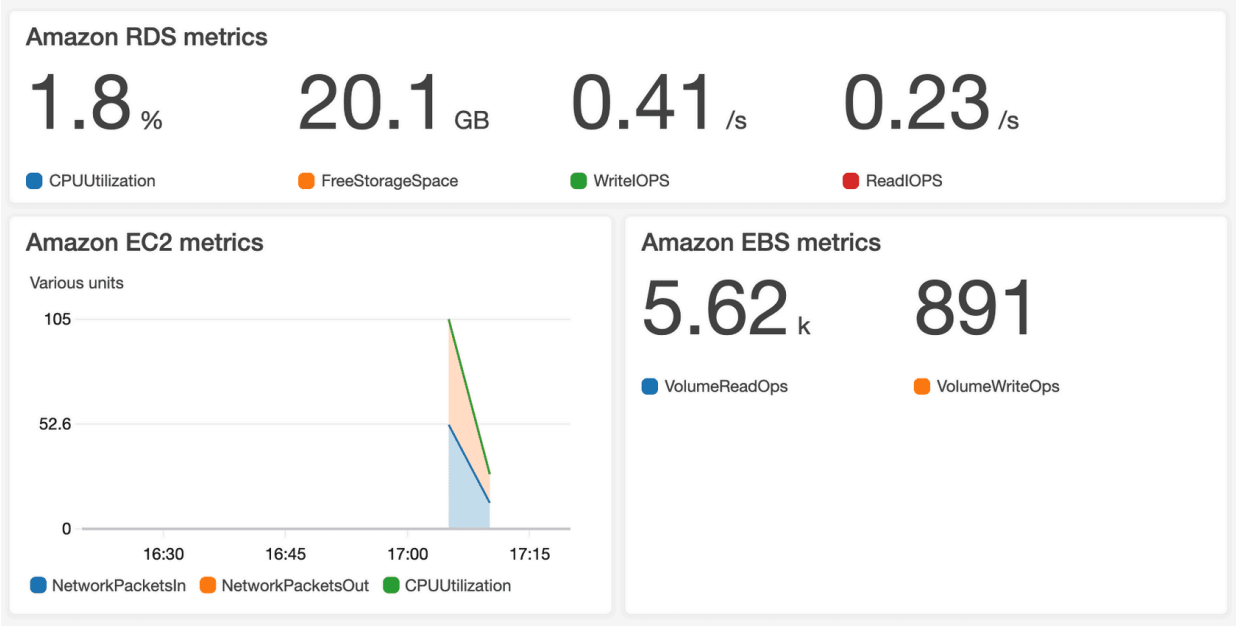
CloudWatch Alarms

With CloudWatch, you can create alarms that automatically perform actions if the value of your metric has gone above or below a predefined threshold.

For example, suppose that your company's developers use Amazon EC2 instances for application development or testing purposes. If the developers occasionally forget to stop the instances, the instances will continue to run and incur charges.

In this scenario, you could create a CloudWatch alarm that automatically stops an Amazon EC2 instance when the CPU utilization percentage has remained below a certain threshold for a specified period. When configuring the alarm, you can specify to receive a notification whenever this alarm is triggered.

CloudWatch Dashboard



The CloudWatch dashboard feature enables you to access all the metrics for your resources from a single location. For example, you can use a CloudWatch dashboard to monitor the CPU utilization of an Amazon EC2 instance, the total number of

requests made to an Amazon S3 bucket, and more. You can even customize separate dashboards for different business purposes, applications, or resources.

AWS CloudTrail

AWS CloudTrail records API calls for your account. The recorded information includes the identity of the API caller, the time of the API call, the source IP address of the API caller, and more. You can think of CloudTrail as a "trail" of breadcrumbs (or a log of actions) that someone has left behind them.

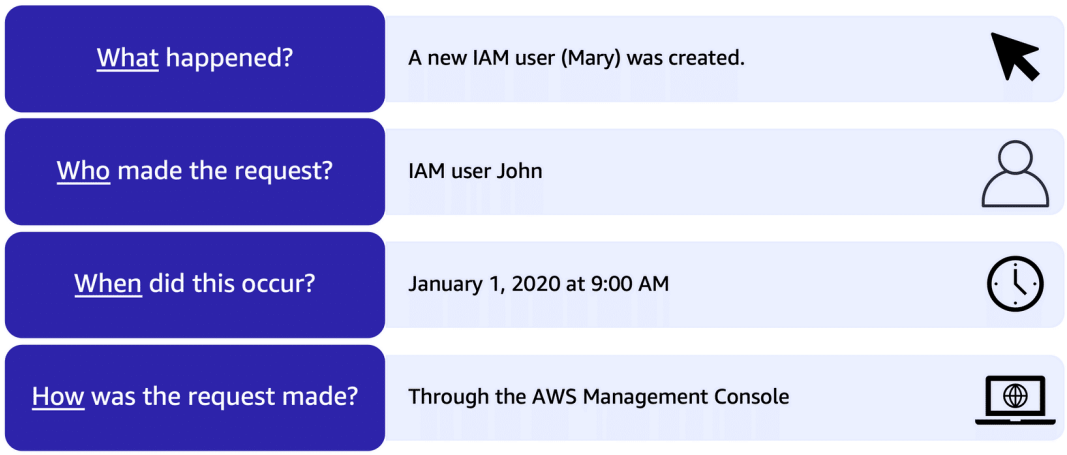
Recall that you can use API calls to provision, manage, and configure your AWS resources. With CloudTrail, you can view a complete history of user activity and API calls for your applications and resources.

Events are typically updated in CloudTrail within 15 minutes after an API call. You can filter events by specifying the time and date that an API call occurred, the user who requested the action, the type of resource that was involved in the API call, and more.

Example: AWS CloudTrail Event

Suppose that the coffee shop owner is browsing through the AWS Identity and Access Management (IAM) section of the AWS Management Console. They discover that a new IAM user named Mary was created, but they do not who, when, or which method created the user.

To answer these questions, the owner navigates to AWS CloudTrail.



In the CloudTrail Event History section, the owner applies a filter to display only the events for the "CreateUser" API action in IAM. The owner locates the event for the

API call that created an IAM user for Mary. This event record provides complete details about what occurred:

On January 1, 2020 at 9:00 AM, IAM user John created a new IAM user (Mary) through the AWS Management Console.

CloudTrail Insights

Within CloudTrail, you can also enable CloudTrail Insights. This optional feature allows CloudTrail to automatically detect unusual API activities in your AWS account.

For example, CloudTrail Insights might detect that a higher number of Amazon EC2 instances than usual have recently launched in your account. You can then review the full event details to determine which actions you need to take next.

AWS Trusted Advisor

AWS Trusted Advisor is a web service that inspects your AWS environment and provides real-time recommendations in accordance with AWS best practices.

Trusted Advisor compares its findings to AWS best practices in five categories: cost optimization, performance, security, fault tolerance, and service limits. For the checks in each category, Trusted Advisor offers a list of recommended actions and additional resources to learn more about AWS best practices.

The guidance provided by AWS Trusted Advisor can benefit your company at all stages of deployment. For example, you can use AWS Trusted Advisor to assist you while you are creating new workflows and developing new applications. Or you can use it while you are making ongoing improvements to existing applications and resources.

AWS Trusted Advisor Dashboard

When you access the Trusted Advisor dashboard on the AWS Management Console, you can review completed checks for cost optimization, performance, security, fault tolerance, and service limits.

The AWS Cloud Adoption Framework (CAF) is like a roadmap for moving your business to the cloud. Imagine you're moving houses:

- **Instead of packing boxes randomly, you need a plan.** CAF helps you plan your cloud journey.

- **It breaks down the move into manageable parts.** These parts are called "perspectives."

- **These perspectives cover different aspects of your business:**
    - **Business:** How will the cloud help your business goals?
    - **People:** Do your employees have the skills for the cloud?
    - **Governance:** How will you manage and control your cloud?
    - **Platform:** What technology will you use in the cloud?
    - **Security:** How will you keep your data safe in the cloud?
    - **Operations:** How will you run and maintain your cloud?

- **CAF provides guidance and best practices for each perspective.** It helps you avoid common pitfalls and make the most of the cloud.

- **Essentially, it's a structured way to make sure your cloud move is successful.**

Migration Strategies

6 Strategies for Migration

When migrating applications to the cloud, six of the most common migration strategies that you can implement are:

- Rehosting
- Replatforming
- Refactoring/re-architecting
- Repurchasing
- Retaining
- Retiring

Rehosting

Rehosting also known as “lift-and-shift” involves moving applications without changes.

In the scenario of a large legacy migration, in which the company is looking to implement its migration and scale quickly to meet a business case, the majority of applications are rehosted.

Replatforming

Replatforming, also known as “lift, tinker, and shift,” involves making a few cloud optimizations to realize a tangible benefit. Optimization is achieved without changing the core architecture of the application.

Refactoring/re-architecting

Refactoring (also known as re-architecting) involves reimagining how an application is architected and developed by using cloud-native features. Refactoring is driven by a strong business need to add features, scale, or performance that would otherwise be difficult to achieve in the application’s existing environment.

Repurchasing

Repurchasing involves moving from a traditional license to a software-as-a-service model.

For example, a business might choose to implement the repurchasing strategy by migrating from a customer relationship management (CRM) system to Salesforce.com.

Retaining

Retaining consists of keeping applications that are critical for the business in the source environment. This might include applications that require major refactoring before they can be migrated, or, work that can be postponed until a later time.

Retiring

Retiring is the process of removing applications that are no longer needed.

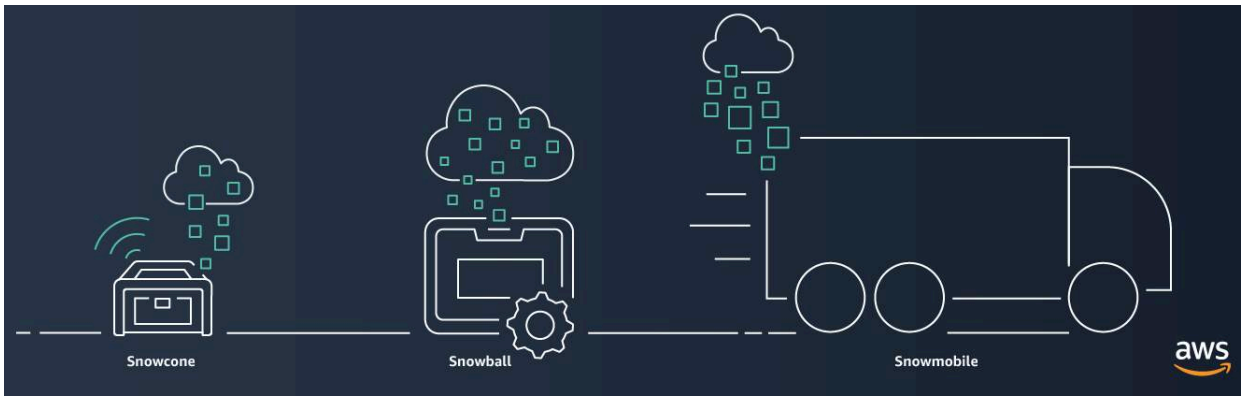
The AWS Snow Family is essentially a set of physical devices that Amazon Web Services (AWS) provides to help you move large amounts of data into and out of the AWS cloud. Think of them as rugged, secure, and portable storage and compute units. Here's a breakdown:

- **Why they exist:**
  - Sometimes, you have so much data that it's impractical or impossible to transfer it over the internet.
  - Other times, you need to process data in places where you don't have reliable internet access (like remote locations).

- **What they do:**
  - They allow you to physically transport large datasets to AWS.
  - Some devices also offer on-board computing, so you can process data locally.

- **The "family" members:**
  - **AWS Snowcone:**
    - A small, portable device for edge computing and data transfer in tight spaces.
  - **AWS Snowball:**
    - A larger, rugged device for petabyte-scale data migration. It comes in different versions optimized for storage or compute.
  - **AWS Snowmobile:**
    - A massive, truck-mounted storage container for exabyte-scale data transfers.

In essence, the AWS Snow Family solves the problem of moving massive amounts of data when network connectivity is limited or non-existent.



Innovate with AWS

When examining how to use AWS services, it is important to focus on the desired outcomes. You are properly equipped to drive innovation in the cloud if you can clearly articulate the following conditions:

- The current state
- The desired state
- The problems you are trying to solve

Consider some of the paths you might explore in the future as you continue on your cloud journey.

Serverless Applications

With AWS, serverless refers to applications that don't require you to provision, maintain, or administer servers. You don't need to worry about fault tolerance or availability. AWS handles these capabilities for you.

AWS Lambda is an example of a service that you can use to run serverless applications. If you design your architecture to trigger Lambda functions to run your code, you can bypass the need to manage a fleet of servers.

Building your architecture with serverless applications enables your developers to focus on their core product instead of managing and operating servers.

Artificial Intelligence

AWS offers a variety of services powered by artificial intelligence (AI).

For example, you can perform the following tasks:

- Convert speech to text with Amazon Transcribe.
- Discover patterns in text with Amazon Comprehend.
- Identify potentially fraudulent online activities with Amazon Fraud Detector.
- Build voice and text chatbots with Amazon Lex.

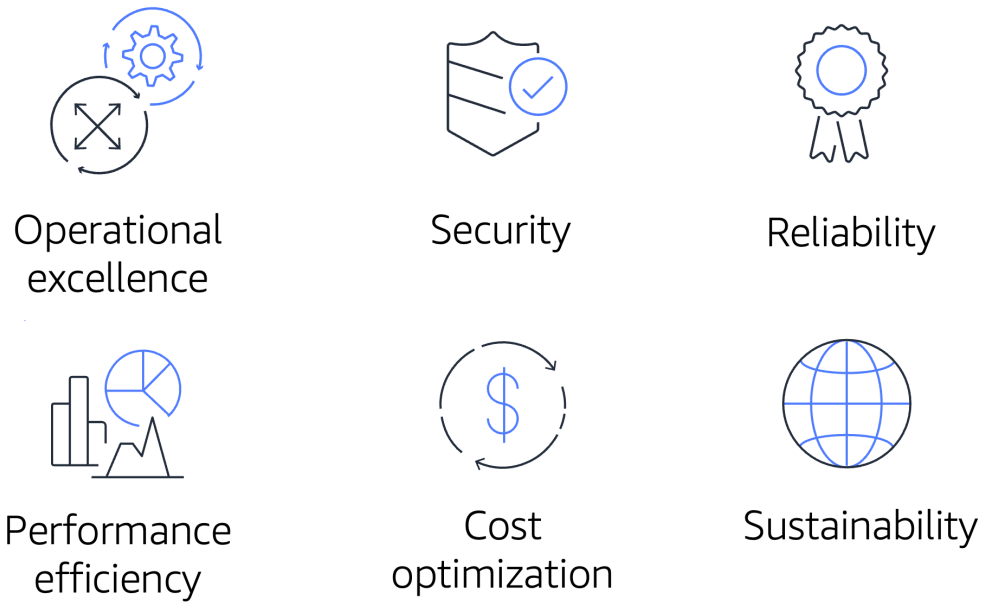
Machine Learning

Traditional machine learning (ML) development is complex, expensive, time consuming, and error prone. AWS offers Amazon SageMaker to remove the difficult work from the process and empower you to build, train, and deploy ML models quickly.

You can use ML to analyze data, solve complex problems, and predict outcomes before they happen.

The AWS Well-Architected Framework

The AWS Well-Architected Framework helps you understand how to design and operate reliable, secure, efficient, and cost-effective systems in the AWS Cloud. It provides a way for you to consistently measure your architecture against best practices and design principles and identify areas for improvement.



Operational Excellence

Operational excellence is the ability to run and monitor systems to deliver business value and to continually improve supporting processes and procedures.

Design principles for operational excellence in the cloud include performing operations as code, annotating documentation, anticipating failure, and frequently making small, reversible changes.

Security

The Security pillar is the ability to protect information, systems, and assets while delivering business value through risk assessments and mitigation strategies.

When considering the security of your architecture, apply these best practices:

- Automate security best practices when possible.
- Apply security at all layers.
- Protect data in transit and at rest.

Reliability

Reliability is the ability of a system to do the following:

- Recover from infrastructure or service disruptions
- Dynamically acquire computing resources to meet demand
- Mitigate disruptions such as misconfigurations or transient network issues

Reliability includes testing recovery procedures, scaling horizontally to increase aggregate system availability, and automatically recovering from failure.

Performance Efficiency

Performance efficiency is the ability to use computing resources efficiently to meet system requirements and to maintain that efficiency as demand changes and technologies evolve.

Evaluating the performance efficiency of your architecture includes experimenting more often, using serverless architectures, and designing systems to be able to go global in minutes.

Cost Optimization

Cost optimization is the ability to run systems to deliver business value at the lowest price point. Cost optimization includes adopting a consumption model, analyzing and attributing expenditure, and using managed services to reduce the cost of ownership.

Sustainability

In December 2021, AWS introduced a sustainability pillar as part of the AWS Well-Architected Framework. Sustainability is the ability to continually improve sustainability impacts by reducing energy consumption and increasing efficiency across all components of a workload by maximizing the benefits from the provisioned resources and minimizing the total resources required.

To facilitate good design for sustainability:

- Understand your impact
- Establish sustainability goals
- Maximize utilization
- Anticipate and adopt new, more efficient hardware and software offerings
- Use managed services
- Reduce the downstream impact of your cloud workloads