E-LEARNING

AWS Cloud Practitioner Essentials

Module 1 – Introduction to Amazon Web Services

Contents

Learning objectives 1

What is a client-server model? 1

Deployment models for cloud computing 2

Benefits of cloud computing 3

Quiz 4

Additional resources 4

Learning objectives

In this module, you will learn how to:

- Summarize the benefits of AWS.
- Describe differences between on-demand delivery and cloud deployments.
- Summarize the pay-as-you-go pricing model.

What is a client-server model?

You just learned more about AWS and how almost all of modern computing uses a basic client-server model. Let's recap what a client-server model is.

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 fulfils it by returning the information to the client.

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 are cloud-based, on-premises, and hybrid.

Cloud-Based Deployment

- 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 that are fully based in the cloud.

On-Premises Deployment

- 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

- 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**.

For example, suppose that a company wants to use cloud services that can automate batch data processing and analytics. However, the company has several legacy applications that are more suitable on premises and will not be migrated to the cloud. With a hybrid deployment, the company would be able to keep the legacy applications on premises while benefiting from the data and analytics services that run in the cloud.

Benefits of cloud computing

Consider why a company might choose to take a particular cloud computing approach when addressing business needs.

Trade upfront expense for variable expense

Upfront expense refers to data centres, physical servers, and other resources that you would need to invest in before using them. Variable expense means you only pay for computing resources you consume instead of investing heavily in data centres and servers before you know how you're going to use them.

By taking a cloud computing approach that offers the benefit of variable expense, companies can implement innovative solutions while saving on costs.

Stop spending money to run and maintain data centres

Computing in data centres often requires you to spend more money and time managing infrastructure and servers.

A benefit of cloud computing is the ability to focus less on these tasks and more on your applications and customers.

Stop guessing capacity

With cloud computing, you don't have to predict how much infrastructure capacity you will need before deploying an application.

For example, you can launch Amazon EC2 instances when needed, and pay only for the compute time you use. Instead of paying for unused resources or having to deal with limited capacity, you can access only the capacity that you need. You can also scale in or scale out in response to demand.

Benefit from massive economies of scale

By using cloud computing, you can achieve a lower variable cost than you can get on your own.

Because usage from hundreds of thousands of customers can aggregate in the cloud, providers, such as AWS, can achieve higher economies of scale. The economy of scale translates into lower pay-as-you-go prices.

Increase speed and agility

The flexibility of cloud computing makes it easier for you to develop and deploy applications. This flexibility provides you with more time to experiment and innovate. When computing in data centres, it may take weeks to obtain new resources that you need. By comparison, cloud computing enables you to access new resources within minutes.

Go global in minutes

The global footprint of the AWS Cloud enables you to deploy applications to customers around the world quickly, while providing them with **low latency**. This means that even if you are located in a different part of the world than your customers, customers are able to access your applications with minimal delays. Later in this course, you will explore the AWS global infrastructure in greater detail. You will examine some of the services that you can use to deliver content to customers around the world.

Quiz

1) What is cloud computing?
    - On-demand delivery of IT resources and applications through the internet with pay-as-you-go pricing

2) What is another name for on-premises deployment?
    - Private cloud deployment

3) How does the scale of cloud computing help you to save costs?
    - The aggregated cloud usage from a large number of customers results in lower pay-as-you-go prices.

Additional resources

To learn more about the concepts that were explored in Module 1, review these resources.

- AWS glossary
- Whitepaper: Overview of Amazon Web Services
- AWS Fundamentals: Overview
- What is cloud computing?
- Types of cloud computing
- Cloud computing with AWS

E-LEARNING

AWS Cloud Practitioner Essentials

Module 2 – Compute in the Cloud

Contents

Learning objectives 2
Amazon Elastic Compute Cloud (Amazon EC2) 2
Amazon EC2 instance types 4
Amazon EC2 pricing 5
Scalability 7
Amazon EC2 Auto Scaling 7
Example: Amazon EC2 Auto Scaling 8
Elastic Load Balancing 9
Example: Elastic Load Balancing 10
Messaging and Queuing 11
Monolithic applications and microservices 12
Amazon Simple Queue Service (Amazon SQS) 13
Amazon Simple Notification Service (Amazon SNS) 14
Serverless computing 16
AWS Lambda 17
How AWS Lambda works 17
Containers 18
Amazon Elastic Container Service (Amazon ECS) 19
Amazon Elastic Kubernetes Service (Amazon EKS) 19
AWS Fargate 20
Summary 20
Quiz 21
Additional resources 22

Learning objectives

In this module, you will learn how to:

- Describe the benefits of Amazon EC2 at a basic level.
- Identify the different Amazon EC2 instance types.
- Differentiate between the various billing options for Amazon EC2.
- Summarize the benefits of Amazon EC2 Auto Scaling.
- Summarize the benefits of Elastic Load Balancing.
- Give an example of the uses for Elastic Load Balancing.
- Summarize the differences between Amazon Simple Notification Service (Amazon SNS) and Amazon Simple Queue Service (Amazon SQS).
- Summarize additional AWS compute options.

Amazon Elastic Compute Cloud (Amazon EC2)

Amazon Elastic Compute Cloud (Amazon EC2) provides secure, resizable compute capacity in the cloud as Amazon EC2 instances.

Imagine you are responsible for the architecture of your company's resources and need to support new websites. With traditional on-premises resources, you have to do the following:

- Spend money upfront to purchase hardware.
- Wait for the servers to be delivered to you.
- Install the servers in your physical data centre.
- Make all the necessary configurations.

The time and money it takes to get up and running with on-premises resources is fairly high. When you own your own fleet of physical servers, you first have to do a bunch of research to see what type of servers you want to buy and how many you'll need. Then you purchase that hardware up front. You'll wait for multiple weeks or months for a vendor to deliver those servers to you. You then take them to a data centre that you own or rent to install them, rack and stack them, and wire them all up. Then you make sure that they are secure and powered up and then they're ready to be used. Only then can you begin to host your applications on top of these servers. The worst part is, once you buy these servers you are stuck with them whether you use them or not.

By comparison, with an Amazon EC2 instance you can use a virtual server to run applications in the AWS Cloud.

- You can provision and launch an Amazon EC2 instance within minutes.
- You can stop using it when you have finished running a workload.
- You pay only for the compute time you use when an instance is running, not when it is stopped or terminated.

- You can save costs by paying only for server capacity that you need or want.

How Amazon EC2 works

(1) 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.

(2) 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.

(3) 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.

Multi-tenancy

EC2 runs on top of physical host machines managed by AWS using virtualization technology. When you spin up an EC2 instance, you aren't necessarily taking an entire host to yourself. Instead, you are sharing the host with multiple other instances, otherwise known as virtual machines. And a **hypervisor running on the host machine** is responsible for **sharing the underlying physical resources between the virtual machines**.

This idea of **sharing underlying hardware** is called **multi-tenancy**. The hypervisor is responsible for coordinating this **multi-tenancy** and it is managed by AWS. The hypervisor is responsible for isolating the virtual machines from each other as they share resources from the host. This means EC2 instances are secure. Even though they may be sharing resources, one EC2 instance is not aware of any other EC2 instances also on that host. They are secure and separate from each other.

Amazon EC2 instance types

Amazon EC2 instance types are **optimized for different tasks**. When selecting an instance type, consider the specific needs of your workloads and applications. This might include requirements for compute, memory, or storage capabilities.

General purpose instances provide a balance of compute, memory, and networking resources. You can use them for a variety of workloads, such as:

- application servers
- gaming servers
- backend servers for enterprise applications
- small and medium databases

Suppose that you have an application in which the resource needs for compute, memory, and networking are roughly equivalent. You might consider running it on a general purpose instance because the application does not require optimization in any single resource area.

Compute optimized instances are ideal for compute-bound applications that benefit from **high-performance processors**. Like general purpose instances, you can use compute optimized instances for workloads such as web, application, and gaming servers.

However, the difference is compute optimized applications are ideal for high-performance web servers, **compute-intensive applications servers, and dedicated gaming servers**. You can also use compute optimized instances for **batch processing workloads** that require **processing many transactions** in a single group.

Memory optimized instances are designed to deliver fast performance for workloads that **process large datasets in memory**. In computing, memory is a temporary storage area. It holds all the data and instructions that a central processing unit (CPU) needs to be able to complete actions. Before a computer program or application is able to run, it is loaded from storage into memory. This preloading process gives the CPU direct access to the computer program.

Suppose that you have a workload that requires **large amounts of data to be preloaded** before running an application. This scenario might be a high-performance database or a workload that involves performing real-time processing of a large amount of unstructured data. In these types of use cases, consider using a memory optimized instance. Memory optimized instances enable you to run workloads with high memory needs and receive great performance.

Accelerated computing instances use hardware accelerators, or coprocessors, to perform some functions more efficiently than is possible in software running on CPUs. Examples of these functions include **floating-point number calculations, graphics processing, and data pattern matching.**

In computing, a **hardware accelerator** is a component that can expedite data processing. Accelerated computing instances are ideal for workloads such as **graphics applications, game streaming, and application streaming.**

Storage optimized instances are designed for workloads that require **high, sequential read and write access to large datasets on local storage**. Examples of workloads suitable for storage optimized instances include **distributed file systems, data warehousing applications, and high-frequency online transaction processing (OLTP)** systems.

In computing, the term input/output operations per second (**IOPS**) is a metric that measures the performance of a storage device. It indicates how many different input or output operations a device can perform in one second. Storage optimized instances are designed to deliver tens of thousands of low-latency, random IOPS to applications.

You can think of input operations as data put into a system, **such as records entered into a database**. An output operation is data generated by a server. An example of output might be the analytics performed on the records in a database. If you have an application that has a high IOPS requirement, a storage optimized instance can provide better performance over other instance types not optimized for this kind of use case.

Amazon EC2 pricing

With Amazon EC2, you pay only for the compute time that you use. Amazon EC2 offers a variety of pricing options for different use cases. For example, if your use case can withstand interruptions, you can save with Spot Instances. You can also save by committing early and locking in a minimum level of use with Reserved Instances.

On-Demand Instances are ideal for short-term, irregular workloads that cannot be interrupted. **No upfront costs or minimum contracts** apply. The instances run continuously until you stop them, and you pay for only the compute time you use.

Sample use cases for On-Demand Instances include developing and testing applications and running applications that have unpredictable usage patterns. On-Demand Instances are not recommended for workloads that last a year or longer because these workloads can experience greater cost savings using Reserved Instances.

This type of on-demand pricing is usually for when you get started and want to spin up servers to test out workloads and play around. You don't need any prior contracts or communication with AWS to use On-Demand pricing. You can also use them to get a **baseline for your average usage**, which leads us to our next pricing option, Savings Plan.

AWS offers **Savings Plans** for several compute services, including Amazon EC2. **Amazon EC2 Savings Plans** enable you to reduce your compute costs by **committing to a consistent amount of compute usage** (measured in dollars per hour) for a 1-year or 3-year term. This term commitment results in savings of up to 72% over On-Demand costs.

Any usage up to the commitment is charged at the discounted Savings Plan rate (for example, $10 an hour). Any usage beyond the commitment is charged at regular On-Demand rates.

Later in this course, you will review AWS Cost Explorer, a tool that enables you to visualize, understand, and manage your AWS costs and usage over time.

Reserved Instances are a billing discount applied to the use of **On-Demand Instances** in your account, and these are suited for **steady-state workloads or ones with predictable usage**. You can purchase Standard Reserved and Convertible Reserved Instances for a 1-year or 3-year term, and Scheduled Reserved Instances for a 1-year term. You realize greater cost savings with the 3-year option.

At the end of a Reserved Instance term, you can continue using the Amazon EC2 instance without interruption. However, you are charged On-Demand rates until you do one of the following:

- Terminate the instance.
- Purchase a new Reserved Instance that matches the instance attributes (instance type, Region, tenancy, and platform).

Spot Instances are ideal for workloads with **flexible start and end times**, or that can withstand interruptions. They allow you to request spare Amazon EC2 computing capacity for up to 90% off of the On-Demand price. The catch here is that **AWS can reclaim the instance at any time** they need it, giving you a two-minute warning to finish up work and save state. You can always resume later if needed. So when choosing Spot Instances, **make sure your workloads can tolerate being interrupted.** A good example of those are **batch workloads**.

Suppose that you have a background processing job that can start and stop as needed (such as the data processing job for a customer survey). You want to start and stop the processing job without affecting the overall operations of your business. If you make a Spot request and Amazon EC2 capacity is available, your Spot Instance launches. However, if you make a Spot request and Amazon EC2 capacity is unavailable, the request is not successful until capacity becomes available. The unavailable capacity might delay the launch of your background processing job.

After you have launched a Spot Instance, if capacity is no longer available or demand for Spot Instances increases, your instance may be interrupted. This might not pose any issues for your background processing job. However, in the earlier example of developing and testing applications, you would most likely want to avoid unexpected interruptions. Therefore, choose a different EC2 instance type that is ideal for those tasks.

Dedicated Hosts are physical servers with Amazon EC2 instance capacity that is fully dedicated to your use. These are usually for meeting certain compliance requirements and **nobody else will share tenancy of that host**.

You can use your existing per-socket, per-core, or per-VM software licenses to help maintain license compliance. You can purchase On-Demand Dedicated Hosts and Dedicated Hosts Reservations. Of all the Amazon EC2 options that were covered, Dedicated Hosts are the most expensive.

Scalability

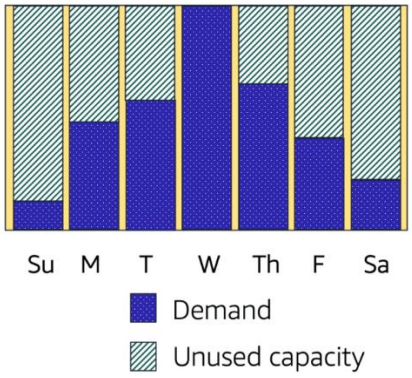
Scalability involves beginning with only the resources you need and designing your architecture to **automatically respond to changing demand by scaling out or in**. As a result, you pay for only the resources you use. You don't have to worry about a lack of computing capacity to meet your customers' needs.

Here is the on-premise data centre dilemma. If your business is like 99% of all businesses out in the world, your customer workloads vary over time: perhaps over a simple 24 hour period, or you might have seasons where you're busy, and weeks that are not in demand. If you're building out a data centre, the question is, what is the right amount of hardware to purchase? If you buy for the average amount, the average usage, you won't be wasting money on average. But when the peak loads come in, you won't have the hardware to service the customers, especially during the critical moments to expect to be making all your results. Now, if you buy for the top max load, you might have happy customers, but for most of the year, you'll have idle resources, which means your average utilization is very low.

If you wanted the scaling process to happen automatically, which AWS service would you use? The AWS service that provides this functionality for Amazon EC2 instances is **Amazon EC2 Auto Scaling**.

Amazon EC2 Auto Scaling

If you've tried to access a website that wouldn't load and frequently timed out, the website might have received more requests than it was able to handle. This situation is similar to waiting in a long line at a coffee shop, when there is only one barista present to take orders from customers.



Amazon EC2 **Auto Scaling enables you to automatically add or remove Amazon EC2 instances in response to changing application demand**. By automatically scaling your instances in and out as needed, you are able to maintain a greater sense of application availability.

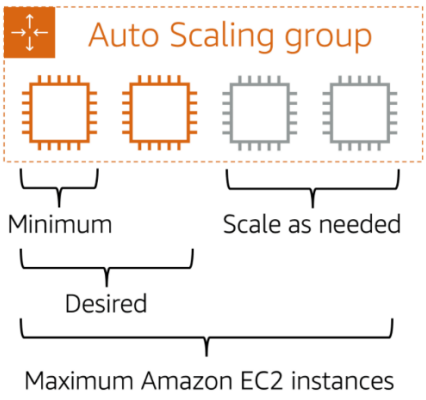
Within Amazon EC2 Auto Scaling, you can use two approaches: dynamic scaling and predictive scaling.

- *Dynamic scaling* responds to changing demand.
- *Predictive scaling* automatically schedules the right number of Amazon EC2 instances based on predicted demand.

Example: Amazon EC2 Auto Scaling

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. 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.

Elastic Load Balancing

Elastic Load Balancing (ELB) 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.

It is engineered to address the undifferentiated heavy lifting of load balancing. **Elastic Load Balancing is a Regional construct**, and we'll explain more of what that means in later videos. But the key value for you is that because it **runs at the Region level** rather than on individual EC2 instances, the service is **automatically highly available** with no additional effort on your part.

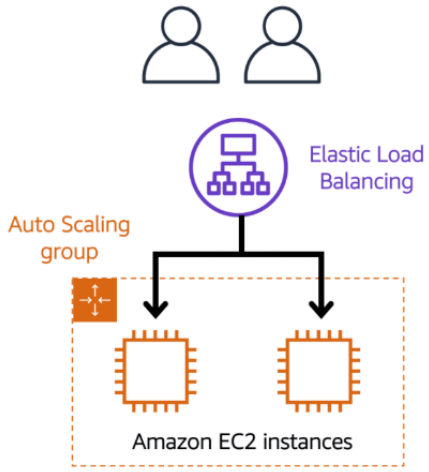
Although Elastic Load Balancing and Amazon EC2 Auto Scaling are separate services, they work together to help ensure that applications running in Amazon EC2 can provide high performance and availability.

ELB is automatically scalable. As your traffic grows, ELB is designed to handle the additional throughput with no change to the hourly cost. When your EC2 fleet auto-scales out, as each instance comes online, the auto-scaling service just lets the Elastic Load Balancing service know that it's ready to handle the traffic, and off it goes. Once the fleet scales in, ELB first

stops all new traffic, and waits for the existing requests to complete, to drain out. Once they do that, then the auto-scaling engine can terminate the instances without disruption to existing customers.

Because **ELB is regional**, it's a **single URL** that each front end instance uses. Then the ELB directs traffic to the back end that has the least outstanding requests. Now, if the back end scales, once the new instance is ready, it just tells the ELB that it can take traffic and it gets to work. The front end doesn't know and doesn't care how many back end instances are running. This is true **decoupled architecture**.

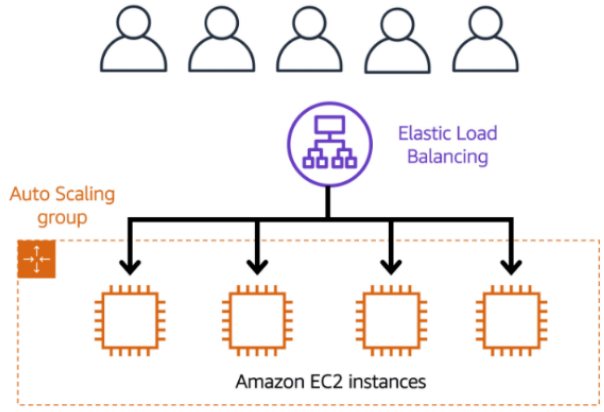
Example: Elastic Load Balancing



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.

Messaging and Queuing

In the coffee shop, there are cashiers taking orders from the customers and baristas making the orders. Currently, the cashier takes the order, writes it down with a pen and paper, and delivers this order to the barista. The barista then takes the paper and makes the order. When the next order comes in, the process repeats. This works great as long as both the cashier and the barista are in sync. But what would happen if the cashier took the order and turned to pass it to the barista and the barista was out on break or busy with another order? Well, that cashier is stuck until the barista is ready to take the order. And at a certain point, the order will probably be dropped so the cashier can go serve the next customer.

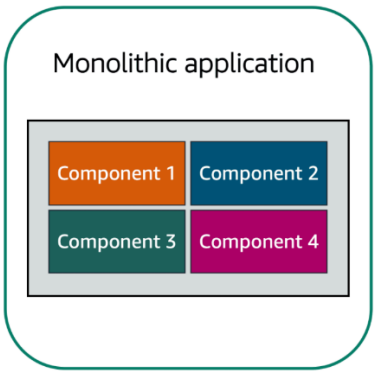
You can see how this is a flawed process, because as soon as either the cashier or barista is out of sync, the process will degrade, causing slowdowns in receiving orders and failures to complete orders at all. A much better process would be to introduce some sort of buffer or queue into the system. Instead of handing the order directly to the barista, the cashier would post the order to some sort of buffer, like **an order board**.

This idea of **placing messages into a buffer is called messaging and queuing**. Just as our cashier sends orders to the barista, applications send messages to each other to communicate. If applications **communicate directly** like our cashier and barista previously, this is called being **tightly coupled**. For example, if we have Application A and it is sending messages directly to Application B, if Application B has a failure and cannot accept those messages, Application A will begin to see errors as well. This is a **tightly coupled architecture.**

A more reliable architecture is **loosely coupled**. This is an architecture where if one component fails, it is isolated and therefore won't cause cascading failures throughout the whole system.

In a message queue, messages are sent into the queue by Application A and they are processed by Application B. If Application B fails, Application A doesn't experience any disruption. Messages being sent can still be **sent to the queue and will remain there until they are eventually processed**. This is **loosely coupled**. This is what we strive to achieve with architectures on AWS. And this brings me to two AWS services that can assist in this regard. Amazon Simple Queue Service or SQS and Amazon Simple Notification Service or SNS.

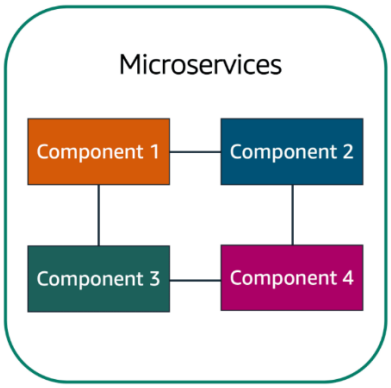
Monolithic applications and microservices



Applications are made of multiple components. The components communicate with each other to transmit data, fulfil 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. 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 fulfil different functions. **Two services facilitate application integration**: Amazon Simple Notification Service (Amazon SNS) and Amazon Simple Queue Service (Amazon SQS).

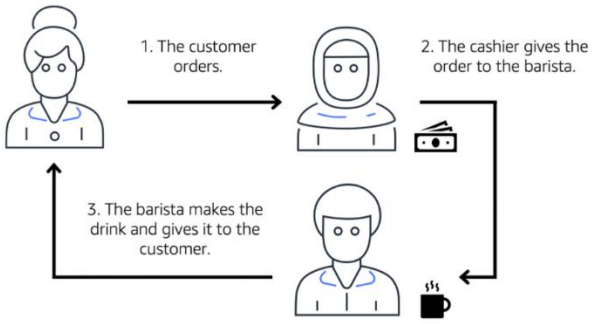
Amazon Simple Queue Service (Amazon SQS)

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.

SQS allows you to send, store, and receive messages between software components at any volume. This is without losing messages or requiring other services to be available. Think of messages as our coffee orders and the order board as an SQS queue. Messages have the person's name, coffee order, and time they ordered. The **data contained within a message is called a payload, and it's protected until delivery**. SQS queues are where messages are placed until they are processed. AWS manages the underlying infrastructure for you to host those queues. These scale automatically, are reliable, and are easy to configure and use.

Step 1 – Fulfilling an Order



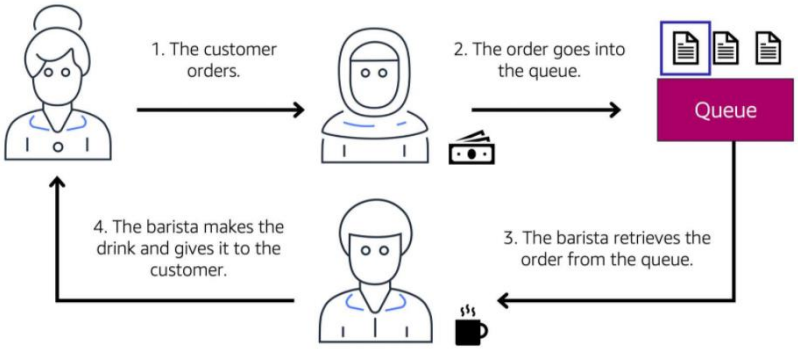
Suppose that the coffee shop has an ordering process in which a cashier takes orders, and a barista makes the orders. Think of the cashier and the barista as two separate components of an application. First, the cashier takes an order and writes it down on a piece of paper. Next, the cashier delivers 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.

Step 2 – 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 complements.

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.

For decoupled applications and microservices, **Amazon SQS enables you to send, store, and retrieve messages between components**. This decoupled approach enables the separate components to work more efficiently and independently.

Note that Amazon SQS is a **message queuing service**, and is therefore not the best choice for **publishing messages to subscribers** since it does **not use the message subscription and topic model that is involved with Amazon SNS**.

Amazon Simple Notification Service (Amazon SNS)

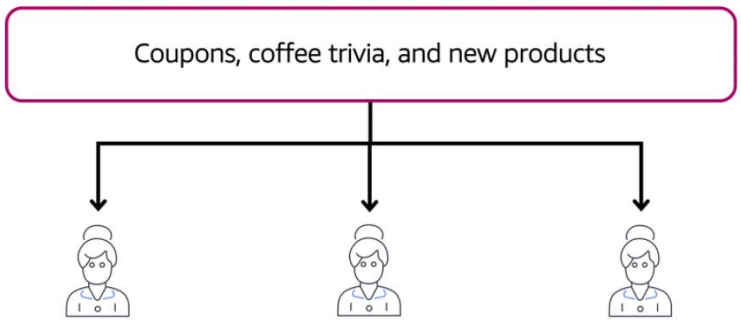
Amazon Simple Notification Service (Amazon SNS) is a **publish/subscribe** service. Using Amazon SNS **topics**, a **publisher publishes messages to subscribers**. This is similar to the coffee shop; the cashier provides coffee orders to the barista who makes the drinks.

Amazon SNS is similar to SQS in that it is used to send out messages to services, but it can **also send out notifications to end users**. It does this in a different way called a

publish/subscribe or **pub/sub model**. This means that you can create something called an **SNS topic which is just a channel for messages** to be delivered.

Additionally, SNS can be used to **fan out notifications to end users using mobile push, SMS, and email.** Taking this back to our coffee shop, we could send out a notification when a customer's order is ready. This could be a simple SMS to let them know to pick it up or even a mobile push.

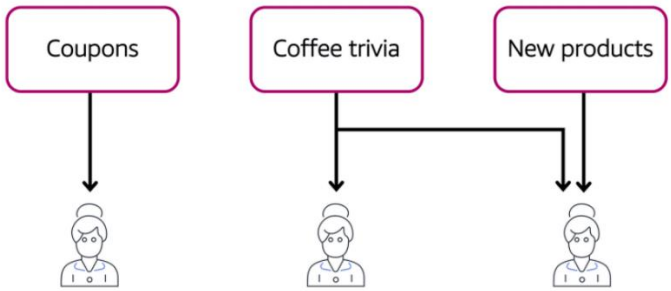
Step 1 – Publishing updates from a single topic



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.

Step 2 – 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.

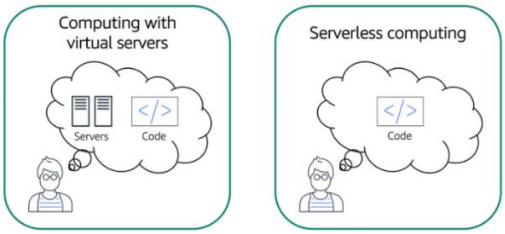
Although this example from the coffee shop involves subscribers who are people, in Amazon SNS, **subscribers can be web servers, email addresses, AWS Lambda functions, or several other options.**

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.

EC2 requires that you set up and manage your fleet of instances over time. When you're using EC2, **you are responsible for patching your instances** when new software packages come out, **setting up the scaling** of those instances as well as ensuring that you've architected your solutions to be hosted in a highly available manner. This is still not as much management as you would have if you hosted these on-premises. But management processes will still need to be in place.



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. Serverless means that you **cannot actually see or access the underlying infrastructure** or instances that are hosting your application. Instead, all the **management of the underlying environment from a provisioning, scaling, high availability, and maintenance perspective are taken care of** for you. All you need to do is focus on your application and the rest is taken care of.

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**.

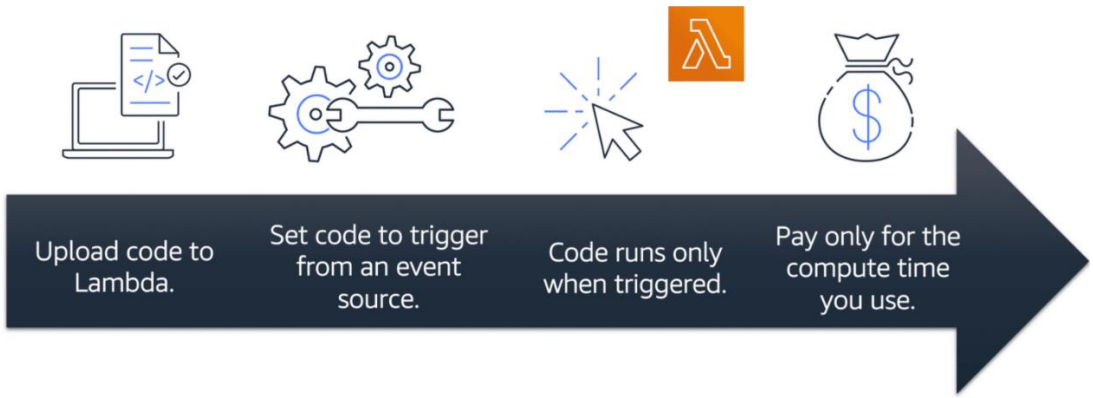
While using AWS Lambda, you pay only for the compute time that you consume. Charges apply only when your code is running. You can also run code for virtually any type of application or backend service, all with zero administration.

Lambda's a service that allows you to upload your code into what's called a **Lambda function**. Configure a trigger and from there, the service waits for the trigger. When the trigger is detected, the code is automatically run in a managed environment, an environment you do not need to worry too much about because it is **automatically scalable, highly available and all of the maintenance in the environment itself is done by AWS**. If you have one or 1,000 incoming triggers, Lambda will scale your function to meet demand.

Lambda is designed to **run code under 15 minutes** so this **isn't for long running processes like deep learning.** It's **more suited for quick processing** like a **web backend, handling requests or a backend expense report processing** service where each invocation takes less than 15 minutes to complete.

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.

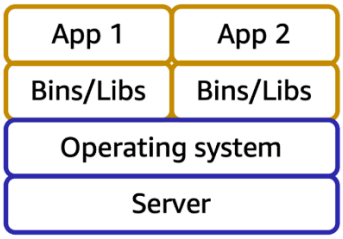
Containers

If you weren't quite ready for serverless yet or you need access to the underlying environment, but still want efficiency and portability, you should look at AWS container services like Amazon Elastic Container Service, otherwise known as ECS. Or Amazon Elastic Kubernetes Service, otherwise known as EKS.

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.

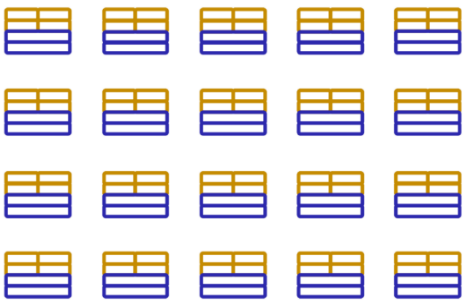
A container in this case is a **Docker container**. Docker is a widely used platform that uses **operating system level virtualization to deliver software in containers**. Now a container is a package for your code where you package up your application, its dependencies as well as any configurations that it needs to run. **These containers run on top of EC2 instances** and run in isolation from each other similar to how virtual machines work, but in this case, the host is an EC2 instance.

Step 1 – 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.

Step 2 – 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.

Container orchestration services help you to deploy, manage, and scale your containerized applications. Next, you will learn about two services that provide container orchestration: Amazon Elastic Container Service and Amazon Elastic Kubernetes Service.

When you use Docker containers on AWS, you need processes to start, stop, restart, and monitor containers running across not just one EC2 instance, but a number of containers together which is called a **cluster**.

The process of doing these tasks is called **container orchestration** and it turns out it's really hard to do on your own. **Orchestration tools** were created to help you manage your containers. **ECS** is designed to help you run your containerized applications at scale **without the hassle of managing your own container orchestration software**. **EKS does a similar thing**, but uses different tooling and with different features.

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.

Amazon Elastic Kubernetes Service (Amazon EKS)

Amazon Elastic Kubernetes Service (Amazon EKS) is a fully managed service that you can use to run Kubernetes on AWS.

Kubernetes is open-source software that enables you to deploy and manage containerized applications at scale. A large community of volunteers maintains Kubernetes, and AWS actively works together with the Kubernetes community. As new features and functionalities release for Kubernetes applications, you can easily apply these updates to your applications managed by Amazon EKS.

AWS Fargate

Both Amazon ECS and Amazon EKS can run on top of EC2. **But if you don't want to even think about using EC2s to host your containers** because you either don't need access to the underlying OS or you don't want to manage those EC2 instances, you can use a compute platform called AWS Fargate. **AWS Fargate** is a serverless compute engine for containers. It works with both Amazon ECS and Amazon EKS.

When using AWS Fargate, you do not need to provision or manage servers. AWS Fargate manages your server infrastructure for you. You can focus more on innovating and developing your applications, and you pay only for the resources that are required to run your containers.

Choice of Compute Service

If you are trying to host traditional applications and want full access to the underlying operating system like Linux or Windows, you are going to want to use **EC2**.

If you are looking to host short running functions, service-oriented or event driven applications and you don't want to manage the underlying environment at all, look into the serverless **AWS Lambda**.

If you are looking to run Docker container-based workloads on AWS, you first need to choose your orchestration tool. Do you want to use Amazon **ECS** or Amazon **EKS**? After you choose your tool, you then need to choose your platform. Do you want to run your containers on EC2 instances that you manage or in a serverless environment like AWS **Fargate** that is managed for you?

Summary

We define cloud computing as the on-demand delivery of IT resources over the internet with pay-as-you-go pricing. This means that you can make requests for IT resources like compute, networking, storage, analytics, or other types of resources, and then they're made available for you on demand. You don't pay for the resource upfront. Instead, you just provision and pay at the end of the month.

AWS offers services and many categories that you use together to create your solutions. We've only covered a few services so far, you learned about Amazon EC2. With **EC2**, you can **dynamically spin up and spin down virtual servers called EC2 instances**. When you launch an EC2 instance, you choose the instance family. The instance family determines the **hardware** the instance runs on.

And you can have instances that are built for your specific purpose. The categories are **general** purpose, **compute** optimized, **memory** optimized, **accelerated** computing, and **storage** optimized.

You can **scale** your EC2 instances either **vertically by resizing the instance, or horizontally by launching new instances and adding them to the pool**. You can set up automated **horizontal scaling, using Amazon EC2 Auto Scaling**.

Once you've scaled your EC2 instances out horizontally, you need something to **distribute the incoming traffic across those instances**. This is where the Elastic Load Balancer comes into play.

EC2 instances have different pricing models. There is **On-Demand**, which is the most flexible and has no contract, spot pricing, which allows you to utilize unused capacity at a discounted rate, **Savings Plans or Reserved Instances**, which allow you to enter into a contract with AWS to get a discounted rate when you commit to a certain level of usage, and Savings Plans which apply to AWS **Lambda**, and AWS **Fargate**, as well as EC2 instances.

We also covered messaging services. There is Amazon Simple Queue Service or SQS**. This service allows you to decouple system components. Messages remain in the queue until they are either consumed or deleted.** Amazon Simple Notification Service or SNS, is used for sending messages like emails, text messages, push notifications, or even HTTP requests. Once a message is published, it is sent to all of these subscribers.

You also learned that AWS has different types of compute services beyond just virtual servers like EC2. There are **container services** like Amazon Elastic Container Service, or ECS. And there's Amazon Elastic Kubernetes Service, or EKS. **Both ECS and EKS are container orchestration tools.** You can use these tools with EC2 instances, but if you don't want to manage that, you don't need to. You can use AWS **Fargate**, which allows you **to run your containers on top of a serverless compute platform.**

Then there is AWS Lambda, which allows you to just **upload your code, and configure it to run based on triggers.** And you only get charged for when the code is actually running. No containers, no virtual machines. Just code and configuration.
Hopefully that sums it up. Catch you in the next one.

Quiz

You want to use an Amazon EC2 instance for a batch processing workload. What would be the best Amazon EC2 instance type to use?

- Compute optimized instance type

What are the contract length options for Amazon EC2 Reserved Instances?

- 1 year and 3 years

You have a workload that will run for a total of 6 months and can withstand interruptions. What would be the most cost-efficient Amazon EC2 purchasing option?

- Spot instance

Which process is an example of Elastic Load Balancing?

- Ensuring that no single Amazon EC2 instance has to carry the full workload on its own

You want to deploy and manage containerized applications. Which service can you use?

- Amazon Elastic Kubernetes Service (Amazon EKS). Amazon EKS is a fully managed Kubernetes service. Kubernetes is open-source software that enables you to deploy and manage containerized applications at scale.

Additional resources

To learn more about the concepts that were explored in Module 2, review these resources.

- Compute on AWS
- AWS Compute Blog
- AWS Compute Services
- Hands-On Tutorials: Compute
- Category Deep Dive: Serverless
- AWS Customer Stories: Serverless

Module 3 – Global Infrastructure and Reliability

Contents

Learning objectives 1
Selecting a Region 1
Availability Zones 3
Edge Locations 4
Ways to interact with AWS services 5
AWS Elastic Beanstalk 6
AWS CloudFormation 7
Summary 7
Quiz 8
Additional resources 9

Learning objectives

- Summarize the benefits of the AWS Global Infrastructure.
- Describe the basic concept of Availability Zones.
- Describe the benefits of Amazon CloudFront and edge locations.
- Compare different methods for provisioning AWS services.

Selecting a Region

It's not good enough to have one giant data center where all of the resources are. If something were to happen to that data center, like a power outage or a natural disaster, everyone's applications would go down all at once. You need high availability and fault tolerance. Turns out, it's not even good enough to have two giant data centers. Instead, AWS operates in all sorts of different areas around the world called Regions. **Regions are geographically isolated areas.**

Throughout the globe, AWS builds Regions to be closest to where the business traffic demands. Locations like Paris, Tokyo, Sao Paulo, Dublin, and Ohio**. Inside each Region, we**

have multiple data centers that have all the compute, storage, and other services you need to run your applications.

Each Region can be connected to each other Region through a **high speed fibre network**, controlled by AWS, a truly global operation from corner to corner if you need it to be. Now before we get into the architecture of how each Region is built, it's important to know that you, the business decision maker, gets to choose which Region you want to run out of. And each Region **is isolated from every other Region** in the sense that absolutely **no data goes in or out of your environment in that Region without you explicitly granting permission** for that data to be moved. This is a critical **security** conversation to have.

For example, you might have government compliance requirements that your financial information in Frankfurt cannot leave Germany. Well this is absolutely the way AWS operates outta the box. Any data stored in the Frankfurt Region never leaves the Frankfurt Region, or data in the London region never leaves London, unless you explicitly, with the right credentials and permissions, request the data be exported.

Regional data sovereignty is part of the critical design of AWS Regions. With data being subject to the local laws and statutes of the country where the Region lives.

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. Locating close to your customer base, usually the right call.

Available services within a Region

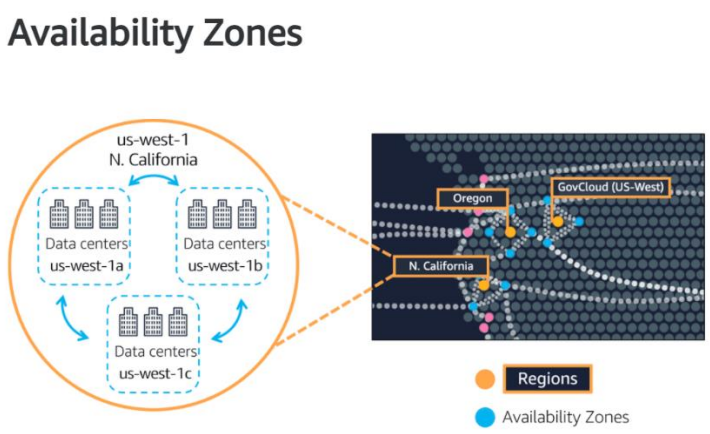
Sometimes, the closest Region might not have all the features that you want to offer to customers. AWS is frequently innovating by creating new services and expanding on features within existing services. However, making new services available around the world sometimes requires AWS to build out physical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

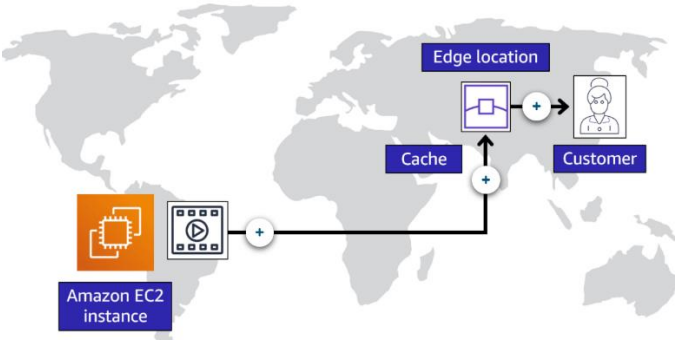


An **Availability Zone** is a **single data center or a group of data centers within a Region**. Availability Zones are located tens of miles apart from each other. This is close enough to have low latency (the time between when content requested and received) between Availability Zones. However, if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

Each Availability Zone is one or more discrete data centers with redundant power, networking, and connectivity. When you launch an Amazon EC2 instance, it launches a virtual machine on a physical hardware that is installed in an Availability Zone. This means **each AWS Region consists of multiple isolated and physically separate Availability Zones within a geographic Region**.

Edge Locations

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



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

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.

Caching copies of data closer to the customers all around the world uses the concept of **content delivery networks**, or **CDNs**. CDNs are commonly used, and on AWS, we call our CDN **Amazon CloudFront**.

Amazon CloudFront is a service that helps deliver data, video, applications, and APIs to customers around the world with low latency and high transfer speeds. Amazon CloudFront uses what are called **Edge locations**, all around the world, to help accelerate communication with users, no matter where they are.

Edge locations are separate from Regions, so you can push content from inside a Region to a collection of Edge locations around the world, in order to accelerate communication and content delivery. AWS Edge locations, also run more than just CloudFront. They run a **domain name service, or DNS, known as Amazon Route 53**, helping direct customers to the correct web locations with reliably low latency.

But what if your business wants to use, AWS services inside their own building? Well sure. AWS can do that for you. Introducing **AWS Outposts**, where AWS will basically install a fully **operational mini Region, right inside your own data center**. That's owned and operated by AWS, using 100% of AWS functionality, but isolated within your own building. It's not a solution most customers need, but if you have specific problems that can only be solved by staying in your own building, AWS Outposts can help.

Ways to interact with AWS services

We've been talking about a few different AWS resources as well as the AWS global infrastructure. You may be wondering, how do I actually interact with these services? And the answer is APIs. **In AWS, everything is an API call**. An API is an application programming interface. And what that means is, there are pre-determined ways for you to interact with AWS services. And 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**. You can use the AWS Management Console, the AWS Command Line Interface, the AWS Software Development Kits, or various other tools like AWS CloudFormation, to create requests to send to AWS APIs to create and manage AWS resources.

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.

Through the console, you can manage your AWS resources visually and in a way that is easy to digest. This is great for getting started and building your knowledge of the services. It's also useful for building out test environments or viewing AWS bills, viewing monitoring and working with other non-technical resources. The AWS Management Console is most likely the first place you will go when you are learning about AWS.

However, once you are up and running in a production type environment, you don't want to rely on the point and click style that the console gives you to create and manage your AWS resources. For example, in order to create an Amazon EC2 Instance, you need to click through various screens, setting all the configurations you want, and then you launch your instance. By having humans do this sort of **manual provisioning**, you're opening yourself up to potential **errors**. It's pretty easy to forget to check a checkbox or misspell something when you are doing everything manually.

The answer to this problem is to use tools that allow you to script or program the API calls.

AWS Command Line Interface (CLI)

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.

The CLI allows you to make API calls using the terminal on your machine. This is different than the visual navigation style of the Management Console. Writing commands using the **CLI makes actions scriptable and repeatable**. So, you can write and run your commands to launch an EC2 Instance. And if you want to launch another, you can just run the pre-written command again. This makes it **less susceptible to human error**. And you can have these scripts run automatically, like on a schedule or triggered by another process.

Automation is very important to having a successful and predictable cloud deployment over time.

AWS Software Development Kits (SDK)

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

There are also other ways you can **manage your AWS environment** using **managed tools** like AWS **Elastic Beanstalk**, and AWS **CloudFormation**.

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

Instead of clicking around the console or writing multiple commands to build out your network, EC2 instances, scaling and Elastic Load Balancers. You can instead **provide your application code and desired configurations to the AWS Elastic Beanstalk service**, which then takes that information and builds out your environment for you.

AWS Elastic Beanstalk also makes it easy to save environment configurations, so they can be deployed again easily. AWS Elastic Beanstalk gives you the convenience of not having to provision and manage all of these pieces separately, while still giving you the visibility and control of the underlying resources.

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 is an **infrastructure as code tool** that allows you to define a wide variety of AWS resources in a **declarative way** using **JSON or YAML** text-based documents called CloudFormation templates. A declarative format like this allows you to define what you want to build without specifying the details of exactly how to build it. CloudFormation lets you define what you want and the CloudFormation engine will worry about the details on calling APIs to get everything built out.

It also isn't just limited to EC2-based solutions. CloudFormation supports many different AWS resources from storage, databases, analytics, machine learning, and more. Once you **define your resources in a CloudFormation template**, CloudFormation will **parse the template and begin provisioning all the resources you defined in parallel**. CloudFormation manages all the calls to the backend AWS APIs for you. You can run the same CloudFormation template in multiple accounts or multiple regions, and it will create identical environments across them. There is less room for human error as it is a totally automated process.

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.

Recap

To recap, the AWS Management Console is great for learning and providing a visual for the user. The AWS Management Console is a manual tool. So right off the bat, it isn't a great option for automation. You can instead use the CLI to script your interactions with AWS using the terminal. You can use the SDKs to write programs to interact with AWS for you or you can use manage tools like AWS Elastic Beanstalk or AWS CloudFormation.

Summary

We covered how logical clusters of data centers make up Availability Zones, Availability Zones in turn make up Regions, and those are spread globally. You then choose what

Regions and Availability Zones you want to operate out of and as a best practice, you should always deploy infrastructure across at least two Availability Zones. And some AWS services like Elastic Load Balancing, Amazon SQS, and Amazon SNS already do this for you.

We also talked about Edge locations and how you can deploy content there to speed up delivery to your customers. We even touched upon edge devices like AWS Outposts which allow you to run AWS infrastructure right in your own data center.

Another thing we discussed was how to provision AWS resources through various options, such as the AWS Management Console, the SDK, CLI, AWS Elastic Beanstalk, and AWS CloudFormation, where you learned how you can set up your infrastructure as code.

I hope you learned how globally available AWS is and how easy it is to get started with provisioning resources.

Quiz

Which statement is TRUE for the AWS global infrastructure?

- A Region consists of two or more Availability Zones. For example, the South America (São Paulo) Region is sa-east-1. It includes three Availability Zones: sa-east-1**a**, sa-east-1**b**, and sa-east-1**c**.

Which factors should be considered when selecting a Region? (Select TWO.)

- Compliance with data governance and legal requirements
- Proximity to your customers

Which statement best describes Amazon CloudFront?

- A global content delivery service. Amazon CloudFront is a content delivery service. It uses a network of edge locations to cache content and deliver content to customers all over the world. When content is cached, it is stored locally as a copy. This content might be video files, photos, webpages, and so on.

Which site does Amazon CloudFront use to cache copies of content for faster delivery to users at any location?

- Edge location

Which action can you perform with AWS Outposts?

- Extend AWS infrastructure and services to your on-premises data center.

Additional resources

Review these resources to learn more about the concepts that were explored in Module 3.

- Global Infrastructure
- Interactive map of the AWS Global Infrastructure
- Regions and Availability Zones
- AWS Networking and Content Delivery Blog
- Tools to Build on AWS
- AWS Customer Stories: Content Delivery

Module 4 – Networking

Contents

Learning objectives 2
Amazon Virtual Private Cloud 2
Internet gateway 3
Virtual private gateway 3
AWS Direct Connect 4
Subnets 5
Network traffic in a VPC 6
Network access control lists (ACLs) 6
Stateless packet filtering 7
Security groups 8
Stateful packet filtering 9
Domain Name System (DNS) 11
Amazon Route 53 12
Example: How Amazon Route 53 and Amazon CloudFront deliver content 13
Summary 14
Quiz 14
Additional resources 15

Learning objectives

In this module, you will learn how to:

- Describe the basic concepts of networking.
- Describe the difference between public and private networking resources.
- Explain a virtual private gateway using a real life scenario.
- Explain a virtual private network (VPN) using a real life scenario.
- Describe the benefit of AWS Direct Connect.
- Describe the benefit of hybrid deployments.
- Describe the layers of security used in an IT strategy.
- Describe the services customers use to interact with the AWS global network.

Amazon Virtual Private Cloud

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)**.

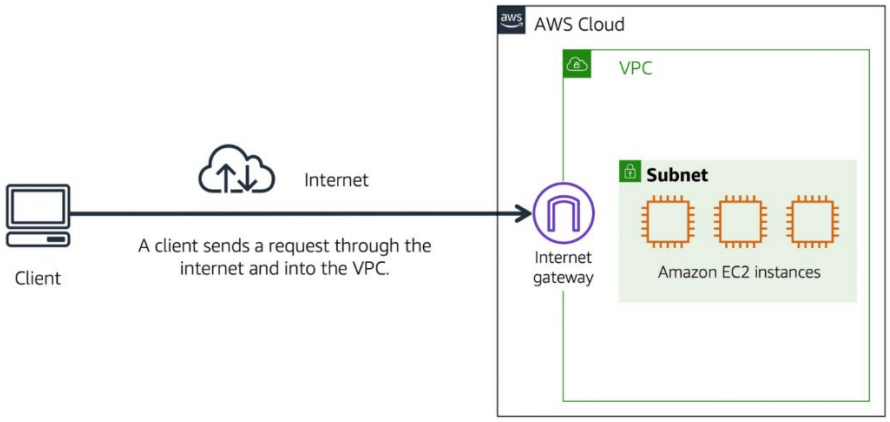
A VPC, or Virtual Private Cloud, is essentially your **own private network** in AWS. A VPC allows you to define your **private IP range for your AWS resources**, and you place things like EC2 instances and ELBs inside of your VPC. Amazon VPC enables you to provision an isolated section of the AWS Cloud. In this isolated section, you can launch resources in a virtual network that you define. These resources can be public facing so they have access to the internet, or private with no internet access, usually for backend services like databases or application servers.

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.
Subnets are chunks of IP addresses in your VPC that allow you to group resources together.

Now, in our coffee shop, we have different employees and one is a cashier. They take customers' orders and thus we want customers to interact with them, so we put them in what we call a public subnet. Hence they can talk to the customers or the internet. But for our baristas, we want them to focus on making coffee and not interact with customers directly, so we put them in a private subnet.

Internet gateway

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



You can think of it as a hardened fortress where nothing goes in or out without explicit permission. You have a gateway on the VPC that only permits traffic in or out of 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.

Virtual private gateway

What if you have a VPC that includes only private resources? For example, you might have resources that you only want to be reachable if someone is logged into your private network. This might be internal services, like an HR application or a backend database. This means we want a private gateway that only allows people in if they are **coming from an approved network**, not the public internet.

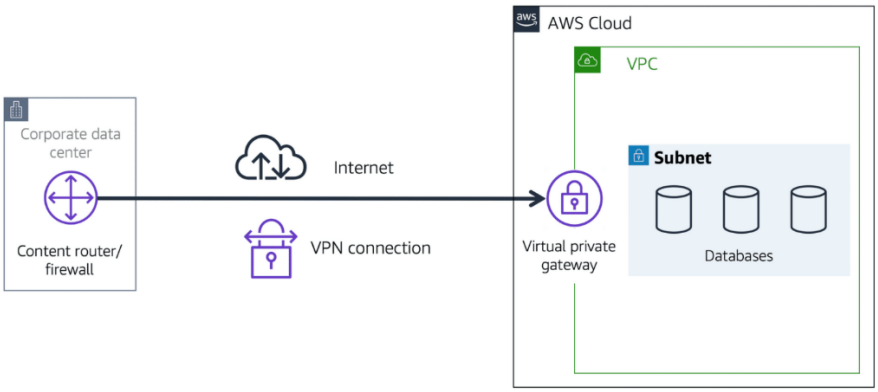
To access private resources in a VPC, you can use a **virtual private gateway**, which is like a private doorway. It allows you to create a **VPN connection between a private network, like your on-premises data centre and internal corporate network to your VPC**.

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. Although VPN connections are private and they are encrypted, but they still use a regular internet

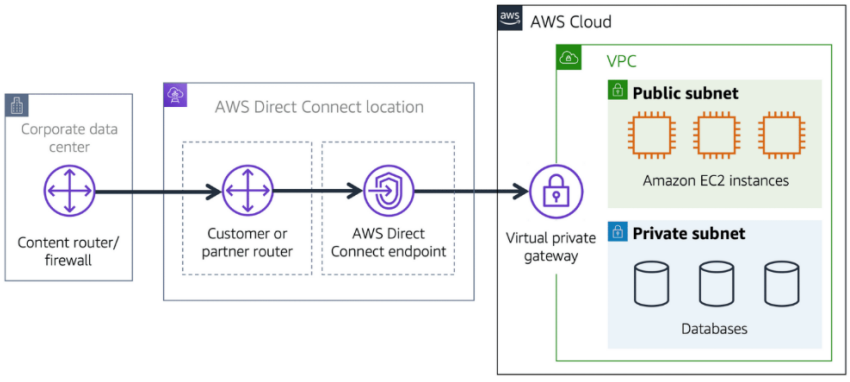
connection that has bandwidth that is being shared by many people using the internet.



AWS Direct Connect

What if you still want a private connection, but you want it to be dedicated and shared with no one else? You want the **lowest amount of latency possible with the highest amount of security possible**. With AWS, you can achieve that using what is called AWS Direct Connect. **AWS Direct Connect** is a service that enables you to establish a **dedicated private connection between your data centre 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.



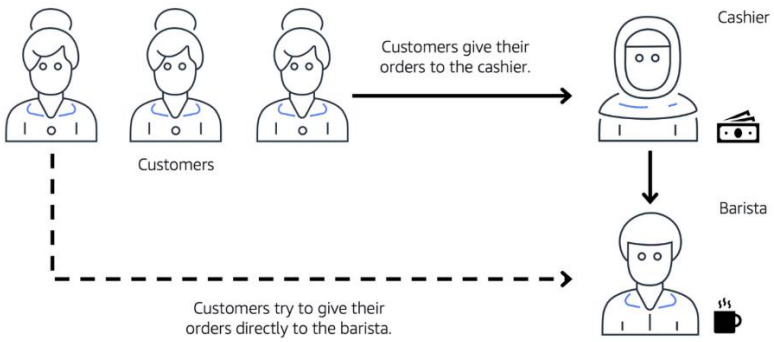
The private connection that AWS Direct Connect provides helps you to reduce network costs and increase the amount of bandwidth that can travel through your network.

This can help you meet high regulatory and compliance needs, as well as **sidestep any potential bandwidth issues**. It's also important to note that one VPC might have **multiple types of gateways attached** for multiple types of resources all residing in the same VPC, just in **different subnets**.

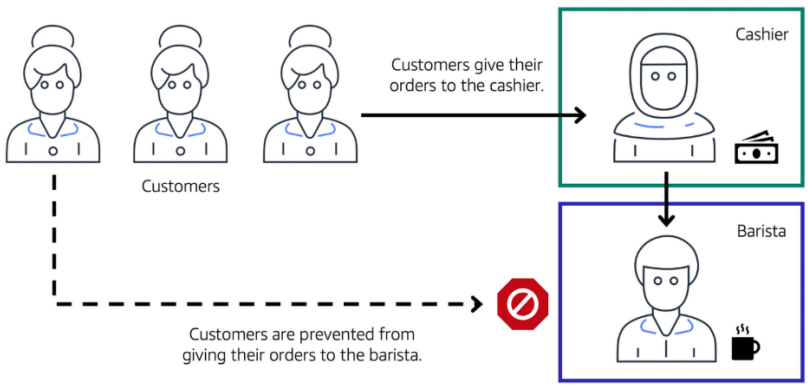
Subnets

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.



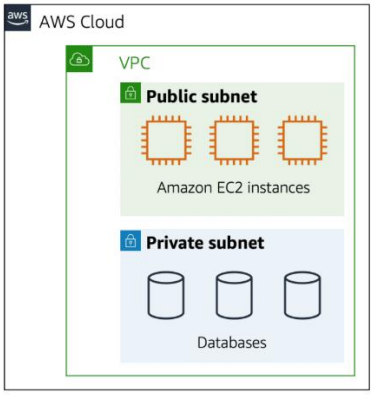
To fix this, the owners of the coffee shop divide the counter area by placing the cashier and the barista in separate workstations. The cashier's workstation is public facing and designed to receive customers. The barista's area is private. The barista can still receive orders from the cashier but not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

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.



Now, the only technical reason to use subnets in a VPC is to control access to the gateways. The public subnets have access to the internet gateway; the private subnets do not.

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

Subnets can also control traffic permissions. 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.

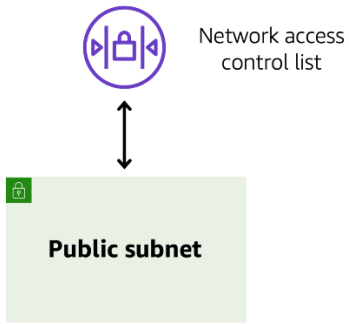
Packets are messages from the internet, and every packet that **crosses the subnet boundaries gets checked against** something called a **network access control list (ACL)** or network ACL. This check is to see if the packet has permissions to either **leave or enter the subnet** based on who it was sent from and how it's trying to communicate.

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, travellers are trying to enter into a different country. You can think of the travellers

as packets and the passport control officer as a network ACL. The passport control officer checks travellers' credentials when they are both entering and exiting out of the country. If a traveller 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 travellers, they cannot come in.



With network ACL, approved traffic can be sent on its way, and potentially harmful traffic, like attempts to gain control of a system through administrative requests, they get blocked before they ever touch the target. You can't hack what you can't touch.

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.

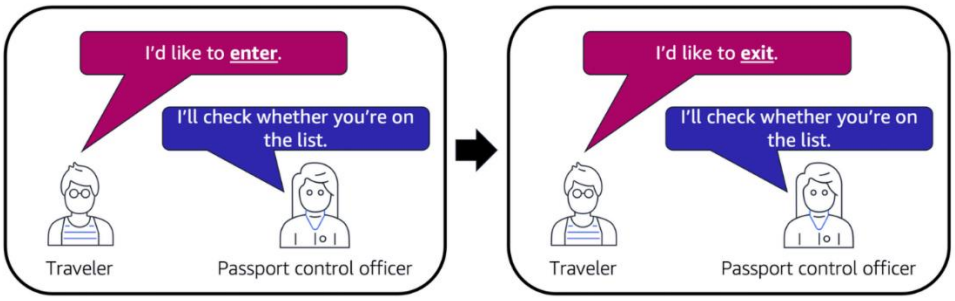
By default, your account's default network ACL allows all inbound and outbound traffic, but you can modify it by adding your own rules. For custom network ACLs, all inbound and outbound traffic is denied until you add rules to specify which traffic to allow. Additionally, all network ACLs have an explicit 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 traveller 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.



Now, this sounds like great security, but it doesn't answer all of the network control issues. Because a network ACL only gets to evaluate a packet if it crosses a subnet boundary, in or out. It **doesn't evaluate if a packet can reach a specific EC2 instance or not**. Sometimes, you'll have multiple EC2 instances in the same subnet, but they might have **different rules** around who can send those messages, what port those messages are allowed to be sent to. So you **need instance level network security** as well i.e. 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

To solve instance level access questions, we introduce security groups. A security group is a virtual firewall that controls inbound and outbound traffic for **an Amazon EC2 instance.**

Every EC2 instance, when it's launched, automatically comes with a security group. And **by default, the security group does not allow any traffic into the instance at all** (i.e. a security group denies all inbound traffic and allows all outbound traffic). **All ports are blocked**; all IP addresses sending packets are blocked. That's very secure, but perhaps not very useful.

You can add custom rules to configure which traffic to allow or deny. For example, in the case of a website, you want web-based traffic or HTTPS to be accepted but not the other types of traffic, say an operating system or administration requests.

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.** With security groups, you allow specific traffic in and by default, all traffic is allowed out.

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.

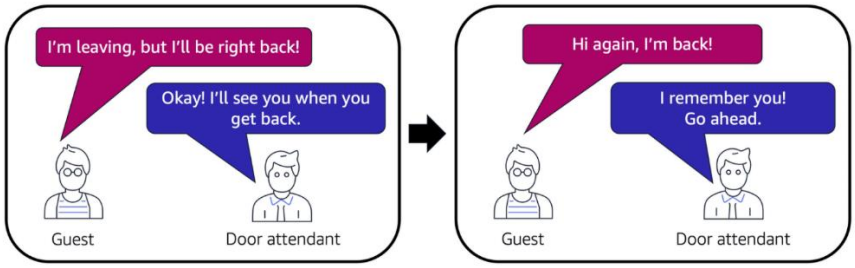
Now, wait a minute. We just described two different engines each doing the exact same job. Let good packets in, keep bad packets out. **The key difference** between a security group and a network ACL is the **security group is stateful**, meaning it has some kind of a memory when it comes to who to allow in or out, and the **network ACL is stateless**, which remembers nothing and **checks every single packet that crosses** its border regardless of any circumstances.

Stateful packet filtering

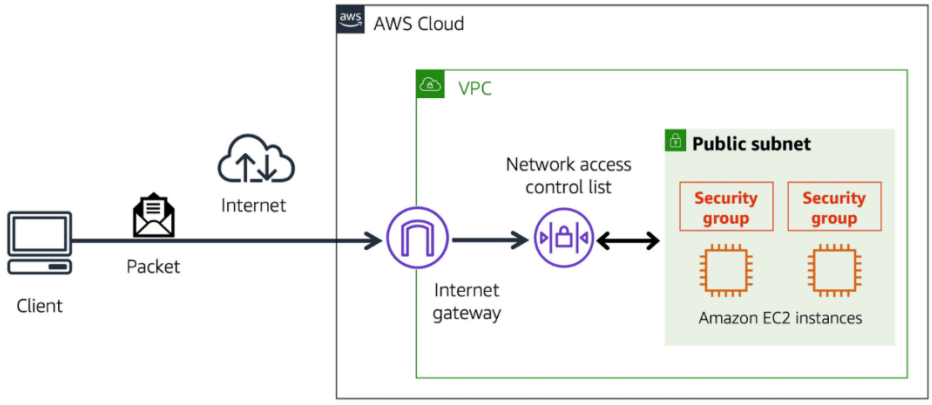
Security groups perform **stateful** packet filtering. They remember previous decisions made for incoming packets.

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

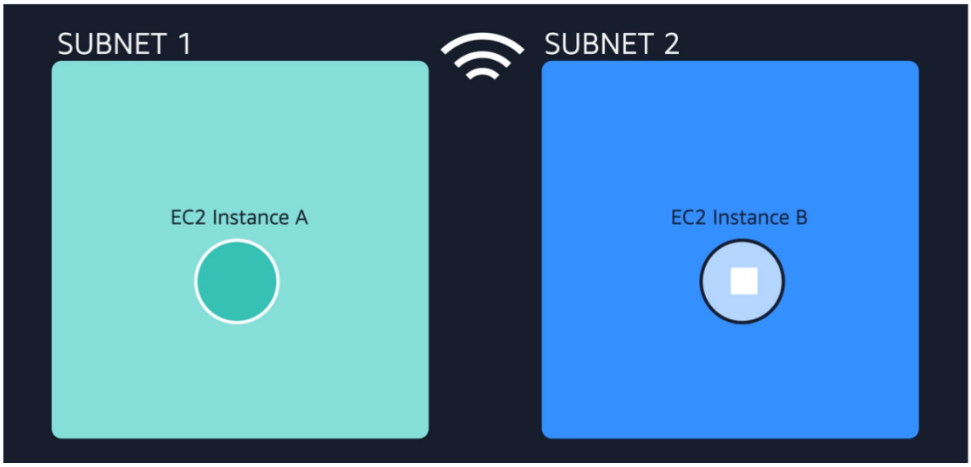
When a packet response for that request returns to the instance, the security group remembers your previous request. The security group allows the response to proceed, regardless of inbound security group rules.



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**.



Example



All right. Let's start with instance A. We want to send a packet from instance A to instance B in a different subnet, same VPC. So instance A sends the packet. Now, the first thing that happens is that packet meets the boundary of the security group of EC2 instance A. **By default, all outbound traffic is allowed from a security group.** So you can walk right by the doormen and leave, cool, right. The packet made it past the security group of instance A.

Now it has to leave the subnet 1 boundary. At the boundary of subnet 1, the passport must now make it through passport control, the network ACL. The network ACL doesn't care about what the security group allowed. It has its own list of who can pass and who can't. If the target address is allowed, you can keep going on your journey, which it is.

So now we have exited the original subnet and now the packet goes to the target subnet 2 where instance B lives. Now at this target subnet, once again, we have passport control. Just because the packet was allowed out of its home country does not mean that it can enter the destination country or subnet in this case. They both have unique passport officers with their own checklists. You have to be approved on both lists, or you could get turned away at the border. Well, turns out the packet is on the approved list for this subnet, so it's allowed to enter through the network ACL into the subnet. Almost there. Now, we're approaching the target instance, instance B. Every EC2 Instance has their own security group. You want

to enter this instance, the doorman will need to check to see if you're allowed in, and we're on the list. The packet has reached the target instance.

Once the transaction is complete, now it's just time to come home. It's the **return traffic pattern**. It's the most interesting, because this is where the **stateful** versus **stateless** nature of the different engines come into play. Because the packet still has to be evaluated at each checkpoint.

Security groups, by default, allow all return traffic. So they don't have to check a list to see if they're allowed out. Instead, **security groups automatically allow the return traffic to pass** by no matter what. Passport control here at the subnet boundary, these network ACLs do not remember state. They don't care that the packet came in here. It might be on a you-can't-leave list. Every ingress and egress is checked with the appropriate list at the subnet boundary (network ACL). The package return address has to be on their approved list to make it across the border. Made it to the border of the origin subnet 1, but we have to negotiate passport network ACL control here as well. Stateless controls, means it always checks its list.

The packet pass the network ACL, the subnet level, which means the packets now made it back to instance A but the security group, the doorman is still in charge here. The key difference though is these security groups are **stateful**. The security group recognizes the packet from before. So it **doesn't need to check to see if it's allowed in**. Come on home.

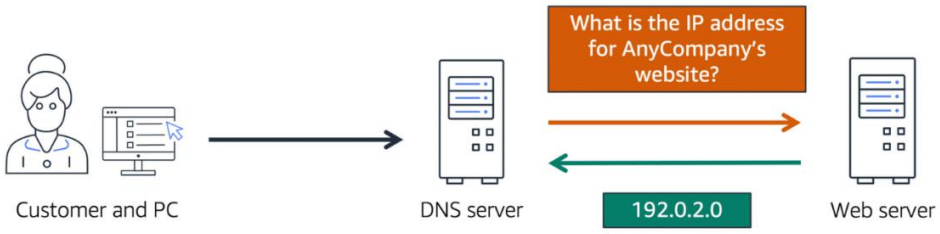
Now, this may seem like we spent a lot of effort just getting a packet from one instance to another and back. You might be concerned about all the network overhead this might generate. The reality is all of these exchanges happen instantly as part of how AWS Networking actually works. Good network security will take advantage of both network ACLs and security groups, because security in-depth is critical for modern architectures.

Domain Name System (DNS)

We've been talking a lot about how you interact with your AWS infrastructure. But how do your customers interact with your AWS infrastructure?

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.

Think of DNS as a translation service. But instead of translating between languages, it translates website names into IP, or Internet Protocol, addresses that computers can read.

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.

If we go further, Route 53 can **direct traffic to different endpoints** using several different routing policies, such as **latency-based routing, geolocation DNS, geoproximity, and weighted round robin**. If we take geolocation DNS, that means we direct traffic based on where the customer is located. So traffic coming from say North America is routed to the Oregon Region, and traffic in Ireland is routed to the Dublin Region, as an example.

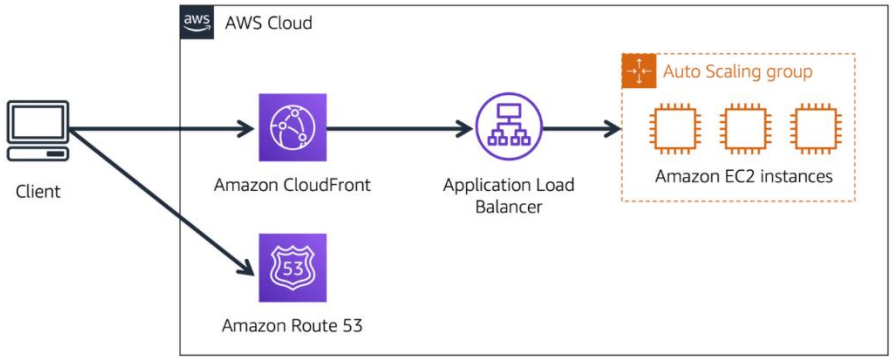
Another feature of Route 53 is the ability to **manage the DNS records for domain names**. You can **register new domain names directly in Route** 53, so you can buy and manage your own domain names right on AWS. You can also transfer DNS records for existing domain names managed by other domain registrars.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

Speaking of websites, there is another service which can help speed up delivery of website assets to customers, **Amazon CloudFront**. In the previous module, you learned about Amazon CloudFront, a content delivery service. If you remember, we talked about Edge locations earlier in the course, these locations are serving content as close to customers as

possible, and one part of that, is the content delivery network, or CDN. For those who need reminding, a CDN is a network that helps to deliver edge content to users based on their geographic location.

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.

If we go back to our North America versus Ireland example, say we have a user in Seattle, and they want to access a website, to speed this up, we host the site in Oregon, and we deploy our static web assets, like images and GIFs in CloudFront in North America. This means they get content delivered as close to them as possible, North America in this case, when they access the site. But for our Irish users, it doesn't make sense to deliver those assets out of Oregon, as the latency is not favourable. Thus we deploy those same static assets in CloudFront, but this time in the Dublin Region. That means they can access the same content, but from a location closer to them, which in turn **improves latency**.

Summary

Networking used to be the exclusive domain of topological geniuses. With AWS, networking is now simplified and abstracted to answer the simple question of who should be allowed to communicate with each other. As long as you can answer that question, then you can set up your network on AWS.

We covered the basics of **VPC**, the virtual private cloud, the way that you isolate your workload in AWS, the fundamentals of network security, including **gateways**, **network ACLs**, and **security groups**, all methods that allow your security engineers to craft a network that allows healthy traffic access while dropping subversive attacks before they reach your instance, ways to **connect to AWS through VPN** and even **Direct Connect**, secure pipelines that are either encrypted over the general internet or exclusive fibre used by you and you alone.

We also talked about the global networks that AWS provides using our Edge locations, how you can use **Route 53** for DNS, and how to use **CloudFront** to cache content closer to your actual consumers.

Quiz

Your company has an application that uses Amazon EC2 instances to run the customer-facing website and Amazon RDS database instances to store customers' personal information. How should the developer configure the VPC according to best practices?

- Place the Amazon EC2 instances in a public subnet and the Amazon RDS database instances in a private subnet.

Which component can be used to establish a private dedicated connection between your company's data centre and AWS?

- AWS Direct Connect

Which statement best describes security groups?

- They are stateful and deny all inbound traffic by default.

Which component is used to connect a VPC to the internet?

- Internet gateway

Which service is used to manage the DNS records for domain names?

- 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 that host in AWS.

Additional resources

To learn more about the concepts that were explored in Module 4, review these resources.

- Networking and Content Delivery on AWS
- AWS Networking & Content Delivery Blog
- Amazon Virtual Private Cloud
- What is Amazon VPC?
- How Amazon VPC works

Module 5 – Storage and Databases

Contents

Learning objectives 2
Instance stores 2
Amazon Elastic Block Store 3
Amazon EBS snapshots 4
Object storage 4
Amazon Simple Storage Service (Amazon S3) 5
Amazon S3 storage classes 5
Comparing Amazon EBS and Amazon S3 8
File storage 8
Comparing Amazon EBS and Amazon EFS 9
Relational databases 9
Amazon Relational Database Service 10
Amazon RDS database engines 10
Amazon Aurora 10
Nonrelational databases 11
Amazon DynamoDB 12
Comparing Amazon RDS and Amazon DynamoDB 12
Amazon Redshift 13
AWS Database Migration Service (AWS DMS) 14
Additional database services 15
Summary 16
Quiz 16
Additional resources 17

Learning objectives

In this module, you will learn how to:

- Summarize the basic concept of storage and databases.
- Describe the benefits of Amazon Elastic Block Store (Amazon EBS).
- Describe the benefits of Amazon Simple Storage Service (Amazon S3).
- Describe the benefits of Amazon Elastic File System (Amazon EFS).
- Summarize various storage solutions.
- Describe the benefits of Amazon Relational Database Service (Amazon RDS).
- Describe the benefits of Amazon DynamoDB.
- Summarize various database services.

With what we have learnt earlier, we can now have an **elastic, scalable, disaster resistant, cost optimized architecture that now has a global, highly secured network that can be deployed entirely programmatically.** Now let's talk about storage and databases.

Instance stores

When you're using Amazon EC2 to run your business applications, those applications need access to CPU, memory, network, and storage. EC2 instances give you access to all those different components, and right now, let's focus on the **storage access**. As applications run, they will oftentimes need access to **block-level storage**.

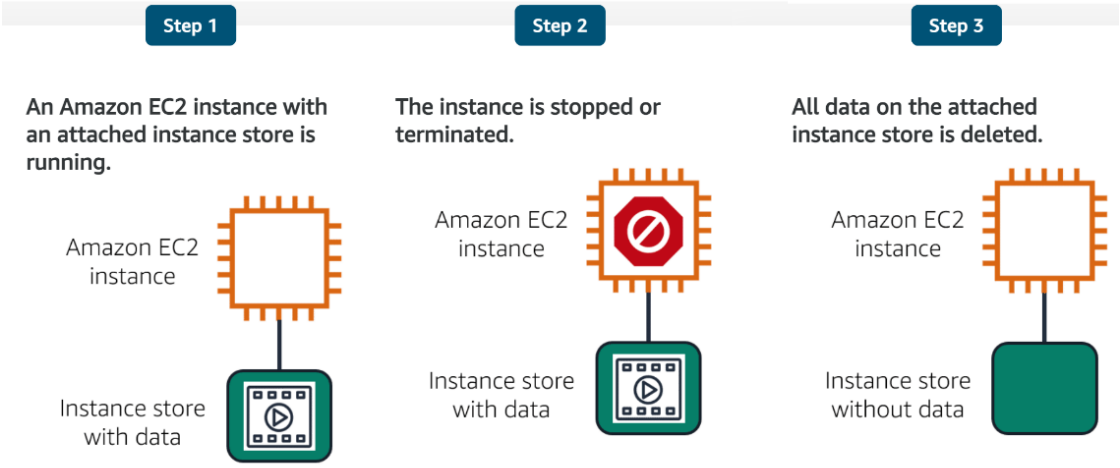
You can think of **block-level storage as a place to store files**. A file being a series of bytes that are stored in blocks on disc. When a file is updated, the whole series of blocks aren't all overwritten. Instead, it updates just the pieces that change. This makes it an efficient storage type when working with applications like databases, enterprise software, or file systems.

When you use your laptop or personal computer, you are accessing block-level storage. All block-level storage is in this case is your hard drive. EC2 **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. The catch here is that since this volume is attached to the underlying physical host, if you **stop or terminate** your EC2 instance, all data written to the instance store volume will be deleted. This means that when the instance is terminated, you lose any data in the instance store.

The reason for this, is that if you **start your instance from a stop state**, it's likely that EC2 instance will **start up on another host**. A host where that volume does not exist. Remember **EC2 instances are virtual machines**, and therefore the **underlying host can change** between

stopping and starting an instance.



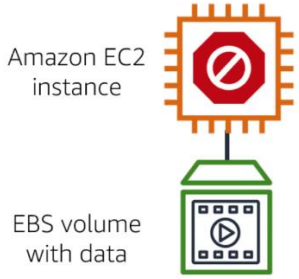
Amazon EC2 instances are virtual servers. If you start an instance from a stopped state, the instance might **start on another host**, where the previously used instance store volume does not exist. Therefore, AWS recommends instance stores for use cases that involve temporary data that you do not need in the long term.

Because of this **ephemeral or temporary nature of instance store volumes**, they are useful in situations where you can lose the data being written to the drive. Such as temporary files, scratch data, and data that can be easily recreated without consequence.

Amazon Elastic Block Store

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**.

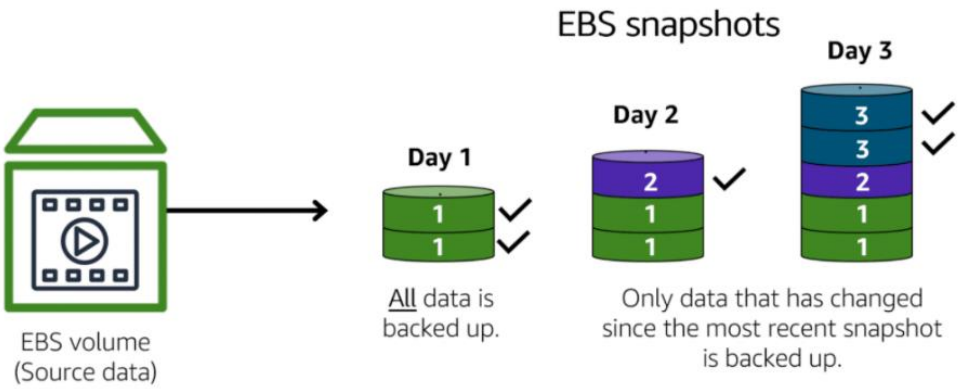
With EBS, you can create virtual hard drives that we call EBS volumes that you can attach to your EC2 instances. These are separate drives from the local instance store volumes, and they aren't tied directly to the host that your EC2 instance is running on. This means that the data that you write to an EBS volume **persists between stops and starts of an EC2 instance** (i.e. beyond the lifecycle of an EC2 instance).



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**. It's very important that you take regular snapshots of your EBS volumes. This way, if a drive ever becomes corrupted, you haven't lost your data. And you can restore that data from a snapshot.

Amazon EBS snapshots



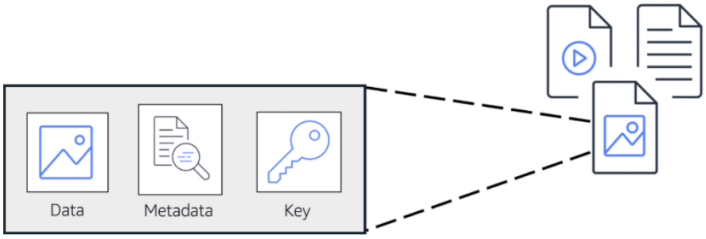
An **EBS snapshot** is an **incremental** backup. This means that the first backup taken of a volume copies all the data. For subsequent backups, **only the blocks of data that have changed** since the most recent snapshot are saved.

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.**

Object storage

Most businesses have data that needs to be stored somewhere. For the coffee shop, this could be receipts, images, Excel spreadsheets, employee training videos, and even text files, among others. Storing these files is where S3 comes in handy because it is a data store that allows you to **store and retrieve an unlimited amount of data at any scale**.

Data is stored as objects, but instead of storing them in a file directory, you store them in what we call **buckets**. Think of a file sitting on your hard drive, which is an object and think of a **file directory, which is the bucket**.



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**.

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. What this means is that you always retain the previous versions of an object, as like a paper trail.

You can even create multiple buckets and store them across different classes or tiers of data. You can then create permissions to limit who can see or even access objects.

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

S3 Standard

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

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. S3 Standard has a higher cost than other storage classes intended for infrequently accessed data and archival storage.

It comes with **11 nines of durability**. That means an object stored in S3 Standard has a 99.999999999 percentage probability that it will remain intact after a period of one year. That's pretty high.

Furthermore, data is stored in such a way that AWS can sustain the concurrent loss of data in two separate storage facilities. This is because data is stored in **at least three facilities**, so multiple copies reside across locations.

Another useful way to use S3 is **static website hosting**, where a static website is a collection of HTML files and each file is akin to a physical page of the actual site. You can do this by simply uploading all your HTML, static web assets, and so forth into a bucket and then checking a box to host it as a static website. You can then enter the bucket's URL and ba-bam! Instant website. And we say static, but that doesn't mean you can't have animations and moving parts to your website.

S3 Standard-Infrequent Access (S3 Standard-IA)

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

S3 Standard-IA is ideal for data infrequently accessed but requires high availability (i.e. rapid access) when needed. It's a perfect place to store **backups**, **disaster recovery files**, or any object that requires a long-term storage.

Both S3 Standard and S3 Standard-IA store data in a **minimum of three Availability Zones**. S3 Standard-IA provides the same level of availability as S3 Standard but with a **lower storage price and a higher retrieval price.**

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

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

Compared to S3 Standard and S3 Standard-IA, which store data in a minimum of three Availability Zones, 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.

S3 Intelligent-Tiering

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

In the 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, **S3 Standard-IA**. If you access an object in the infrequent access tier, Amazon S3 automatically moves it to the frequent access tier, **S3 Standard**.

S3 Glacier

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

Say, we need to **retain data for several years for auditing** purposes. And we don't need it to be retrieved very rapidly. Well, then you can use Amazon S3 Glacier to archive that data.

S3 Glacier 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.

To use Glacier, you can simply move data to it, or you can create vaults and then populate them with archives. And if you have **compliance requirements** around retaining data for, say, a certain period of time, you can employ an S3 Glacier **vault lock policy** and lock your vault. You can specify controls such as **write once/ read many**, or WORM, in a vault lock policy and **lock the policy from future edits**. Once locked, the policy can no longer be changed. You also have three options for retrieval, which range from minutes to hours, and you have the option of uploading directly to Glacier or using S3 Lifecycle policies.

Lifecycle policies are policies you can create that can **move data automatically between tiers.** For example, say we need to keep an object in S3 Standard for 90 days, and then we want to move it to S3-IA for the next 30 days. Then after 120 days total, we want it to be moved to S3 Glacier. With Lifecycle policies, you create that configuration without changing your application code and it will perform those moves for you automatically.

S3 Glacier Deep Archive

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

When deciding between Amazon S3 Glacier and Amazon S3 Glacier Deep Archive, consider how quickly you need to retrieve archived objects. You can retrieve objects stored in the S3 Glacier storage class within a few minutes to a few hours. By comparison, you can retrieve objects stored in the S3 Glacier Deep Archive storage class **within 12 hours**.

Comparing Amazon EBS and Amazon S3

Let's say you're running a photo analysis website where users upload a photo of themselves, and your application finds the animals that look just like them. You have potentially millions of animal pictures that all need to be indexed and possibly viewed by thousands of people at once. This is the perfect use case for S3. **S3 is already web enabled**. **Every object already has a URL** that you can control access rights to who can see or manage the image.

S3 is **regionally distributed**, which means that it has 11 nines of durability, so no need to worry about backup strategies. S3 is your **backup strategy**. Plus the cost savings is substantial overrunning the same storage load on EBS. **With the additional advantage of being serverless, no Amazon EC2 instances are needed**. Sounds like S3 is the judge's winner here for this round.

But wait, round two, you have an 80-gigabyte video file that you're making **edit corrections** on. To know the best storage class here, we need to understand the difference between object storage and block storage. **Object storage treats any file as a complete, discreet object.** Now this is great for documents, and images, and video files that get uploaded and consumed as entire objects, but every time there's a change to the object, you must **re-upload the entire file**. There are **no delta updates**.

Block storage breaks those files down to small component parts or blocks. This means, for that 80-gigabyte file, when you make an edit to one scene in the film and save that change, the engine only updates the blocks where those bits live. If you're making a bunch of micro edits, using **EBS, elastic block storage**, is the perfect use case. If you were using S3, every time you saved the changes, the system would have to upload all 80 gigabytes, the whole thing, every time. EBS clearly wins round two.

This means, if you are using **complete objects or only occasional changes, S3 is victorious**. If you are doing **complex read, write, change functions, then, absolutely, EBS is your knockout winner**. Your winner depends on your individual workload. Each service is the right service for specific needs. Once you understand what you need, you will know which service is your champion!

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.

It's extremely common for businesses to have shared file systems across their applications. For example, you might have multiple servers running analytics on large amounts of data being stored in a shared file system.

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.

EFS allows you to have multiple instances accessing the data in EFS at the same time. It scales up and down as needed without you needing to do anything to make that scaling happen.

Comparing Amazon EBS and Amazon EFS

Amazon EBS volumes attach to EC2 instances and are an Availability Zone-level resource i.e. 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**. You can save files on it. You can also run a database on top of it. Or store applications on it. It's a hard drive. If you provision a two terabyte EBS volume and fill it up, it doesn't automatically scale to give you more storage. So that's EBS.

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, meaning that Amazon EFS can have multiple instances reading and writing from it at the same time. Additionally, **on-premises servers can access Amazon EFS using AWS Direct Connect.**

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 fulfil 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

The benefits of Amazon RDS include **automated patching, backups, redundancy, failover, disaster recovery**, all of which you normally have to manage for yourself. This makes it an extremely attractive option to AWS customers, as it allows you to focus on business problems and not maintaining databases.

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 is AWS' **most managed relational database option**. It comes in two forms, **MySQL** and **PostgreSQL**. And is priced is 1/10th the cost of commercial grade databases. That's a pretty cost effective database.

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.** You therefore also get point in time recovery, so you can recover data from a specific period.

You can also deploy up to 15 read replicas, so you can offload your reads and scale performance.

Nonrelational databases

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

Relational databases works great for a lot of use cases, and has been the standard type of database historically. However, these types of rigid SQL databases, can **have performance and scaling issues when under stress**. The rigid schema also makes it so that you cannot have any variation in the types of data that you store in a table. So, it might not be the best fit for a dataset that is a little bit less rigid, and is being accessed at a very high rate. This is where non-relational, or NoSQL, databases come in.

Nonrelational databases are sometimes referred to as "NoSQL databases" because they use structures other than rows and columns to organize data. Non-relational databases tend to have simple flexible schemas, not complex rigid schemas, laying out multiple tables that all relate to each other.

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:

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

Amazon DynamoDB

Amazon DynamoDB is a key-value database service. It delivers single-digit millisecond performance at any scale. It is a **non-relational, NoSQL database**.

With DynamoDB, you create tables. A DynamoDB table, is just a place where you can store and query data. Data is organized into items, and items have attributes. Attributes are just different features of your data. If you have one item in your table, or 2 million items in your table, DynamoDB manages the underlying storage for you.

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**.

DynamoDB, **beyond being massively scalable, is also highly performant**. DynamoDB has a millisecond response time. And when you have applications with potentially millions of users, having scalability and reliable lightning fast response times is important. 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.

DynamoDB stores this data redundantly **across availability zones** and mirrors the data across multiple drives under the hood for you. This makes the burden of operating a highly available database, much lower.

Comparing Amazon RDS and Amazon DynamoDB

Relational databases have been around since the moment businesses started using computers. Being able to build complex analysis of data spread across multiple tables, is the strength of any relational system. In this round, you have a **sales supply chain management system** that you have to analyse for weak spots. Using RDS is the clear winner here because it's built for business analytics, because you **need complex relational joins**. Round one easily goes to RDS.

Round two, the use case, pretty much anything else. Now that sounds weird, but despite what your standalone legacy database vendor would have you believe**, most of what people use expensive relational databases for, has nothing to do with complex relationships**. In fact, **a lot of what people put into these databases ends up just being look-up tables.**

For this round, imagine you have an employee contact list: names, phone numbers, emails, employee IDs. Well, this is all single table territory. I could use a relational database for this, but the things that make relational databases great, **all of that complex functionality, creates overhead and lag and expense if you're not actually using it**.

This is where non-relational databases, Dynamo DB, delivers the knockout punch. By eliminating all the overhead, **DynamoDB allows you to build powerful, incredibly fast databases where you don't need complex joint functionality**. DynamoDB comes out the undisputed champion.

Once again, the winner depends on your individual workload. Each service is the right service for specific needs. And once you understand what you need, you will know again, which service is your champion.

Amazon Redshift

We just spent a lot of time discussing the kinds of workflow that require fast, reliable, current data. Databases that can handle 1,000s of transactions per second, storage that is highly available and massively durable. But sometimes, we have a business need that goes outside what is happening right now to what did happen. This data analysis is the realm of a whole different class of databases. Sure, you could use the one size fits all model of a single database for everything, but modern databases that are engineered for high speed, real time ingestion, and queries may not be the best fit.

Let me explain. In order to handle the **velocity** of real time read/write functionality, most relational databases tend to function fabulously at certain capacities. How much content it actually stores. The problem with historical analytics, data that answers questions like, "Show me how production has improved since we started", is the data collection never stops. In fact, with modern telemetry and the explosion of IoT, the **volume** of data will overwhelm even the beefiest traditional relational database.

It gets worse. Not just the volume, but the **variety** of data can be a problem. You want to run business intelligence or BI projects against data coming from different data stores like your inventory, your financial, and your retail sales systems? **A single query against multiple databases sounds nice, but traditional databases don't handle them easily.**

Once data becomes too complex to handle with traditional relational databases, you've entered the world of data warehouses. Data warehouses are engineered specifically for this kind of **big data**, where you are looking at historical analytics as opposed to operational analysis.

Now, let's be clear. Historical may be as soon as: show me **last hour's** sales numbers across all the stores. The key thing is, the data is now set. We're not selling any more from the last hour because that is now in the past. Compare that question to, "How many bags of coffee do we still have in our inventory right now?" Which could be changing as we speak. **As long as your business question is looking backwards at all, then a data warehouse is the right solution for that line of business intelligence.**

A data warehouse is a system that pulls together data from many different sources within an organization for reporting and analysis. The reports created from complex queries

within a data warehouse are used to make business decisions. A data warehouse stores **historical data** about your business so that you can analyse and extract insights from it. It does **not** store current information, nor is it updated in real-time.

Now there are many data warehouse solutions out on the market. If you already have a favourite one, running it on AWS is just a matter of getting the data transferred. But beyond that, there may still be a lot of undifferentiated heavy lifting that goes into **keeping a data warehouse tuned, resilient, and continuously scaling**. Wouldn't it be nice if your data warehouse team could focus on the data instead of the unavoidable care and feeding of the engine?

Introducing Amazon Redshift. **Amazon Redshift** is a **data warehousing service** that you can use **for big data analytics**. It offers the ability to collect data from many sources and helps you to understand relationships and trends across your data. This is **data warehousing as a service**. It's massively scalable. Redshift nodes in multiple petabyte sizes is very common. In fact, in cooperation with Amazon Redshift Spectrum, you can **directly run a single SQL query against exabytes of unstructured data running in data lakes.**

But it's more than just being able to handle massively larger data sets. Redshift uses a variety of innovations that allow you to achieve up to 10 times higher performance than traditional databases, when it comes to these kinds of business intelligence workloads.

We have whole classes that you or your data teams can take that explain how it is built, and why it can return such improved results. The key for you is to understand that when you need **big data BI solutions**, Redshift allows you to get started with a single API call. Less time waiting for results, more time getting answers.

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.

DMS helps customers migrate existing databases onto AWS in a secure and easy fashion. 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 (**homogenous** migration) or different types (**heterogeneous** migration). During the migration, your **source database remains operational**, reducing downtime for any applications that rely on the database.

For heterogeneous migrations, it's a **two-step** process. Since the schema structures, data types, and database code are different between source and target, we first need to convert them using the **AWS Schema Conversion Tool**. This will convert the source schema and code to match that of the target database. The next step is then to use DMS to migrate data from the source database to the target 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. In this case, you use DMS to migrate a copy of your production database to your **dev or test environments**, either once-off or continuously.

Database consolidation: Combining several databases into a **single central** database

Continuous replication: Sending ongoing copies of your data to other target sources instead of doing a one-time migration. This could be for disaster recovery or because of geographic separation.

Additional database services

Amazon DocumentDB is a document database service that supports **MongoDB** workloads. (MongoDB is a document database program). This is great for content management, catalogs, and user profiles.

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 social networking and recommendation engines, fraud detection, and knowledge graphs.

Amazon Quantum Ledger Database (Amazon QLDB) is an **immutable 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 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 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 (DAX) is an **in-memory cache for DynamoDB**. It helps to dramatically **improve response (read) times** for your nonrelational data from single-digit milliseconds to microseconds.

Summary

You learned about all the different types of AWS storage mechanisms. Let's recap them, shall we?

The first one we learned about is **Elastic Block Store (EBS)** volumes, and you attach those to EC2 instances so you have local storage that is not **ephemeral**.

You learned about how **S3** and how you can store objects in AWS with the click of a button or call of an API.

We even discussed the various relational database options available on AWS, such as Amazon **RDS**. Or for the workloads that just need a key-value pair, we have the non-relational offering called **DynamoDB**.

Next up was **EFS** for file storage use cases. We then have **Amazon Redshift** for all our data warehouse needs. And to aid in migration of existing databases, we have **DMS** or Database Migration Service.

We also touched upon the lesser known storage services, like **DocumentDB**, **Neptune**, **QLDB**, and Amazon **Managed Blockchain**. Lastly, we talked about how caching solutions like **ElastiCache** and DynamoDB Accelerator (**DAX**) can be used.

That's a lot of places to store different types of data, and hopefully you've learned the correct place to store each type.

Quiz

Which Amazon S3 storage classes are optimized for archival data? (Select TWO.)

- S3 Glacier
- S3 Glacier Deep Archive
- Note: Objects stored in the S3 Glacier storage class can be retrieved within a few minutes to a few hours. By comparison, objects that are stored in the S3 Glacier Deep Archive storage class can be retrieved within 12 hours.

Which statement or statements are TRUE about Amazon EBS volumes and Amazon EFS file systems?

- EBS volumes store data within a single Availability Zone. Amazon EFS file systems store data across multiple Availability Zones. An EBS volume must be located in the same Availability Zone as the Amazon EC2 instance to which it is attached. Data in an Amazon EFS file system can be accessed concurrently from all the Availability Zones in the Region where the file system is located.

You want to store data in an object storage service. Which AWS service is best for this type of storage?

- Amazon Simple Storage Service (Amazon S3)

Which statement best describes Amazon DynamoDB?

- A serverless key-value database service. Amazon DynamoDB is a key-value database service. It is serverless, which means that you do not have to provision, patch, or manage servers.

Which service is used to query and analyse data across a data warehouse?

- Amazon Redshift. It is a data warehousing service that you can use for big data analytics. Use Amazon Redshift to collect data from many sources and help you understand relationships and trends across your data.

Additional resources

To learn more about the concepts that were explored in Module 5, review these resources.

- Cloud Storage on AWS
- AWS Storage Blog
- Hands-On Tutorials: Storage
- AWS Customer Stories: Storage
- AWS Database Migration Service
- Databases on AWS
- Category Deep Dive: Databases
- AWS Database Blog
- AWS Customer Stories: Databases

Module 6 – Security

Contents

Learning objectives 2
The AWS shared responsibility model 2
AWS Identity and Access Management (IAM) 4
AWS account root user 4
IAM users 5
IAM policies 5
Example: IAM policy 6
IAM groups 6
IAM roles 7
Multi-factor authentication 8
AWS Organizations 9
Organizational units 9
AWS Artifact 11
Customer Compliance Centre 12
Denial-of-service attacks 13
Distributed denial-of-service attacks 14
Examples of DDoS Attacks 14
AWS Shield 15
AWS Key Management Service (AWS KMS) 16
AWS WAF 16
Amazon Inspector 17
Amazon GuardDuty 18
Summary 18
Quiz 19
Additional resources 20

Learning objectives

In this module, you will learn how to:

- Explain the benefits of the shared responsibility model.
- Describe multi-factor authentication (MFA).
- Differentiate between AWS Identity and Access Management (IAM) security levels.
- Explain the main benefits of AWS Organizations.
- Describe security policies at a basic level.
- Summarize the benefits of compliance with AWS.
- Explain additional AWS security services at a basic level.

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**.

This is your operating system. You're 100% in charge of this. AWS does not have any backdoor into your system here. You and you alone have the only encryption key to log onto the root of this OS or to create any user accounts there.

AWS: Security of the cloud

AWS is responsible for security *of* the cloud.

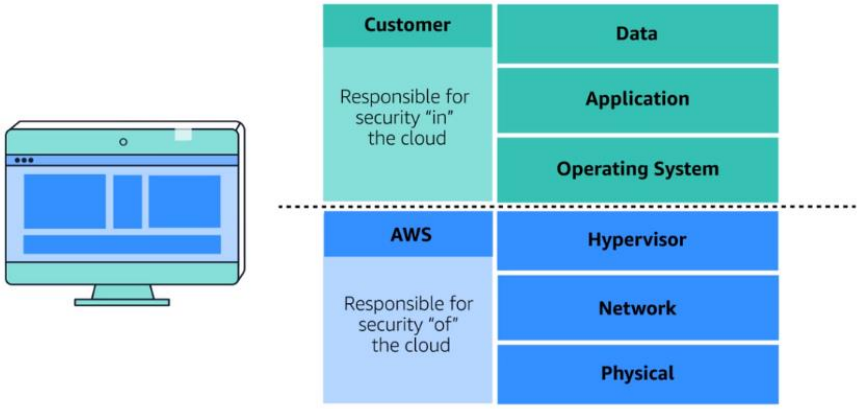
AWS operates, manages, and controls the components at all layers of infrastructure. This includes areas such as the host operating system, the virtualization layer, and even the physical security of the data centres from which services operate.

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 centres
- Hardware and software infrastructure
- Network infrastructure
- Virtualization infrastructure

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



AWS Identity and Access Management (IAM)

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

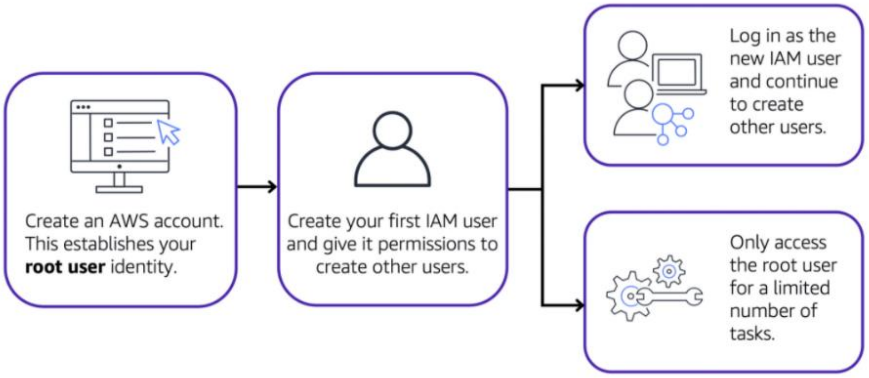
IAM gives you the flexibility to configure access based on your company's specific operational and security needs. You do this by using a combination of IAM features, which are explored in detail in this lesson:

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

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 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**. Because that user is so powerful, we recommend that as soon as you create an AWS account and log in with your root user, you turn on **multi-factor authentication**, or MFA, to ensure that you need not only the email and password, but also a randomized token to log in.

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. This is based on the **principle of least privilege**, where a user is granted access only to what they need. 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 **JSON** 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.

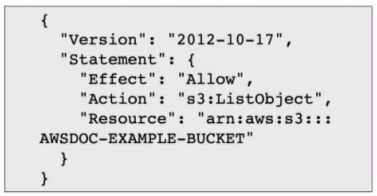
Best practice:

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"
  }
}
```

This example IAM policy allows permission to access the objects in the Amazon S3 bucket with ID: *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.

There were only two potential options for the **effect** on any policy. **Either allow or deny**. **For action**, you can list any AWS API call and **for resource**, you would list what AWS resource that specific API call is for.

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

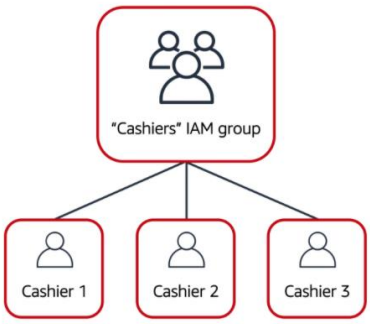
One way to make it easier to manage your users and their permissions is to organize them into IAM groups. An **IAM group is a collection of IAM users**. When you assign an IAM policy to a group, all users in the group are granted permissions specified by the policy.

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.

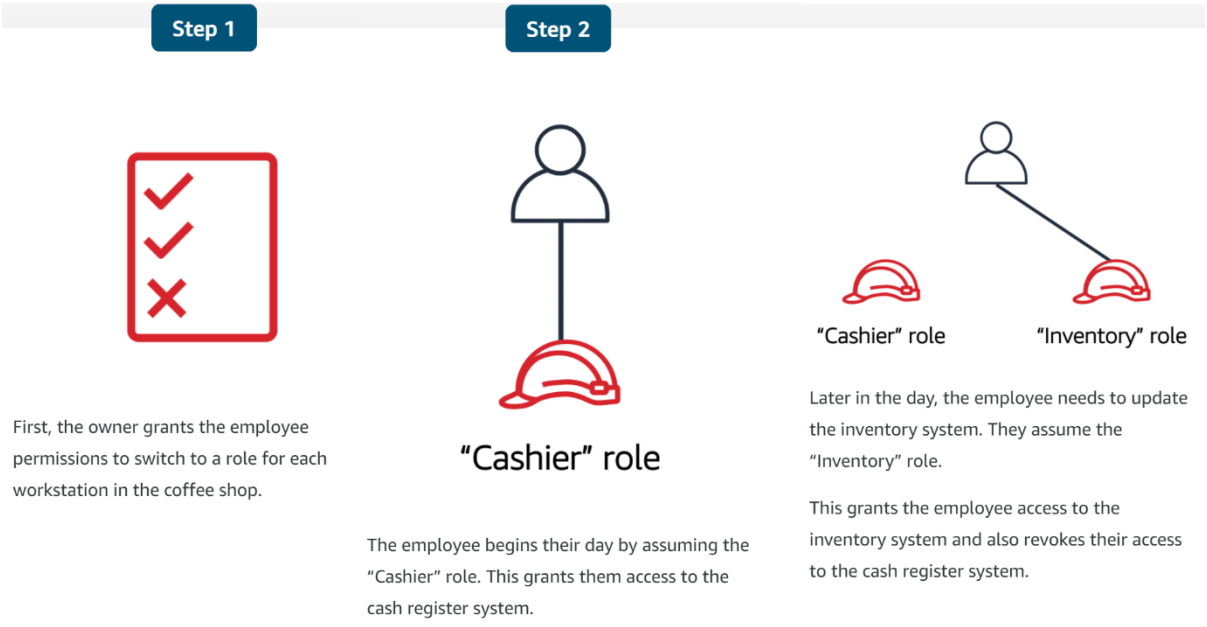
When the employee needs to switch to a different task, **they give up their access to one workstation and gain access to the next workstation**. The employee can easily switch between workstations, but at any given point in time, they can have access to only a single workstation. This same concept exists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

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

Roles have associated permissions that allow or deny specific actions. And these roles can be assumed for temporary amounts of time. It is similar to a user, but has **no username and password.** Instead, it is an **identity that you can assume to gain access to temporary permissions.** You use roles to temporarily grant access to AWS resources, to users, external identities, applications, and even other AWS services. When an identity assumes a role, it **abandons all of the previous permissions** that it has and it assumes the permissions of that role.

Best practice:

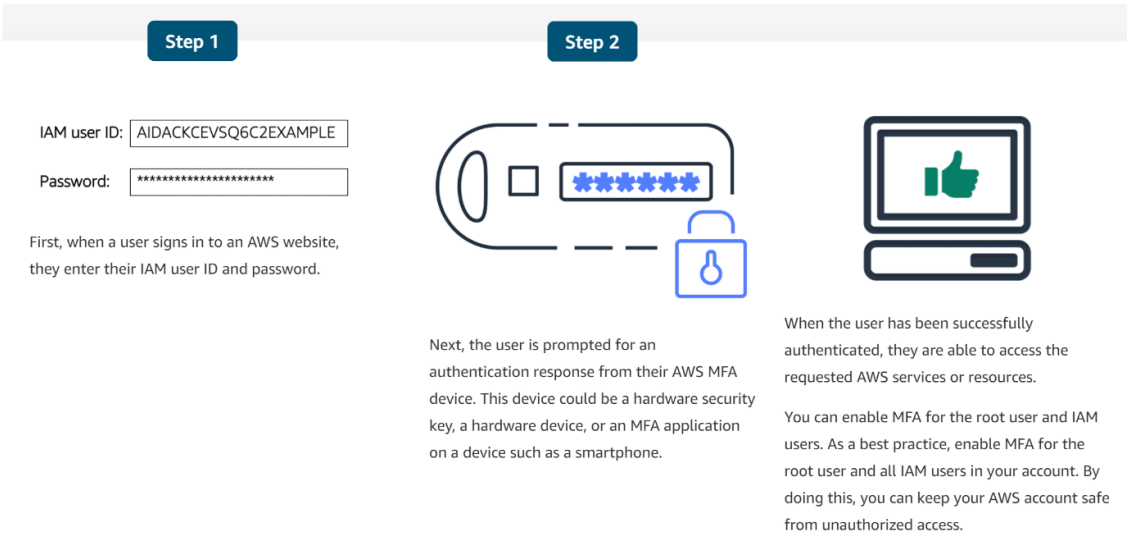
IAM roles are ideal for situations in which access to services or resources needs to be **granted temporarily**, instead of long-term.



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.

How multi-factor authentication works



AWS Organizations

With your first foray into the AWS Cloud, you most likely will start with one AWS account and have everything reside in there. Most people start this way, but as your company grows or even begins their cloud journey, it's important to have a separation of duties.

For example, you want your developers to have access to development resources, have your accounting staff able to access billing information, or even have business units separate so that they can experiment with AWS services without effecting each other.

Suppose that your company has multiple AWS accounts. You can use **AWS Organizations** to consolidate and **manage multiple AWS accounts within a central location**. The easiest way to think of Organizations is as a **central location to manage multiple AWS accounts**. You can manage billing control, access, compliance, security, and share resources across your AWS accounts.

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. In other words, you can apply service control policies (SCPs) to the **organization root, an individual member account, or an organizational unit (OU)**.

An **SCP affects all IAM users**, groups, and roles within an account, **including the AWS account root user**. You can apply IAM policies to IAM users, groups, or roles, but you **cannot apply an IAM policy to the AWS account root user.**

Consolidated billing is another feature of AWS Organizations. This means you can use the primary account of your organization to consolidate and pay for all member accounts. Another advantage of consolidated billing is **bulk discounts**. You will learn about consolidated billing in a later module.

Organizational units

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

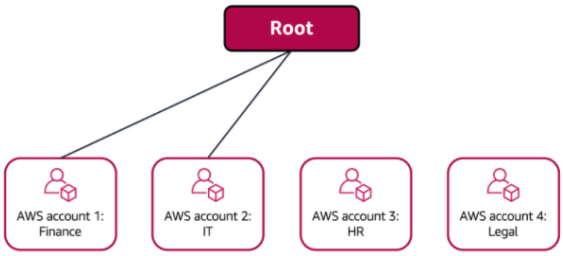
By organizing **separate accounts into OUs**, you can more easily isolate workloads or applications that have specific security requirements. For instance, if your company has accounts that can access only the AWS services that meet certain regulatory requirements, you can put these accounts into one 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.

Step 1

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.

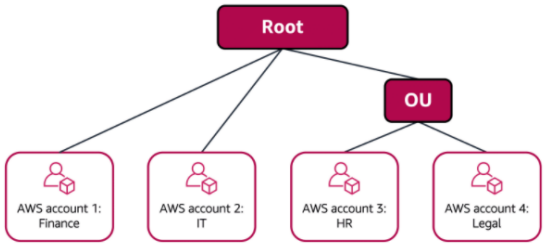
In designing your organization, you consider the business, security, and regulatory needs of each department. You use this information to decide which departments group together in OUs.

Step 2



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.

Step 3



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.

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. For example, you could be audited for taxes to see that you have run the back office correctly and have followed the law. You rely on documentation, records and inspections to pass audits and compliance checks as they come along. You'll need to devise a similar way to meet compliance and auditing in AWS.

AWS complies with a **long list of assurance programs** that you can find online. This means that segments of your compliance have already been completed, and you can focus on meeting compliance within your own architectures that you build on top of AWS.

The next thing to know in regards to compliance and AWS, is that the Region you choose to operate out of, might help you meet compliance regulations. If you can only legally store data in the country that the data is from, you can choose a Region that makes sense for you and AWS will not automatically replicate data across Regions.

AWS offers **multiple whitepapers and documents that you can download and use for compliance reports.** Since you aren't running the data centre yourself, you can essentially request that AWS provides you with documentation proving that they are following best practices for security and compliance. One place you can access these documents is through a service called AWS Artifact.

AWS Artifact is a service that provides **on-demand access to AWS security and compliance reports and select online agreements**. With AWS Artifact, you can gain access to compliance reports done by third parties who have validated a wide range of compliance standards. 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**). Another example is that if you run software that deals with consumer data in the EU, you

would need to make sure that you're in compliance with GDPR.

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.

Customer Compliance Centre

The **Customer Compliance Centre** contains resources to help you learn more about AWS compliance.

In the Customer Compliance Centre, you can read customer compliance stories to discover how companies in regulated industries have solved various compliance, governance, and audit challenges. You can also access compliance whitepapers and documentation on topics such as:

- AWS answers to key compliance questions
- An overview of AWS risk and compliance
- An auditing security checklist

Additionally, the Customer Compliance Centre includes an **auditor learning path**. This learning path is designed for individuals in auditing, compliance, and legal roles who want to learn more about how their internal operations can demonstrate compliance using the AWS Cloud.

To know if you are compliant in AWS, please remember that we follow a shared responsibility. The underlying platform is secure and **AWS can provide documentation on what types of compliance requirements they meet**, through services like AWS Artifact and whitepapers. But, beyond that, what you build on AWS is up to you. You control the architecture of your applications and the solutions you build, and they need to be built with compliance, security, and the shared responsibility model in mind.

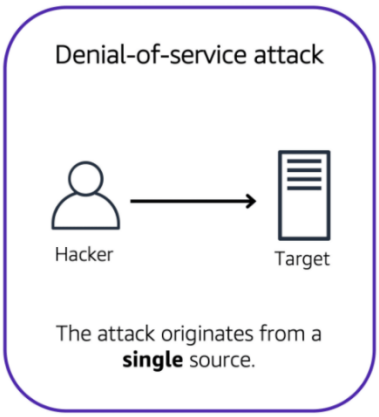
Denial-of-service attacks

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**.

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

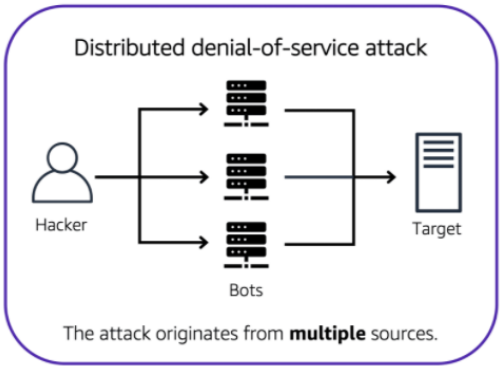


For example, an attacker **might flood a website or application with excessive network traffic until the targeted website or application becomes overloaded and is no longer able to respond**. If the website or application becomes unavailable, this denies service to users who are trying to make legitimate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

Distributed denial-of-service attacks

A single machine attacking your application has no hope of providing enough of an attack by itself, so the distributed part is that the attack leverages other machines around the internet to unknowingly attack your infrastructure. 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.

Examples of DDoS Attacks

UDP Flood

It is based on the helpful parts of the internet, like the National Weather Service. Now anyone can send a small request to the Weather Service, and ask, "Give me weather," and in return, the Weather Service's fleet of machines will send back a massive amount of weather telemetry, forecasts, updates, lots of stuff. So the attack here is simple. The bad actor sends a simple request, give me weather. But it gives a **fake return address** on the request, **your return address**. So now the Weather Service very happily floods your server with megabytes of rain forecasts, and your system could be brought to a standstill, just sorting through the information it never wanted in the first place.

HTTP level attacks

Some attacks are much more sophisticated, like the HTTP level attacks, which look like normal customers asking for normal things like complicated product searches over and over

and over, all coming from an army of zombified bot machines. They ask for so much attention that regular customers can't get in.

Slowloris attack

Imagine standing in line at the coffee shop, when someone in front of you takes seven minutes to order whatever it is they're ordering, and you don't get to order until they finish and get out of your way. Well, Slowloris attack is the exact same thing. Instead of a normal connection, I would like to place an order, the attacker pretends to have a terribly slow connection. You get the picture. Meanwhile, your production servers are standing there waiting for the customer to finish their request so they can dash off and return the result. But until they get the entire packet, they can't move on to the next thread, the next customer. A few Slowloris attackers can exhaust the capacity of your entire front end with almost no effort at all.

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

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

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.

For the sharpest, most sophisticated attacks, AWS offers specialized defence tools called AWS Shield with **AWS WAF**. AWS WAF uses a **web application firewall** to filter incoming traffic for the signatures of bad actors. It has extensive **machine learning capabilities**, and can recognize new threats as they evolve and proactively help defend your system against an ever-growing list of destructive vectors.

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**. **Encryption is the securing of a message or data in a way that only authorized parties can access it.**

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.

For example, server-side encryption at rest is enabled on all DynamoDB table data. And that helps prevent unauthorized access. **DynamoDB's encryption at rest** also integrates with AWS KMS, or Key Management Service, for managing the encryption key that is used to encrypt your tables. That's the key for your door, remember? And without it, you won't be able to access your data.

Similarly, in-transit means that the data is traveling between, say A and B. Where A is the AWS service, and B could be a client accessing the service. Or even another AWS service itself. For example, let's say we have a Redshift instance running. And we want to connect it with a SQL client. We use **secure sockets layer**, or **SSL connections** to encrypt data, and we can use service certificates to validate, and authorize a client. This means that **data is protected when passing between Redshift, and our client.** And this functionality exists in numerous other AWS services such as SQS, S3, RDS, and many more.

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

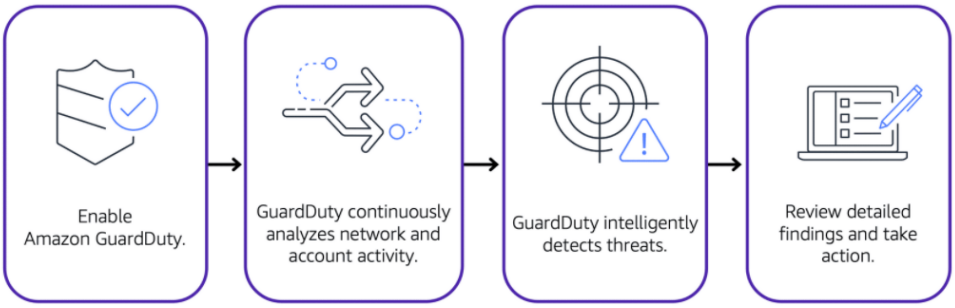
Suppose that the developers at the coffee shop are developing and testing a new ordering application. They want to make sure that they are designing the application in accordance with security best practices. However, they have several other applications to develop, so they cannot spend much time conducting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

It uses integrated threat intelligence such as known **malicious IP addresses, anomaly detection, and machine learning to identify threats more accurately**. The best part is that it runs **independently from your other AWS services**. So it won't affect performance or availability of your existing infrastructure, and workloads.

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.

Summary

Alright, it's time to break down what we just covered. First things first, AWS follows a **shared responsibility model**. AWS is responsible for security of the cloud, and you are responsible for security in the cloud.

With **IAM**, you have users, groups, roles, and policies. Users log in with a username and password and by default they have no permissions. Groups are groupings of users and roles

are identities that you can assume to gain access to temporary credentials and permissions for a configurable amount of time. In order to give permissions to an identity, you need to create **policies** that either explicitly allow or deny a specific action in AWS. With IAM also comes identity federation.

If you have an existing corporate identity store, you can federate those users to AWS, using **role based access**, which allows your users to use one login for both your corporate systems as well as AWS. One final point to remember about IAM is that you should make sure that you turn on **multi-factor authentication** for users, but especially for your root user which has all the permissions by default and cannot be restricted.

Next up, we discussed **AWS Organizations**. With AWS, it's likely you'll have multiple accounts. Accounts are commonly used to isolate workloads, environments, teams, or applications. AWS Organizations helps you manage multiple accounts in a hierarchical fashion.

We then discussed **compliance**. AWS uses third-party auditors to prove its adherence to a wide variety of compliance programs. You can use the **AWS Compliance Center** to find more information on compliance and **AWS Artifact** to **gain access to compliance documents**. The compliance requirements you have will vary from application to application and between areas of operation.

Then we talked about distributed denial-of-service attacks, or **DDoS** attacks, and how to combat them with AWS using **tools like ELB, security groups, AWS Shield, and AWS WAF**. We also talked about encryption. In AWS, you are the owner of your data, and you are responsible for security. That means you need to pay attention to encryption, in transit and at rest.

There are lots of considerations when dealing with security in AWS. Security is AWS's top priority, and will continue to be so. Please make sure you read the documentation on securing your AWS resources, as it does vary from service to service.

Use the **least privilege principle when scoping permissions for users** and roles in IAM, **encrypt your data at every layer, both in transit and at rest.** And make sure you use AWS services to protect your environment.

Quiz

Which statement best describes an IAM policy?

- A document that grants or denies permissions to AWS services and resources. IAM policies provide you with the flexibility to customize users’ levels of access to resources.

An employee requires temporary access to create several Amazon S3 buckets. Which option would be the best choice for this task?

- IAM role

Which statement best describes the principle of least privilege?

- Granting only the permissions that are needed to perform specific tasks. When you grant permissions by following the principle of least privilege, you prevent users or roles from having more permissions than needed to perform specific job tasks.

Which service helps protect your applications against distributed denial-of-service (DDoS) attacks?

- AWS Shield. As network traffic comes into your applications, AWS Shield uses a variety of analysis techniques to detect potential DDoS attacks in real time and automatically mitigates them.

Which task can AWS Key Management Service (AWS KMS) perform?

- Create cryptographic keys

Additional resources

To learn more about the concepts that were explored in Module 6, review these resources.

- Security, Identity, and Compliance on AWS
- Whitepaper: Introduction to AWS Security
- Whitepaper: Amazon Web Services - Overview of Security Processes
- AWS Security Blog
- AWS Compliance
- AWS Customer Stories: Security, Identity, and Compliance

Module 7 – Monitoring and Analytics

Contents

Learning objectives 1
Introduction 1
Amazon CloudWatch 2
CloudWatch alarms 2
CloudWatch dashboard 3
AWS CloudTrail 3
Example: AWS CloudTrail event 4
CloudTrail Insights 5
AWS Trusted Advisor 5
AWS Trusted Advisor dashboard 5
Summary 6
Quiz 7
Additional resources 7

Learning objectives

In this module, you will learn how to:

- Summarize approaches to monitoring your AWS environment.
- Describe the benefits of Amazon CloudWatch.
- Describe the benefits of AWS CloudTrail.
- Describe the benefits of AWS Trusted Advisor.

Introduction

Every business, including this coffee shop, can use metrics to measure how well systems and processes are running. This idea of observing systems, collecting metrics, evaluating those metrics over time, and then using them to make decisions or take actions, is what we call monitoring.

It's important to set up monitoring in the cloud. With the elastic nature of AWS services that dynamically scale up and down, you'll want to keep a close pulse on your AWS resources to ensure that your systems are running as expected.

For example, if an EC2 instance is being over-utilized, you can trigger a scaling event that automatically would launch another EC2 instance. Or if an application starts sending error responses at an unusually high rate, you can alert an employee to go take a look at what's going on.

In the next few videos, we will cover a variety of tools that help you monitor your AWS environment. This monitoring will help you measure how your systems are performing, alert you when things aren't right, and even help you debug and troubleshoot issues as they come along.

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

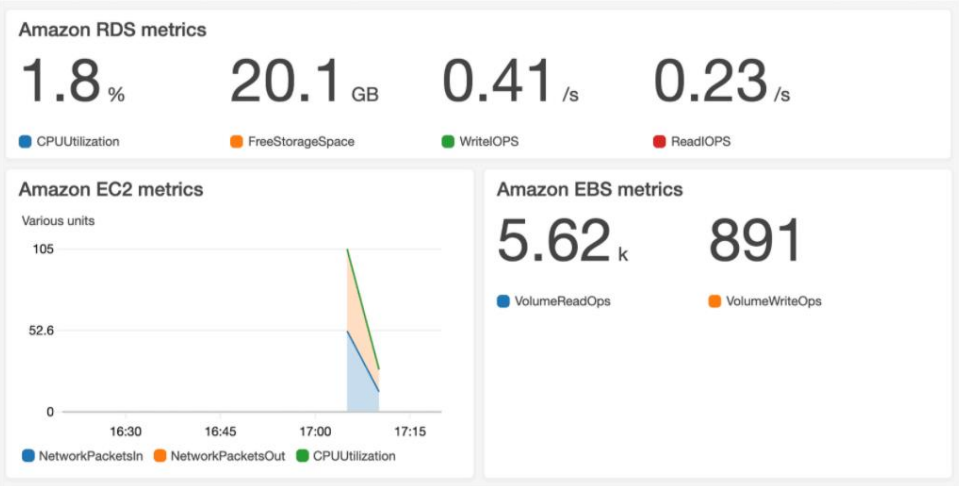
With CloudWatch, you can create **alarms** that automatically perform actions if the value of your metric has gone above or below a predefined threshold.

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.

Even better, CloudWatch alarms are integrated with **SNS**.

CloudWatch dashboard



The CloudWatch **dashboard** feature enables you to access all the metrics for your resources from a single location. This enables you to collect metrics and logs from all your AWS resources applications, and services that run on AWS and on-premises servers, helping you break down silos so that you can easily gain system-wide visibility. 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.

You can get visibility across your applications, infrastructure, and services, which means you gain insights across your distributed stack so you can correlate and visualize metrics and logs to quickly pinpoint and resolve issues. This in turn means you can reduce **mean time to resolution, or MTTR**, and improve **total cost of ownership, or TCO**. So in our coffee shop, if the MTTR of cleaning hours for restaurant machines is shorter then we can save on TCO with them. This means freeing up important resources like developers to focus on adding business value.

Lastly, you can drive insights to optimize applications and operational resources. By, for example, aggregating usage across an entire fleet of EC2 instances to derive operational and utilization insights. And now our staff can focus on serving coffee versus constantly cleaning machines before they are due to be cleaned.

AWS CloudTrail

AWS CloudTrail is a comprehensive API auditing tool that 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.

CloudTrail can save those logs indefinitely in secure S3 buckets. In addition, with tamper-proof methods like Vault Lock, you then can show absolute provenance of all of these critical security audit logs.

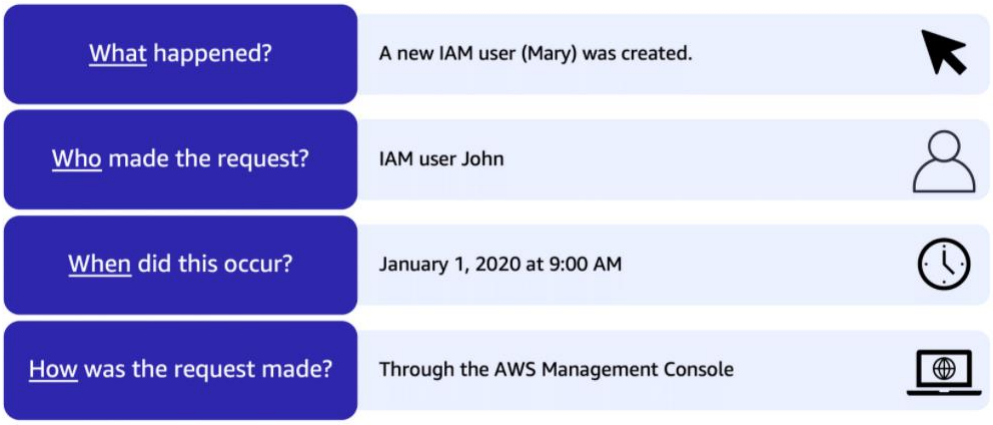
Example: AWS CloudTrail event

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 an automated 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.

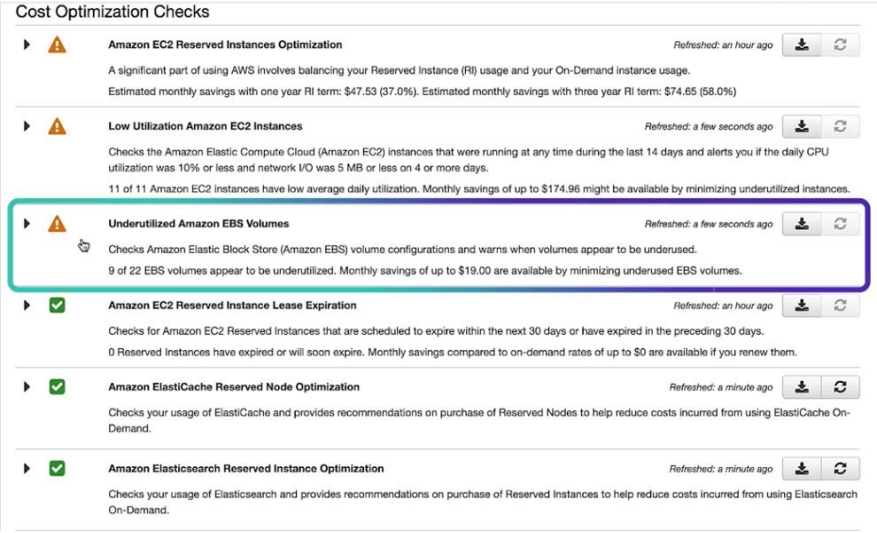
AWS Trusted Advisor dashboard

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.

For each category:

- The green check indicates the number of items for which it detected **no problems**.
- The orange triangle represents the number of recommended **investigations**.
- The red circle represents the number of recommended **actions**.

Example:



Some checks are free and are included in your AWS account, and others are available depending on the level of your support plan. Some examples of checks are, if you don't have multi-factor authentication turned on for your root user, it's going to let you know. If you have **underutilized EC2 instances** that might be able to be turned off in order to save money, or if you have **EBS volumes that haven't been backed up** in a reasonable amount of time, it will let you know that, too.

Trusted Advisor can help point you in the right direction when it comes to the five pillars. You can set up email alerts that go out to billing, operations, and security contacts, as checks get run in your account. Make sure you have Trusted Advisor turned on so that you too can start taking action to optimize your AWS account.

Summary

Understanding what is happening in your environment is key to maintaining efficient, secure, and compliant applications. We discussed how **CloudWatch** can provide near real-time understanding of how your system is behaving, including being alerted to conditions that require your attention.

CloudWatch also gives you the ability to look at those metrics over time as you tune your system for maximum performance.

We discussed how **CloudTrail** can help you know exactly who did what, when, and from where. It answers all of your AWS **auditing** questions, except why they did it.

And finally, we looked at **Trusted Advisor** that compiles a **quick dashboard** of over 40 common concerns around cost, performance, security, and resilience in an actionable dashboard.

Quiz

Which actions can you perform using Amazon CloudWatch? (Select TWO.)

- Monitor your resources' utilization and performance
- Access metrics from a single dashboard

Which service enables you to review the security of your Amazon S3 buckets by checking for open access permissions?

- AWS Trusted Advisor

Which categories are included in the AWS Trusted Advisor dashboard? (Select TWO.)

- Performance
- Fault tolerance

Additional resources

To learn more about the concepts that were explored in Module 7, review these resources.

- Management and Governance on AWS
- Monitoring and Observability
- Configuration, Compliance, and Auditing
- AWS Management & Governance Blog
- Whitepaper: AWS Governance at Scale

Module 8 – Pricing and Support

Contents

Learning objectives 2
AWS Free Tier 2
How AWS pricing works (AWS pricing concepts) 3
AWS Pricing Calculator 4
AWS pricing examples 4
Billing Dashboard 7
Consolidated billing 8
AWS Budgets 8
Example: AWS Budgets 8
AWS Cost Explorer 10
Example: AWS Cost Explorer 10
AWS Support 11
Basic Support 11
Developer, Business, and Enterprise Support 11
Technical Account Manager (TAM) 13
AWS Marketplace 13
AWS Marketplace categories 14
Summary 14
Quiz 15
Additional resources 15

Learning objectives

In this module, you will learn how to:

- Describe AWS pricing and support models.
- Describe the AWS Free Tier.
- Describe key benefits of AWS Organizations and consolidated billing.
- Explain the benefits of AWS Budgets.
- Explain the benefits of AWS Cost Explorer.
- Explain the primary benefits of the AWS Pricing Calculator.
- Distinguish between the various AWS Support Plans.
- Describe the benefits of AWS Marketplace.

AWS Free Tier

The AWS Free Tier enables you to begin using certain services without having to worry about incurring costs for the specified period.

Three types of offers are available:

- Always Free
- 12 Months Free
- Trials

For each free tier offer, make sure to review the specific details about exactly which resource types are included.

Always Free

These offers do not expire and are available to all AWS customers.

For example, AWS Lambda allows 1 million free requests and up to 3.2 million seconds of compute time per month. Amazon DynamoDB allows 25 GB of free storage per month.

12 Months Free

These offers are free for **12 months** following your initial sign-up date to AWS (12 Months Free category).

Examples include specific amounts of Amazon S3 Standard Storage, thresholds for monthly hours of Amazon EC2 compute time, and amounts of Amazon CloudFront data transfer out.

Trials

Short-term free trial offers start from the date you activate a particular service. The length of each trial might vary by number of days or the amount of usage in the service. For example, Amazon Inspector offers a 90-day free trial. Amazon Lightsail (a service that enables you to run virtual private servers) offers 750 free hours of usage over a 30-day period.

Examples

Let's illustrate with a few examples. Let's take AWS Lambda, our serverless compute option. As of March 2020, it allows for one million free invocations per month. That means if you stay under one million invocations, it's always free. This Free Tier never expires.

Another example is **S3, our object store service**. It's **free for 12 months for up to five gigs of storage**. Thereafter, you'll incur a cost. Great for trying out a static website on S3, I might say. And the last example is Lightsail where you can deploy ready-made application stacks. We offer a one month trial of up to 750 hours of usage.

Some other services that qualify under the free tier are SageMaker, Comprehend Medical, DynamoDB, SNS, Cognito, and so much more. If you want to see the full list of 60 or so services, please check out our Resources section.

How AWS pricing works (AWS pricing concepts)

AWS offers a range of cloud computing services with pay-as-you-go pricing.

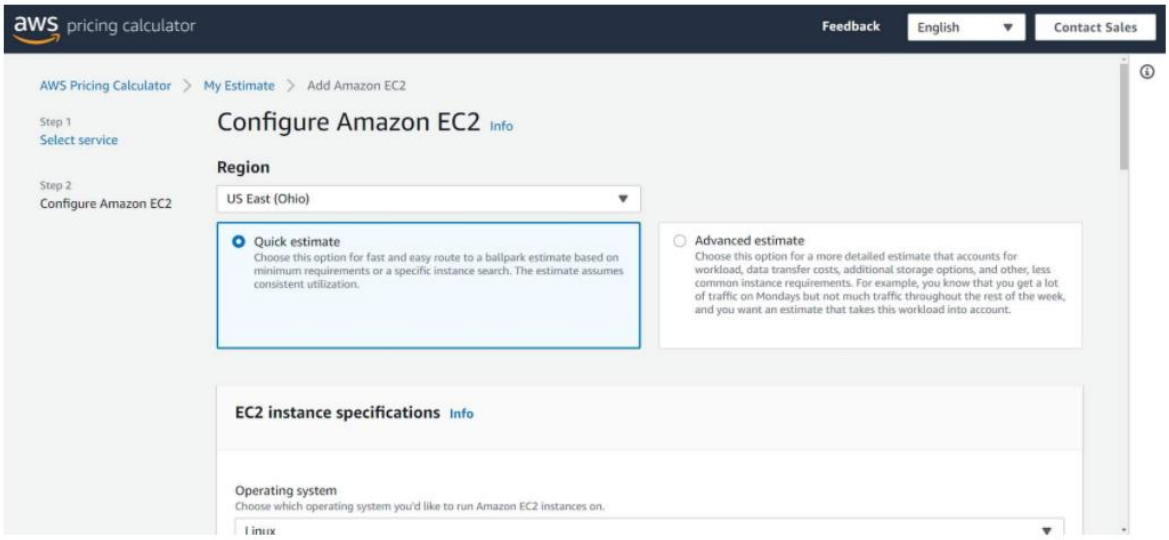
Pay for what you use.
For each service, you pay for exactly the amount of resources that you actually use, without requiring long-term contracts or complex licensing.

Pay less when you reserve.
Some services offer **reservation options** that provide a significant **discount** compared to On-Demand Instance pricing. For example, suppose that your company is using Amazon EC2 instances for a **workload that needs to run continuously**. You might choose to run this workload on Amazon EC2 Instance **Savings Plans**, because the plan allows you to save up to 72% over the equivalent On-Demand Instance capacity.

Pay less with volume-based discounts when you use more.
Some services offer tiered pricing, so the per-unit cost is incrementally lower with increased usage. For example, the more Amazon S3 storage space you use, the less you pay for it per GB.

AWS Pricing Calculator

The **AWS Pricing Calculator** lets you explore AWS services and **create an estimate** for the cost of your use cases on AWS. You can organize your AWS estimates by groups that you define. A group can reflect how your company is organized, such as providing estimates by cost centre. When you have created an estimate, you can save it and generate a link to share it with others.



Suppose that your company is interested in using Amazon EC2. However, you are not yet sure which AWS Region or instance type would be the most cost-efficient for your use case. In the AWS Pricing Calculator, you can enter details such as the kind of operating system you need, memory requirements, and input/output (I/O) requirements. By using the **AWS Pricing Calculator**, you can review an **estimated comparison of different EC2 instance types across AWS Regions.**

AWS pricing examples

This section presents a few examples of pricing in AWS services.

AWS Lambda

For AWS Lambda, you are charged based on the **number of requests for your functions and the time that it takes for them to run**. AWS Lambda allows 1 million free requests and up to 3.2 million seconds of compute time per month.

You can save on AWS Lambda costs by signing up for a **Compute Savings Plan**. A Compute Savings Plan offers lower compute costs in exchange for committing to a consistent amount of usage over a 1-year or 3-year term. This is an example of **paying less when you reserve**.

If you have used AWS Lambda in multiple AWS Regions, you can view the itemized charges by Region on your bill.

In this example, all the AWS Lambda usage occurred in the Northern Virginia Region. The bill lists separate charges for the number of requests for functions and their duration.

Both the **number of requests** and the **total duration of requests** in this example are under the thresholds in the AWS Free Tier, so the account owner would not have to pay for any AWS Lambda usage in this month.

▾ **Lambda**		**$0.00**
▾ **US East (N. Virginia)**		**$0.00**
AWS Lambda Lambda-GB-Second		$0.00
AWS Lambda - Compute Free Tier - 400,000 GB-Seconds - US East (Northern Virginia)	254.575 seconds	$0.00
AWS Lambda Request		$0.00
AWS Lambda - Requests Free Tier - 1,000,000 Requests - US East (Northern Virginia)	680.000 Requests	$0.00

Amazon EC2

With Amazon EC2, you pay for only the compute time that you use while your instances are running.

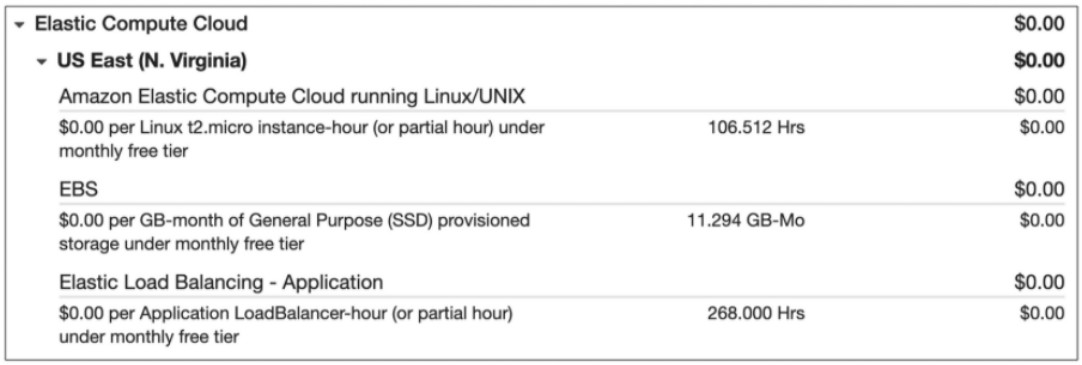
For some workloads, you can significantly reduce Amazon EC2 costs by using Spot Instances. For example, suppose that you are running a **batch processing job that is able to withstand interruptions**. Using a **Spot Instance** would provide you with up to 90% cost savings while still meeting the availability requirements of your workload.

You can find additional cost savings for Amazon EC2 by considering Savings Plans and Reserved Instances.

The service charges in this example include details for the following items:

- Each Amazon EC2 **instance type** that has been used
- The **amount of Amazon EBS storage space** that has been provisioned
- The **length of time that Elastic Load Balancing** has been used

In this example, all the usage amounts are under the thresholds in the AWS Free Tier, so the account owner would not have to pay for any Amazon EC2 usage in this month.



Elastic Compute Cloud		**$0.00**
US East (N. Virginia)		**$0.00**
Amazon Elastic Compute Cloud running Linux/UNIX		$0.00
$0.00 per Linux t2.micro instance-hour (or partial hour) under monthly free tier	106.512 Hrs	$0.00
EBS		$0.00
$0.00 per GB-month of General Purpose (SSD) provisioned storage under monthly free tier	11.294 GB-Mo	$0.00
Elastic Load Balancing - Application		$0.00
$0.00 per Application LoadBalancer-hour (or partial hour) under monthly free tier	268.000 Hrs	$0.00

Amazon S3

For Amazon S3 pricing, consider the following cost components:

- **Storage -** You pay for only the storage that you use. You are charged the rate to store objects in your Amazon S3 buckets based on your objects' **sizes**, **storage classes**, and **how long** you have stored each object during the month.

- **Requests and data retrievals -** You pay for requests made to your Amazon S3 objects and buckets. For example, suppose that you are storing photo files in Amazon S3 buckets and hosting them on a website. **Every time a visitor requests the website that includes these photo files, this counts towards requests you must pay for**.

- **Data transfer -** There is no cost to transfer data between different Amazon S3 buckets or from Amazon S3 to other services within the same AWS Region. However, you **pay for data that you transfer into and out of Amazon S3**, with a few exceptions. There is no cost for data transferred into Amazon S3 from the internet or out to Amazon CloudFront. There is also no cost for data transferred out to an Amazon EC2 instance in the same AWS Region as the Amazon S3 bucket.

- **Management and replication -** You pay for the **storage management features** that you have enabled on your account's Amazon S3 buckets. These features include Amazon S3 inventory, analytics, and object tagging.

The AWS account in this example has used Amazon S3 in two Regions: Northern Virginia and Ohio. For each Region, itemized charges are based on the following factors:

- The number of **requests** to **add** or **copy** objects into a bucket
- The number of **requests** to **retrieve** objects from a bucket
- The amount of **storage space** used

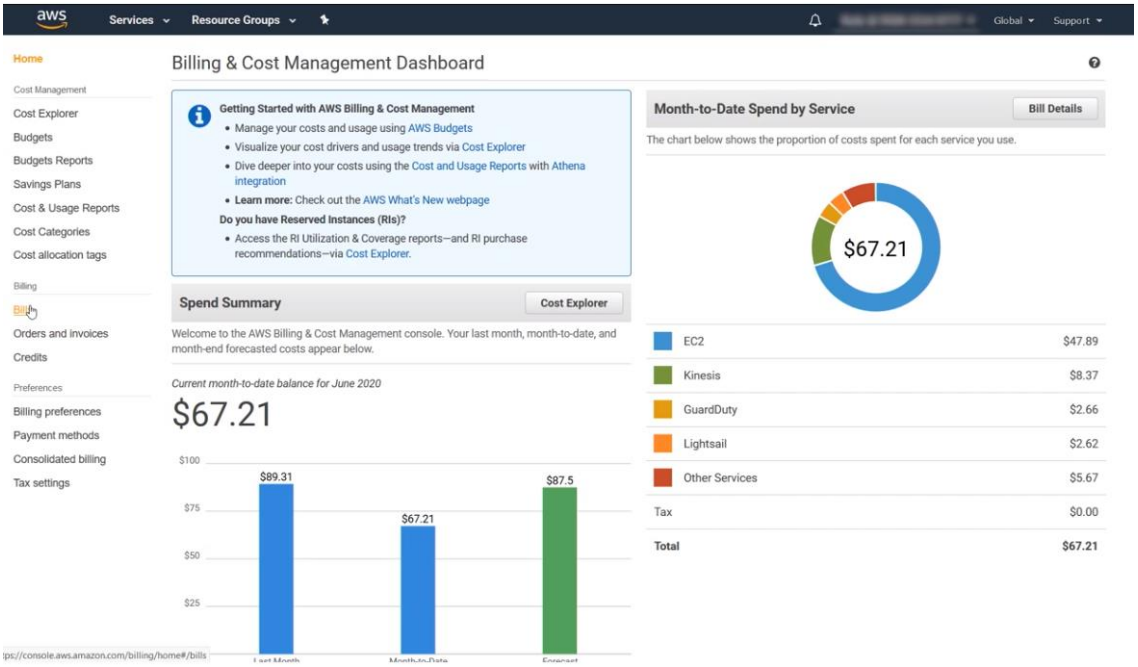
All the usage for Amazon S3 in this example is under the AWS Free Tier limits, so the account owner would not have to pay for any Amazon S3 usage in this month.

Simple Storage Service		**$0.00**
US East (N. Virginia)		**$0.00**
Amazon Simple Storage Service Requests-Tier1		$0.00
$0.00 per request - PUT, COPY, POST, or LIST requests under the monthly global free tier	185.000 Requests	$0.00
Amazon Simple Storage Service Requests-Tier2		$0.00
$0.00 per request - GET and all other requests under the monthly global free tier	923.000 Requests	$0.00
Amazon Simple Storage Service TimedStorage-ByteHrs		$0.00
$0.000 per GB - storage under the monthly global free tier	0.159 GB-Mo	$0.00
US East (Ohio)		**$0.00**
Amazon Simple Storage Service USE2-Requests-Tier2		$0.00
$0.00 per request - GET and all other requests under the monthly global free tier	4.000 Requests	$0.00
Amazon Simple Storage Service USE2-TimedStorage-ByteHrs		$0.00
$0.000 per GB - storage under the monthly global free tier	0.000001 GB-Mo	$0.00

Billing Dashboard

Use the **AWS Billing & Cost Management dashboard** to pay your AWS bill, monitor your usage, and analyse and control your costs.

- Compare your current month-to-date balance with the previous month, and get a forecast of the next month based on current usage.
- View month-to-date spend by service.
- View Free Tier usage by service.
- Access **Cost Explorer** and create budgets.
- Purchase and manage Savings Plans.
- Publish AWS Cost and Usage Reports.



Consolidated billing

In an earlier module, you learned about **AWS Organizations**, a service that enables you to manage multiple AWS accounts from a central location. AWS Organizations also provides the option for **consolidated billing**.

The consolidated billing feature of AWS Organizations enables you to **receive a single bill** for all AWS accounts in your organization. By consolidating, you can easily track the combined costs of all the linked accounts in your organization. The default maximum number of accounts allowed for an organization is **4**, but you can contact AWS Support to increase your quota, if needed.

On your monthly bill, you can **review itemized charges incurred by each account**. This enables you to have greater **transparency** into your organization's accounts while still maintaining the **convenience** of receiving a single monthly bill.

Another benefit of consolidated billing is the ability to **share bulk discount pricing**, Savings Plans, and Reserved Instances across the accounts in your organization. For instance, one account might not have enough **monthly usage to qualify for discount pricing**. However, when multiple accounts are combined, their **aggregated usage may result in a benefit** that applies across all accounts in the organization.

The best part about this is that the feature is free and easy to use. So it simplifies the billing process, lets you **share savings across accounts** and doesn't cost you any extra money.

AWS Budgets

As you're ramping up your AWS deployments, you probably want to make sure you're not spending an unbudgeted amount. As with most companies, you want to track costs and make sure you keep within your limits.

In **AWS Budgets**, you can create budgets to plan your service usage, service costs, and instance reservations.

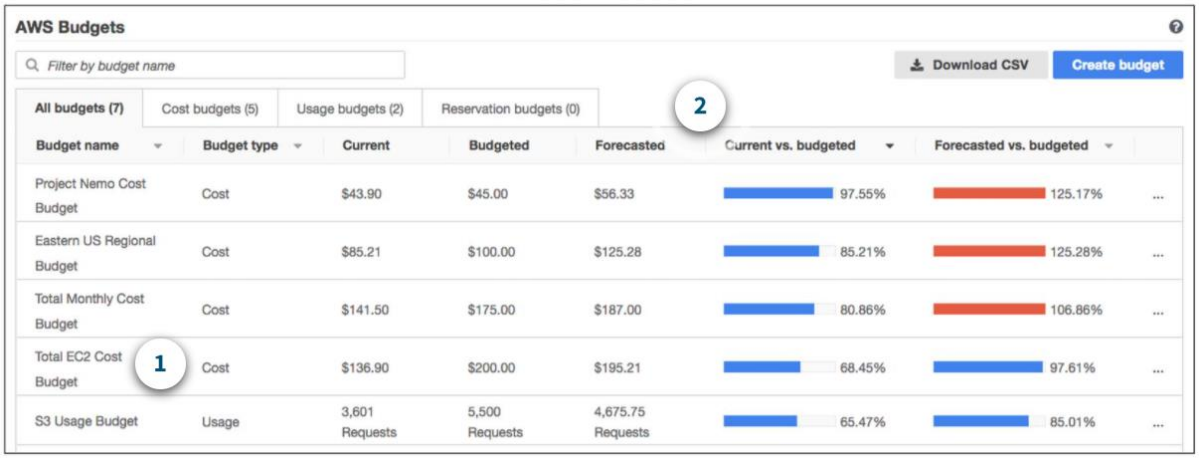
The information in AWS Budgets **updates three times a day**. This helps you to accurately determine how close your usage is to your budgeted amounts or to the AWS Free Tier limits.

In AWS Budgets, you can also set custom alerts when your usage exceeds (or is forecasted to exceed) the budgeted amount.

Example: AWS Budgets

Suppose that you have set a budget for Amazon EC2. You want to ensure that your company's usage of Amazon EC2 does not exceed $200 for the month.

In AWS Budgets, you could set a custom budget to notify you when your usage has reached half of this amount ($100). This setting would allow you to receive an alert and decide how you would like to proceed with your continued use of Amazon EC2.

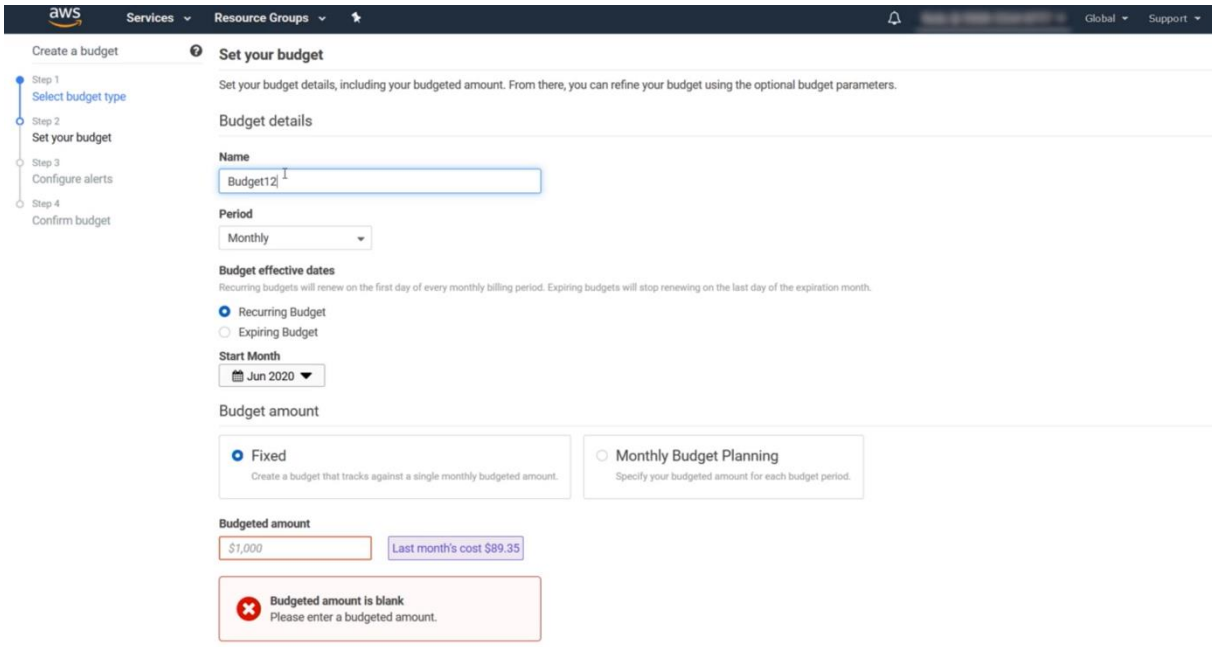


In this sample budget, you can review the following important details:

- The current amount that you have incurred for Amazon EC2 so far this month ($136.90)
- The amount that you are forecasted to spend for the month ($195.21), based on your usage patterns

You can also review comparisons of your current vs. budgeted usage, and forecasted vs. budgeted usage.

For example, in the top row of this sample budget, the forecasted vs. budgeted bar is at 125.17%. The reason for the increase is that the forecasted amount ($56.33) exceeds the amount that had been budgeted for that item for the month ($45.00).



AWS Cost Explorer

As we have already discussed, AWS has a variable cost model, and you only pay for what you use. You don't have one fixed billed amount at the end of every month. Instead, it fluctuates with the resources you use and how you use them. Because of this cost model, it's really important that you can **drill down into your bill** and see just how you are spending money.

AWS Cost Explorer is a console-based tool that enables you to **visualize, understand, and manage your AWS costs and usage over time**.

AWS Cost Explorer includes a default report of the costs and usage for your **top five** cost-accruing AWS services. You can apply custom filters and groups to analyse your data. For example, you can view resource usage at the **hourly level**.

It will show you which services you are spending the most money on, and it gives you 12 months of historical data, so you can track your spending over time. That way, if you see a bump in spending on, say, EC2 from October to December, you can then use that data to go on and figure out why exactly that happened.

Example: AWS Cost Explorer

This example of the **AWS Cost Explorer dashboard** displays monthly costs for Amazon EC2 instances over a 6-month period. The bar for each month separates the costs for different Amazon EC2 instance types (such as t2.micro or m3.large).

By analysing your AWS costs over time, you can make informed decisions about future costs and how to plan your budgets.

One important grouping to note is to **group by tag**. Many resources in AWS are taggable. Tags are essentially user-defined key-value pairs. So you can tag an EC2 instance with a specific project name or a database with the same project name, and then you can come into the AWS Cost Explorer, filter by tag, and see all expenses associated with that tag.

So as you can see, Cost Explorer gives you some powerful defaults for reports, but you can build your own custom ones as well. This will help you identify cost drivers and take action when necessary to curb spending. **Cost optimization** is a priority you should be paying close attention to, and you can use the Cost Explorer to help get you going in the right direction.

AWS Support

AWS offers four different **Support plans** to help you troubleshoot issues, lower costs, and efficiently use AWS services.

You can choose from the following Support plans to meet your company's needs:

- Basic
- Developer
- Business
- Enterprise

Basic Support

Basic Support is free for all AWS customers. It includes access to customer service, whitepapers, documentation, and support communities. With Basic Support, you can also contact AWS for billing questions and service limit increases.

With Basic Support, you have access to a **limited** selection of **AWS Trusted Advisor checks**. Additionally, you can use the **AWS Personal Health Dashboard**, a tool that provides alerts and remediation guidance when AWS is experiencing events that may affect you.

These functions are free for everyone, but as you begin to move mission critical workloads into AWS, we offer higher levels of support to match your levels of need. If your company needs support beyond the Basic level, you could consider purchasing Developer, Business, or Enterprise Support.

Developer, Business, and Enterprise Support

The Developer, Business, and Enterprise Support plans include all the benefits of Basic Support, in addition to the ability to **open an unrestricted number of technical support cases**. These three Support plans have **pay-by-the-month pricing** and require no long-term contracts.

The information in this course highlights only a selection of details for each Support plan. A complete overview of what is included in each Support plan, including pricing for each plan, is available on the AWS Support site.

In general, for pricing, the Developer plan has the lowest cost, the Business plan is in the middle, and the Enterprise plan has the highest cost.

Developer Support

Customers in the **Developer Support** plan have access to features such as:

- Best practice guidance
- Client-side diagnostic tools
- **Building-block architecture support**, which consists of guidance for how to use AWS offerings, features, and services together

Plus, you can now **email customer support directly with a 24 hour response time** on any questions you have. And responses of less than **12 hours in case your systems are impaired**.

For example, suppose that your company is exploring AWS services. You've heard about a few different AWS services. However, you're unsure of how to potentially use them together to build applications that can address your company's needs. In this scenario, the building-block architecture support that is included with the Developer Support plan could help you to identify opportunities for combining specific services and features.

Business Support

Customers with a **Business Support** plan have access to additional features, including:

- **Use-case guidance** to identify AWS offerings, features, and services that can best support your specific needs
- **All AWS Trusted Advisor** checks
- Limited support for third-party software, such as common operating systems and application stack components

Suppose that your company has the Business Support plan and wants to install a common third-party operating system onto your Amazon EC2 instances. You could contact AWS Support for assistance with installing, configuring, and troubleshooting the operating system. For advanced topics such as optimizing performance, using custom scripts, or resolving security issues, you may need to contact the third-party software provider directly.

You are given direct phone access to our support team that has a **four hour response SLA**. If your production system is impaired, and a one hour SLA for production systems down. Additionally, as part of the Business tier, we provide access to infrastructure event management, where for an extra fee, we can help you plan for massive events like brand new launches or global advertising blitzes.

Enterprise Support

In addition to all the features included in the Basic, Developer, and Business Support plans, customers with an **Enterprise Support** plan have access to features such as:

- Application architecture guidance, which is a consultative relationship to support your company's specific use cases and applications
- Infrastructure event management: A short-term engagement with AWS Support that helps your company gain a better understanding of your use cases. This also provides your company with architectural and scaling guidance.
- A **Technical Account Manager**
- **15 minute SLA** for business critical workloads

Technical Account Manager (TAM)

The Enterprise Support plan includes access to a **Technical Account Manager (TAM)**. If your company has an Enterprise Support plan, the TAM is your primary point of contact at AWS. They provide guidance, architectural reviews, and ongoing communication with your company as you plan, deploy, and optimize your applications.

Your TAM provides expertise across the full range of AWS services. They help you design solutions that efficiently use multiple services together through an integrated approach.

For example, suppose that you are interested in developing an application that uses several AWS services together. Your TAM could provide insights into how to best use the services together. They achieve this, while aligning with the specific needs that your company is hoping to address through the new application.

AWS Marketplace

AWS Marketplace is a **curated digital catalogue** that includes thousands of **software listings** from **independent third party software vendors**. You can use AWS Marketplace to find, test, and buy software that runs on AWS. This allows you to accelerate innovation while rapidly and securely deploying a wide range of solutions, while also reducing your total cost of ownership.

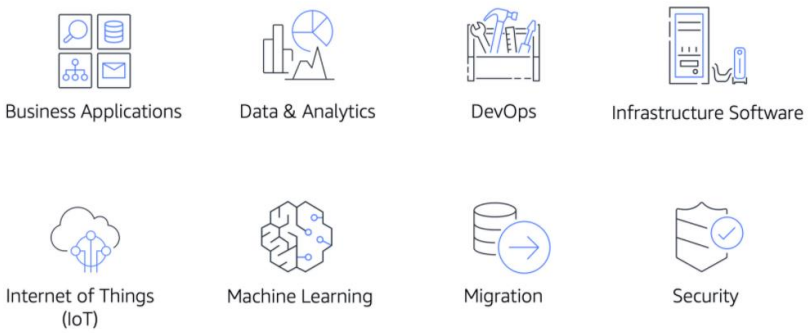
The first key way that the AWS Marketplace helps customers, is that instead of needing to build, install and maintain the foundational infrastructure needed to run these third party applications in the marketplace, customers have options like **one-click deployment** that allows them to quickly procure and use products from thousands of software sellers right when you need them.

For each listing in AWS Marketplace, you can access detailed information on pricing options, available support, and reviews from other AWS customers.

More importantly, as you move forward, most vendors in the marketplace also offer **on-demand pay-as-you-go options.** These **flexible pricing** plans give you more options to pay for the software, the way you actually use it without wasted unused licenses weighing down your balance sheets.

You can also explore software solutions by industry and use case. For example, suppose that your company is in the healthcare industry. In AWS Marketplace, you can review use cases that software helps you to address, such as implementing solutions to protect patient records or using machine learning models to analyse a patient's medical history and predict possible health risks.

AWS Marketplace categories



AWS Marketplace offers products in several categories, such as Infrastructure Products, Business Applications, Data Products, and DevOps.

Within each category, you can narrow your search by browsing through product listings in subcategories. For example, subcategories in the DevOps category include areas such as Application Development, Monitoring, and Testing.

Summary

Well, you've heard a lot about pricing and support, and it's probably a lot more than you initially thought. You learned about the **pay-as-you-go** nature of using AWS cloud resources. We discussed the difference between on-premises and cloud costs, and we even showed you how you can get your feet wet with the **free tier** offered with most AWS services.

We talked about how AWS Organizations can be used and how it can help you with **consolidated billing** of multiple AWS accounts. We covered **AWS Budgets, Cost Explorer**. We even explored AWS **Console billing**, so you could get familiar with it, and then switched gears to talk about the **different support models** offered on AWS. Very handy if you're looking for assistance on your cloud journey.

Speaking of assistance, we introduced you to the expansive AWS partner ecosystem so you can find partners to help you with your workloads. And lastly, we touched upon **AWS Marketplace if you're looking for click-and-go services.**

Quiz

Which action can you perform with consolidated billing?

- Combine usage across accounts to receive volume pricing discounts.

Which pricing tool is used to visualize, understand, and manage your AWS costs and usage over time?

- AWS Cost Explorer

Which pricing tool enables you to receive alerts when your service usage exceeds a threshold that you have defined?

- AWS Budgets

Your company wants to receive support from an AWS Technical Account Manager (TAM). Which support plan should you choose?

- Enterprise

Which service or resource is used to find third-party software that runs on AWS?

- AWS Marketplace

Additional resources

To learn more about the concepts that were explored in Module 8, review these resources.

- AWS Pricing
- AWS Free Tier
- AWS Cost Management
- Whitepaper: How AWS Pricing Works
- Whitepaper: Introduction to AWS Economics
- AWS Support
- AWS Knowledge Centre

Module 9 – Migration and Innovation

Contents

Learning objectives 1
Introduction 1
Six core perspectives of the AWS Cloud Adoption Framework 2
6 strategies for migration 5
AWS Snow Family members 6
Innovation with AWS Services 8
Summary 9
Quiz 10
Additional resources 10

Learning objectives

In this module, you will learn how to:

- Understand migration and innovation in the AWS Cloud.
- Summarize the AWS Cloud Adoption Framework (AWS CAF).
- Summarize the six key factors of a cloud migration strategy.
- Describe the benefits of AWS data migration solutions, such as AWS Snowcone, AWS Snowball, and AWS Snowmobile.
- Summarize the broad scope of innovative solutions that AWS offers.

Introduction

So looks like we've established how to get going on the AWS Cloud, but what if you have existing deployments in on-premises environments or have started a cloud journey without AWS? Well, we'd love to help you move across, especially since you can take advantage of some real savings by moving.

This brings us to migration and innovation. We'll show you all the options from migration tools, the **AWS Cloud Adoption Framework**, and even the **Snow Family**, which are physical devices to migrate data in and out of AWS.

So before we adjourn, stay glued to your seats for information on how to move from on-premises environment or the cloud to AWS. We'll even cover **the six Rs of migration.**

Six core perspectives of the AWS Cloud Adoption Framework

Migrating to the cloud is a process. You don't just snap your fingers and have everything magically hosted in AWS. It takes a lot of effort to get applications migrated to AWS, and having a successful cloud migration is something that requires expertise.

What position you hold in your organization will impact the things that you need to know or help with for your migration. If you are a developer, your role and viewpoint will be much different than a cloud architect, business analyst or financial analyst. Different types of people bring different perspectives to the table for migration, and you want to harness those different perspectives and make sure everyone is on the same page.

You also want to ensure that you have the right talent to help support your migration, so HR will need to hire at the correct rate to enable your migration. This can be a lot to keep track of, and someone new to the cloud might not think of all the different people who need to be involved. The AWS professional services team has created something called the AWS Cloud Adoption Framework that can help you manage this process through guidance.

At the highest level, the **AWS Cloud Adoption Framework (AWS CAF)** organizes guidance into six areas of focus, called **Perspectives**. Each Perspective addresses distinct responsibilities. The planning process helps the right people across the organization prepare for the changes ahead.

In general, the **Business**, **People**, and **Governance** Perspectives focus on business capabilities, whereas the **Platform**, **Security**, and **Operations** Perspectives focus on technical capabilities.

Business Perspective

The **Business Perspective** ensures that **IT aligns with business needs** and that IT investments link to key business results.

Use the Business Perspective to create a **strong business case** for cloud adoption and prioritize cloud adoption initiatives. Ensure that your business strategies and goals align with your IT strategies and goals.

Common roles in the Business Perspective include:

- Business managers
- Finance managers
- Budget owners
- Strategy stakeholders

People Perspective

The **People Perspective** supports development of an **organization-wide change management** strategy for successful cloud adoption.

Use the People Perspective to evaluate organizational structures and roles, new skill and process requirements, and identify gaps. This helps prioritize **training, staffing, and organizational changes.**

Common roles in the People Perspective include:

- Human resources
- Staffing
- People managers

Governance Perspective

The **Governance Perspective** focuses on the **skills and processes to align IT strategy with business strategy**. This ensures that you maximize the business value and minimize risks.

Use the Governance Perspective to understand how to update the staff skills and processes necessary to ensure business governance in the cloud. Manage and measure cloud investments to evaluate business outcomes.

Common roles in the Governance Perspective include:

- Chief Information Officer (CIO)
- Program managers
- Enterprise architects
- Business analysts
- Portfolio managers

Platform Perspective

The **Platform Perspective** includes **principles and patterns for implementing** new solutions on the cloud, and **migrating on-premises workloads** to the cloud.

Use a variety of **architectural** models to understand and communicate the structure of IT systems and their relationships. Describe the architecture of the target state environment in detail.

Common roles in the Platform Perspective include:

- Chief Technology Officer (CTO)
- IT managers
- Solutions architects

Security Perspective

The **Security Perspective** ensures that the organization **meets security objectives** for visibility, auditability, control, and agility.

Use the AWS CAF to structure the **selection and implementation of security controls and permissions** that meet the organization's needs.

Common roles in the Security Perspective include:

- Chief Information Security Officer (CISO)
- IT security managers
- IT security analysts

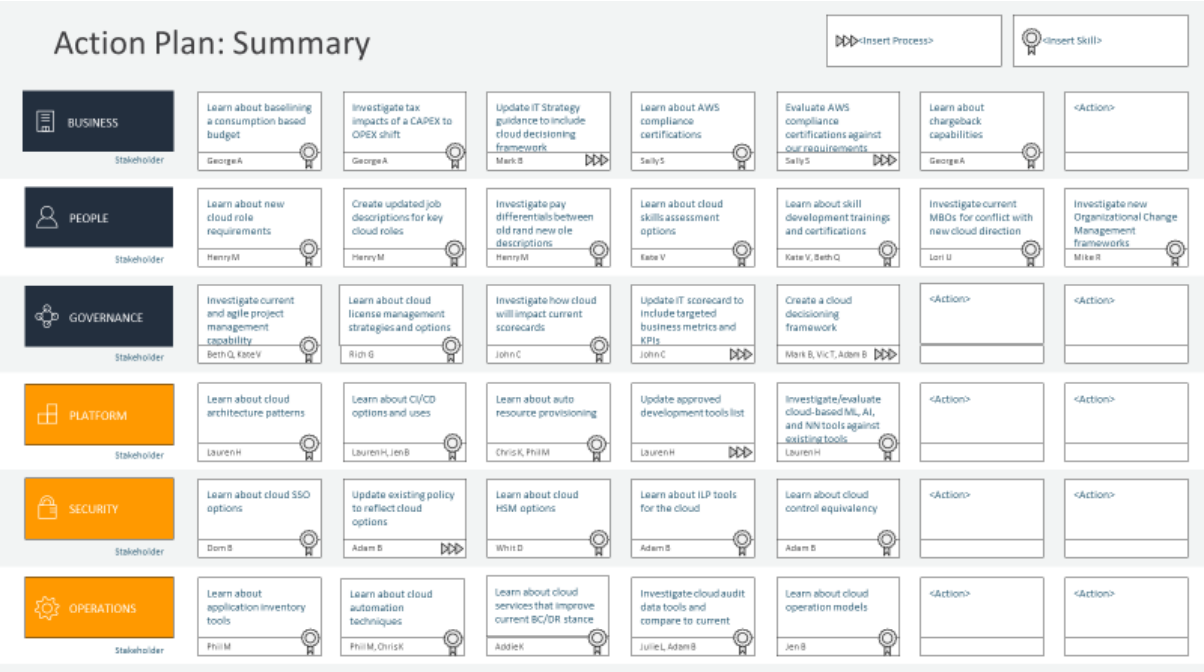
Operations Perspective

The **Operations Perspective** helps you to **enable, run, use, operate, and recover IT workloads** to the level agreed upon with your **business stakeholders**.

Define how day-to-day, quarter-to-quarter, and year-to-year business is conducted. Align with and support the operations of the business. The AWS CAF helps these stakeholders define current **operating procedures** and identify the **process changes** and training needed to implement successful cloud adoption.

Common roles in the Operations Perspective include:

- IT operations managers
- IT support managers



6 strategies for migration

Every application, or application groups, if they're tightly coupled, will have six possible options when it comes to your enterprise migration. We call these the six R's. Once you've gone through the discovery phase and know exactly what you have in your existing environment, you decide which option among the six R's is the best fit based on time, cost, priority, and criticality.

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 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, 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 i.e. you are not touching any core code in the process, and no new dev efforts are involved here. For example, you could take your existing MySQL database and replatform it onto RDS MySQL, without any code changes at all. Or even consider upgrading to Amazon Aurora. This gives significant benefit to your DBA team as well as improved performance without any code changes.

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. **Dramatic changes** to your architecture can be very beneficial to your enterprise but this will come at the **highest initial cost** in terms of planning and human effort.

Repurchasing involves **moving from a traditional license to a software-as-a-service** model i.e. it involves replacing an existing application with a cloud-based version, such as software found in AWS Marketplace. This is common for companies looking to **abandon legacy software vendors** and get a fresh start as part of migration. For example, a business might choose to implement the repurchasing strategy by migrating from a customer relationship management (CRM) system to Salesforce.com. The total upfront expense of the step therefore goes up, but the potential benefits could be substantial.

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

Retiring is the process of removing applications that are no longer needed. Some parts of your enterprise IT portfolio are just no longer needed. We found as much as 10% to 20% of companies' application portfolios include applications that are no longer being used or already have replacements live and functional. Using the AWS migration plan as the opportunity to actually end-of-life these applications can save significant cost and effort for your team. Sometimes you just have to turn off the lights.

AWS Snow Family members

Some of our customers need to get data to AWS and most of them would like to do it in an efficient and timely manner. The usual route is to simply copy the required data over the internet or better yet, if they have a Direct Connect line. However, with the **limitations of bandwidth**, in general, this can take days, weeks, or even months.

For example, a dedicated one gigabyte per second network connection theoretically moves one petabyte of data in about 100 days and in the real world likely longer and at a higher cost.

The **AWS Snow Family** is a collection of **physical devices** that help to physically transport up to exabytes of data into and out of AWS.

AWS Snow Family is composed of **AWS Snowcone**, **AWS Snowball**, and **AWS Snowmobile**.

These devices offer different capacity points, and most **include built-in computing capabilities.** AWS owns and manages the Snow Family devices and integrates with AWS security, monitoring, storage management, and computing capabilities.

It should be noted that all Snow Family devices are designed to be secure and tamper-resistant while on-site or in transit. This means the hardware and software is cryptographically signed, and all data stored is **automatically encrypted** using 256-bit

encryption keys, owned and managed by you, the customer. You can even use **AWS Key Management Service** to generate and manage keys.

AWS Snowcone is a small, rugged, and secure **edge computing** and **data transfer device**. It features 2 CPUs, 4 GB of memory, and **8 TB** of usable storage. Edge computing options are Amazon EC2 instances and AWS IoT Greengrass. Customers usually use these devices to ship terabytes of information such as analytics data, video libraries, image collections, backups, and even tape replacement data.

AWS Snowball offers two types of devices:

- **Snowball Edge Storage Optimized** devices are well suited for large-scale data migrations and recurring transfer workflows, in addition to local computing with higher capacity needs.
    - Storage: **80 TB** of hard disk drive (HDD) capacity for block volumes and Amazon S3 compatible object storage, and 1 TB of SATA solid state drive (SSD) for block volumes.
    - Compute: 40 vCPUs, and 80 GiB of memory to support Amazon EC2 sbe1 instances (equivalent to C5).

- **Snowball Edge Compute Optimized** provides **powerful computing resources** for use cases such as machine learning, full motion video analysis, analytics, and local computing stacks.
    - Storage: **42-TB** usable HDD capacity for Amazon S3 compatible object storage or Amazon EBS compatible block volumes and 7.68 TB of usable NVMe SSD capacity for Amazon EBS compatible block volumes.
    - Compute: 52 vCPUs, **208 GiB of memory**, and an optional NVIDIA Tesla V100 **GPU**. Devices run Amazon EC2 sbe-c and sbe-g instances, which are equivalent to C5, M5a, G3, and P3 instances.

Once you plug them into your infrastructure, you can even run AWS Lambda functions, Amazon EC2-compatible AMI's, or even AWS IoT Greengrass to perform simple processing of data right then and there. Customers usually ship these to remote locations, where it's trickier to have a lot of computing power lying around. The use cases include capturing of streams from IoT devices, image compression, video transcoding, and even industrial signalling.

AWS Snowmobile is an **exabyte**-scale data transfer service used to move large amounts of data to AWS. You can transfer up to **100 petabytes** of data per Snowmobile, a 45-foot long ruggedized shipping container, pulled by a semi-trailer truck. It is tamper resistant, waterproof, temperature controlled, it even has fire suppression and GPS tracking. I mean, can you believe that it also has 24/7 video surveillance with a dedicated security team and escort security vehicle during transit? That's some serious business.

Innovation with AWS Services

There's so much more that can be done on AWS that we have time to talk about here. For example, when it comes to migrating onto AWS, we didn't even cover running **VMware on AWS**. The same VMware based infrastructure that you use on-premise can in many cases, just be lifted up and dropped onto AWS via **VMware Cloud on AWS**. And this is just one of many concepts that make AWS a place where builders go to create and innovate at the pace of their ideas.

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. With Amazon SageMaker, you can quickly and easily begin working on machine learning projects. You do not need to follow the traditional process of manually bringing together separate tools and workflows.

You can use ML to analyse data, solve complex problems, and predict outcomes before they happen. Tools like Amazon **SageMaker** and Amazon Augmented AI, or Amazon **A2I**, provide a machine learning platform that any business can build upon without needing PhD level expertise in-house. Amazon **Augmented AI (Amazon A2I)** provides **built-in human review workflows** for common machine learning use cases, such as content moderation and text extraction from documents. With Amazon A2I, you can also create your own workflows for machine learning models built on Amazon SageMaker or any other tools.

Or perhaps, ready-to-go AI solutions like Amazon **Lex**, the heart of Alexa.

Or what about Amazon **Textract**. Extracting text and data from documents to make them more usable for your enterprise instead of them just being locked away in a repository.

Do you want to put machine learning literally into the hands of your developers? Why not try **AWS DeepRacer**? A chance for your developers to experiment with **reinforcement learning**. One of the newest branches of machine learning algorithms, all while having fun in a racing environment.

AWS offers brand new technologies in things like Internet of Things. Enabling connected devices to communicate all around the world. Speaking of communication around the world, have you ever wanted to have your own **satellite**? Too expensive to launch your own? Why not just use **AWS Ground Station** and only pay for the satellite time you actually need?

Summary

Over the last few videos, you started to learn about migration to AWS, as well as some interesting and innovative services you can use to either aid in your migration or to step your game up with AWS.

You learned about the **AWS Cloud Adoption Framework**. The Cloud Adoption Framework gives you guidance on who to loop into a cloud migration, what their roles are, and the sorts of things that they should be focused on. There's the **Business, People, and Governance Perspectives for nontechnical planning, and the Platform, Security, and Operations Perspectives for technical planning**.

We also talked about the **six R's of migration**. The R's represent different strategies on moving solutions to the cloud. They are **Rehost, Replatform, Repurchase, Refactor, Retire, and Retain.**

There was also the question of how to move massive amounts of data into AWS without going over the network. This is where you learned about **AWS Snowball and AWS Snowmobile,** which allow you to fill in a physical device with your data and have it shipped back to AWS who then uploads it for you. This is useful to sidestep any potential throughput issues, and it is also **more secure** than using high-speed internet.

Quiz

Which Perspective of the AWS Cloud Adoption Framework helps you structure the selection and implementation of permissions?

- Security Perspective

Which strategies are included in the six strategies for application migration? (Select TWO.)

- Retaining and Rehosting

What is the storage capacity of AWS Snowmobile?

- 100 PB

Which statement best describes Amazon Lex?

- A service that enables you to build conversational interfaces using voice and text

Additional resources

To learn more about the concepts that were explored in Module 9, review these resources.

- Migration & Transfer on AWS
- A Process for Mass Migrations to the Cloud
- 6 Strategies for Migrating Applications to the Cloud
- AWS Cloud Adoption Framework
- AWS Fundamentals: Core Concepts
- AWS Cloud Enterprise Strategy Blog
- Modernizing with AWS Blog
- AWS Customer Stories: Data Centre Migration

Module 10 – The Cloud Journey

Contents

Learning objectives 1
Introduction 1
The AWS Well-Architected Framework 3
Advantages of cloud computing 5
Summary 6
Quiz 7
Additional resources 7

Learning objectives

In this module, you will learn how to:

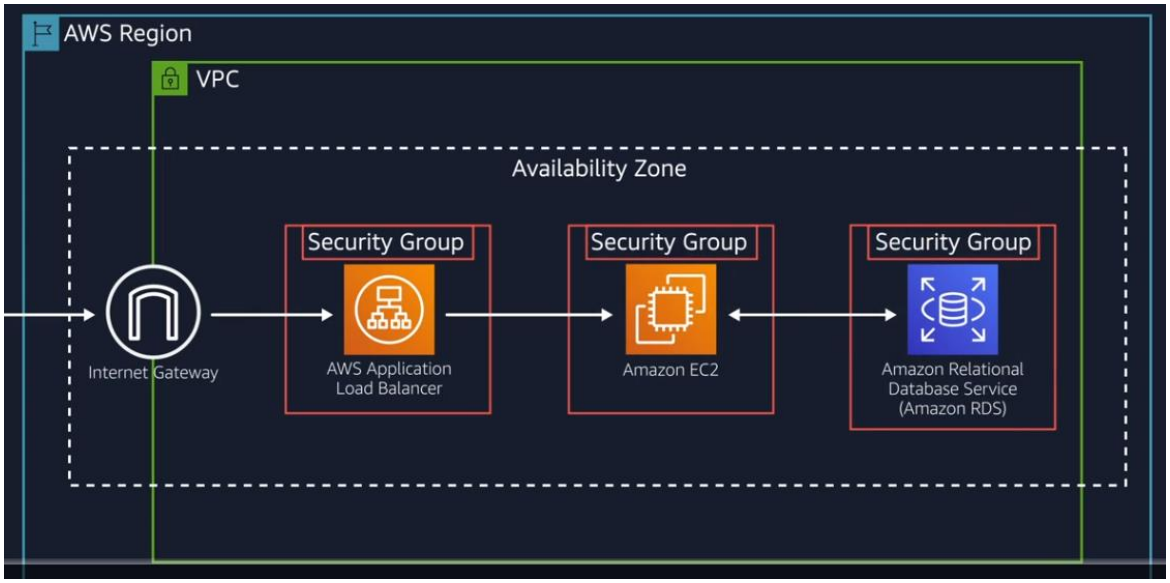
- Summarize the five pillars of the Well-Architected Framework.
- Explain the six benefits of cloud computing.

Introduction

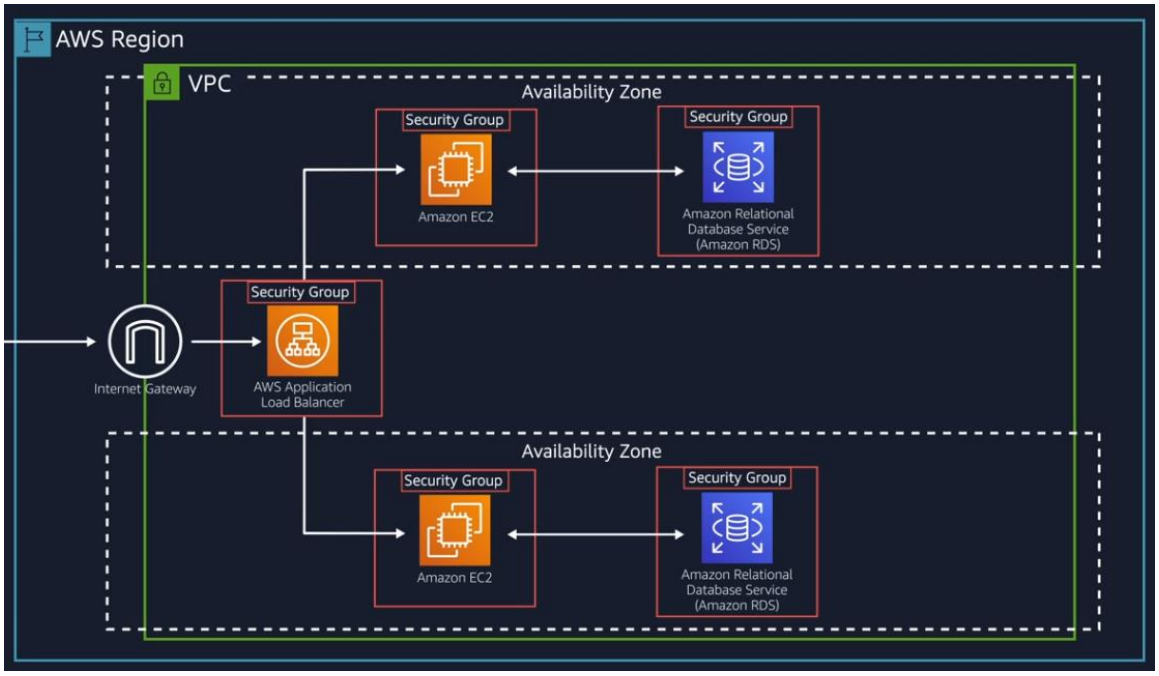
We've covered a lot of different AWS services in this course. And as you know, you use each individual service as building blocks for your solutions. There are endless architectures you can create to solve whatever problem you are trying to solve on AWS.

You can string together services in a simple or complicated manner. Having a lot of options is great, but how do you know if the architecture you've created is, well, good?

Let's look at this basic three-tier architecture. Does this look good? Well, we have a load balancer, some instances, and a backend database. Seems all right to me.



What about this alternate architecture?



This architecture has now been replicated across Availability Zones or AZs. This is **important for reliability**. If one AZ is having issues, your application will still be up and running in the second AZ. It's important that you are able to spot deficiencies in architectures like in this simple example.

Though not all examples are quite as simple. Luckily, there are tools that can help you get there. We're going to cover something called the **Well-Architected Framework**. This is a tool you can use to **evaluate the architectures you build for excellence** in a few different categories.

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.

The Well-Architected Framework is designed to enable architects, developers, and users of AWS to build secure, high-performing, resilient, and efficient infrastructure for their applications. It is based on **five** pillars:

- Operational excellence
- Security
- Reliability
- Performance efficiency
- Cost optimization

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.**

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 is the ability of a system to do the following:

- Recover from infrastructure or service disruptions (e.g. Amazon DynamoDB disruption, or EC2 node failure)
- 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 is the ability to **use computing resources efficiently to meet system requirements** and to maintain that efficiency as demand changes and technologies evolve.

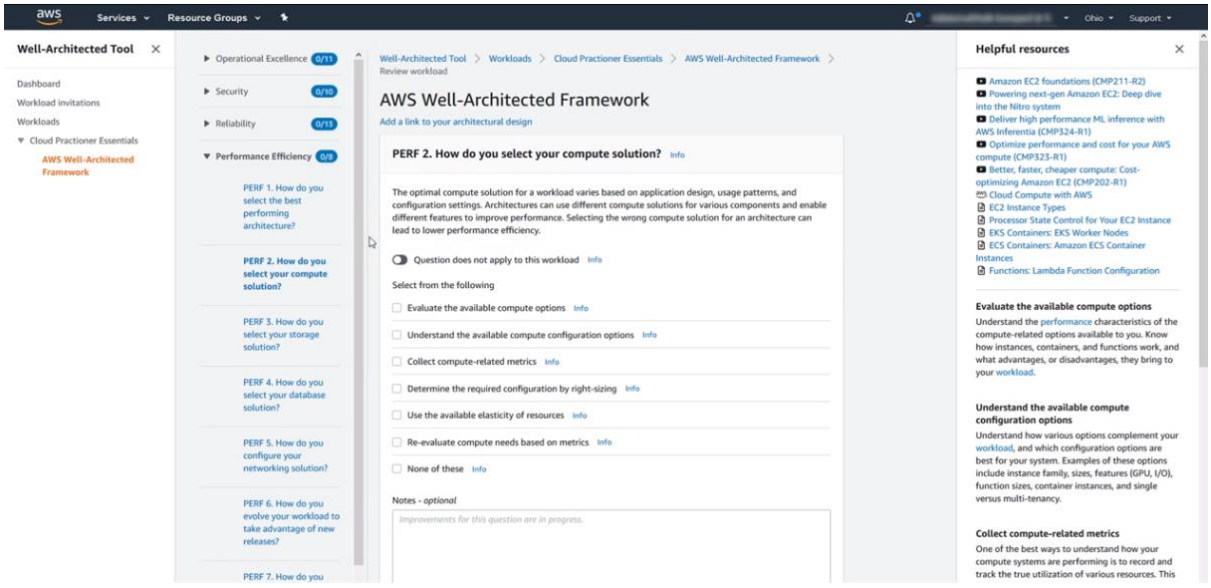
Evaluating the performance efficiency of your architecture includes **experimenting** more often, using **serverless** architectures, and designing systems to be able to go **global** in minutes.

Cost optimization is the ability to run systems to deliver business value at the **lowest price point.**

Cost optimization includes adopting a consumption model**, analysing and attributing expenditure**, and using **managed services to reduce the cost of ownership**.

In the past, you'd need to evaluate these against your AWS infrastructure with the help of a Solutions Architect. Not that you can't, and aren't still encouraged to do that, but we listened to customer feedback, and decided to release the **Framework as a self-service tool**, the **AWS Well-Architected Tool**.

You can access it by the AWS **Management Console**. Create a workload and run it against your AWS account.



Advantages of cloud computing

Operating in the AWS Cloud offers many benefits over computing in on-premises or hybrid environments.

In this section, you will learn about **six advantages of cloud computing**:

- Trade upfront expense for variable expense.
- Benefit from massive economies of scale.
- Stop guessing capacity.
- Increase speed and agility.
- Stop spending money running and maintaining data centres.
- Go global in minutes.

Trade upfront expense for variable expense.
Upfront expenses include data centres, physical servers, and other resources that you would need to invest in before using computing resources. These on-premises data centre costs include things like physical space, hardware, staff for racking and stacking, and overhead for running the data centre.

Instead of investing heavily in data centres and servers before you know how you're going to use them, you can pay only when you consume computing resources.

Benefit from massive economies of scale.
By using cloud computing, you can achieve a lower variable cost than you can get on your own.

Because usage from hundreds of thousands of customers aggregates in the cloud, providers such as AWS can achieve **higher economies of scale**. Economies of scale translate into lower pay-as-you-go prices.

AWS is also an expert at building efficient data centres. We can buy the hardware at a lower price because of the massive volume, and then we install it and run it efficiently. Because of these factors, you can achieve a lower variable cost than you could running a data centre on your own.

Stop guessing capacity.
With cloud computing, you don't have to predict how much infrastructure capacity you will need before deploying an application. You provision the resources you need **for the now**, and you **scale up and down accordingly**.

For example, you can launch Amazon Elastic Compute Cloud (Amazon EC2) instances when needed and pay only for the compute time you use. Instead of paying for resources that are unused or dealing with limited capacity, you can access only the capacity that you need, and scale in or out in response to demand.

Increase speed and agility.
The flexibility of cloud computing makes it easier for you to develop and deploy applications. This flexibility also provides your development teams with more time to experiment and innovate.

With AWS, it's easy to try new things. You can spin up test environments and run experiments on new ways to approach solving a problem. And then if that approach didn't work, you can just delete those resources and stop incurring cost. Traditional data centres don't offer the same flexibility.

Stop spending money running and maintaining data centres.
Cloud computing in data centres often requires you to spend more money and time managing infrastructure and servers. A benefit of cloud computing is the ability to focus less on these tasks and more on your applications and customers.

If you aren't a data centre company, why spend so much money and time running data centres? Let AWS take the undifferentiated heavy lifting off your hands and instead focus on what makes your business valuable.

Go global in minutes.
The AWS Cloud global footprint enables you to quickly deploy applications to customers around the world, while providing them with low latency. Traditionally, you would need to have staff overseas running and operating a data centre for you. With AWS, you can just replicate your architecture to a region in that foreign country.

Summary

In Module 10, you learned about the following concepts:

- The **five pillars** of the AWS Well-Architected Framework:
  - Operational excellence
  - Security
  - Reliability
  - Performance efficiency
  - Cost optimization

- **Six advantages** of cloud computing:
  - Trade upfront expense for variable expense.
  - Benefit from massive economies of scale.
  - Stop guessing capacity.
  - Increase speed and agility.
  - Stop spending money running and maintaining data centres.
  - Go global in minutes.

Quiz

Which pillar of the AWS Well-Architected Framework includes the ability to run workloads effectively and gain insights into their operations?

- Operational excellence

What are the benefits of cloud computing? (Select TWO.)

- Increase speed and agility
- Stop spending money running and maintaining data centres

Additional resources

To learn more about the concepts that were explored in Module 10, review these resources.

- AWS Well-Architected
- Whitepaper: AWS Well-Architected Framework
- AWS Architecture Centre
- Six Advantages of Cloud Computing
- AWS Architecture Blog

Module 11 – AWS Certified Cloud Practitioner Basics

Contents

Learning objectives 1
Exam domains 1
Recommended experience 2
Exam details 2
Whitepapers and resources 3
Exam strategies 3

Learning objectives

In this module, you will learn how to:

- Determine resources for preparing for the AWS Certified Cloud Practitioner exam.
- Describe the benefits of becoming AWS Certified.

Exam domains

The AWS Certified Cloud Practitioner exam includes four domains:

1. Cloud Concepts
2. Security and Compliance
3. Technology
4. Billing and Pricing

The areas covered describe each domain in the Exam Guide for the AWS Certified Cloud Practitioner certification. For a description of each domain, review the AWS Certified Cloud Practitioner website. You are encouraged to read the information in the Exam Guide as part of your preparation for the exam.

Each domain in the exam is weighted. The weight represents the percentage of questions in the exam that correspond to that particular domain. These are approximations, so the questions on your exam may not match these percentages exactly. The exam does not indicate the domain associated with a question. In fact, some questions can potentially fall under multiple domains.

Domain	Percentage of exam
Domain 1: Cloud Concepts	26%
Domain 2: Security and Compliance	25%
Domain 3: Technology	33%
Domain 4: Billing and Pricing	16%
Total	100%

You are encouraged to use these benchmarks to help you determine how to allocate your time studying for the exam.

Recommended experience

Candidates for the AWS Certified Cloud Practitioner exam should have a basic understanding of IT services and their uses in the AWS Cloud platform.

We recommend that you have at least six months of experience with the AWS Cloud in any role, including project managers, IT managers, sales managers, decision makers, and marketers. These roles are in addition to those working in finance, procurement, and legal departments.

Exam details

The AWS Certified Cloud Practitioner exam consists of **65 questions** to be completed in **90 minutes**. The minimum passing score is **70%**.

Two types of questions are included on the exam: multiple choice and multiple response.

- A **multiple-choice** question has one correct response and three incorrect responses, or distractors.
- A **multiple-response** question has two or more correct responses out of five or more options.

On the exam, there is no penalty for guessing. Any questions that you do not answer are scored as incorrect. If you are not sure of what the correct answer is, it's always best for you to guess instead of leaving any questions unanswered.

The exam enables you to flag any questions that you'd like to review before submitting the exam. This helps you to use your time during the exam efficiently, knowing that you can always go back and review any questions that you were initially unsure of.

Whitepapers and resources

As part of your preparation for the AWS Certified Cloud Practitioner exam, we recommend that you review the following whitepapers and resources:

- Overview of Amazon Web Services
- How AWS Pricing Works
- Compare AWS Support Plans

Exam strategies

This section explores several strategies that can help you to increase the probability of passing the exam.

Read the full question.
First, make sure that you read each question in full. Key words or phrases in the question that, if left unread, could result in you selecting an incorrect response option.

Predict the answer before reviewing the response options.
Next, try to predict the correct answer before looking at any of the response options.

This strategy helps you to draw directly from your knowledge and skills without distraction from incorrect response options. If your prediction turns out to be one of the response options, this can be helpful for knowing whether you're on the right track. However, make sure that you review all the other response options for that question.

Eliminate incorrect response options.
Before selecting your response to a question, eliminate any options that you believe to be incorrect.

This strategy helps you to focus on the correct option (or options, for multiple-response questions) and ensure that you have fulfilled all the requirements of the question.

E-LEARNING

AWS Cloud Practitioner Essentials

Module 12 – Final Assessment

Question

01/30

You are running an Amazon EC2 instance and want to store data in an attached resource. Your data is temporary and will not be kept long term. Which resource should you use?

- ✗ Amazon S3 bucket
- ✗ Amazon Elastic Block Store (Amazon EBS) volume
- ✗ Subnet
- ✓ Instance store

Question

02/30

Which service enables you to build the workflows that are required for human review of machine learning predictions?

- ✗ Amazon Lex
- ✗ Amazon Textract
- ✓ Amazon Augmented AI
- ✗ Amazon Aurora

Question

03/30

Which pillar of the AWS Well-Architected Framework focuses on using computing resources in ways that meet system requirements?

- ✓ Performance Efficiency
- ✗ Reliability
- ✗ Security
- ✗ Operational Excellence

Question

04/30

Which statement best describes Amazon GuardDuty?

- ✓ A service that provides intelligent threat detection for your AWS infrastructure and resources
- ✗ A service that helps protect your applications against distributed denial-of-service (DDoS) attacks
- ✗ A service that checks applications for security vulnerabilities and deviations from security best practices
- ✗ A service that lets you monitor network requests that come into your web applications

Question

05/30

Which tool is used to automate actions for AWS services and applications through scripts?

- ✗ Amazon Redshift
- ✓ AWS Command Line Interface
- ✗ Amazon QLDB
- ✗ AWS Snowball

Question

06/30

Which service is used to quickly deploy and scale applications on AWS?

- ✗ Amazon CloudFront
- ✓ AWS Elastic Beanstalk
- ✗ AWS Outposts
- ✗ AWS Snowball

Question

07/30

Which statement is TRUE for AWS Lambda?

- [x] You pay only for compute time while your code is running.
- [ ] The first step in using AWS Lambda is provisioning a server.
- [ ] To use AWS Lambda, you must configure the servers that run your code.
- [ ] Before using AWS Lambda, you must prepay for your estimated compute time.

Question

08/30

You want to send and receive messages between distributed application components. Which service should you use?

- [ ] Amazon ElastiCache
- [x] Amazon Simple Queue Service (Amazon SQS)
- [ ] AWS Snowball
- [ ] Amazon Route 53

Question

09/30

You want to store data in a key-value database.

Which service should you use?

- ✗ Amazon DocumentDB
- ✗ Amazon Aurora
- ✗ Amazon RDS
- ✓ Amazon DynamoDB

Question

10/30

Which service is used to transfer up to 80 PB of data to AWS?

- ✗ Amazon Neptune
- ✗ AWS DeepRacer
- ✓ AWS Snowmobile
- ✗ Amazon CloudFront

Question

11/30

Which Support plans include access to all AWS Trusted Advisor checks? (Select TWO.)

- [x] Business
- [x] Enterprise
- [ ] Basic
- [ ] AWS Free Tier
- [ ] Developer

Question

12/30

Which AWS Trusted Advisor category includes checks for your service limits and overutilized instances?

- [x] Performance
- [ ] Security
- [ ] Cost Optimization
- [ ] Fault Tolerance

Question

13/30

You want Amazon S3 to monitor your objects' access patterns. Which storage class should you use?

- ✗ S3 Glacier
- ✗ S3 One Zone-IA
- ✗ S3 Standard-IA
- ✓ S3 Intelligent-Tiering

Question

14/30

Which action can you perform in Amazon CloudFront?

- ✗ Provision resources by using programming languages or a text file.
- ✓ Deliver content to customers through a global network of edge locations.
- ✗ Run infrastructure in a hybrid cloud approach.
- ✗ Provision an isolated section of the AWS Cloud to launch resources in a virtual network that you define.

Question

15/30

Which compute option reduces costs when you commit to a consistent amount of compute usage for a 1-year or 3-year term?

- ✗ Reserved Instances
- ✗ Spot Instances
- ✓ Savings Plans
- ✗ Dedicated Hosts

Question

16/30

Which service is used to run containerized applications on AWS?

- ✗ Amazon Redshift
- ✓ Amazon Elastic Kubernetes Service (Amazon EKS)
- ✗ Amazon Aurora
- ✗ Amazon SageMaker

Question

17/30

Which statement best describes an Availability Zone?

- ✗ The server from which Amazon CloudFront gets your files
- ✓ A fully isolated portion of the AWS global infrastructure
- ✗ A separate geographical location with multiple locations that are isolated from each other
- ✗ A site that Amazon CloudFront uses to cache copies of content for faster delivery to users at any location

Question

18/30

Which virtual private cloud (VPC) component controls inbound and outbound traffic for Amazon EC2 instances?

- ✓ Security group
- ✗ Subnet
- ✗ Internet gateway
- ✗ Network access control list

Question

19/30

Which tool enables you to visualize, understand, and manage your AWS costs and usage over time?

- ✗ AWS Pricing Calculator
- ✗ AWS Artifact
- ✓ AWS Cost Explorer
- ✗ AWS Budgets

Question

20/30

Which actions can you perform in Amazon Route 53? (Select TWO.)

- ✗ Access AWS security and compliance reports and select online agreements.
- ✗ Automate the deployment of workloads into your AWS environment.
- ✗ Monitor your applications and respond to system-wide performance changes.
- ✓ Manage DNS records for domain names.
- ✓ Connect user requests to infrastructure in AWS and outside of AWS.

Question

21/30

Which migration strategy involves changing how an application is architected and developed, typically by using cloud-native features?

- ✗ Rehosting
- ✗ Repurchasing
- ✗ Replatforming
- ✓ Refactoring

Question

22/30

Which statement best describes Elastic Load Balancing?

- ✗ A service that provides data that you can use to monitor your applications, optimize resource utilization, and respond to system-wide performance changes
- ✗ A service that monitors your applications and automatically adds or removes capacity from your resource groups in response to changing demand
- ✓ A service that distributes incoming traffic across multiple targets, such as Amazon EC2 instances
- ✗ A service that enables you to set up, manage, and scale a distributed in-memory or cache environment in the cloud

Question

23/30

You want to store data in a volume that is attached to an Amazon EC2 instance. Which service should you use?

- ✓ Amazon Elastic Block Store (Amazon EBS)
- ✗ AWS Lambda
- ✗ Amazon ElastiCache
- ✗ Amazon Simple Storage Service (Amazon S3)

Question

24/30

Which component or service enables you to establish a dedicated private connection between your data center and virtual private cloud (VPC)?

- ✗ Internet gateway
- ✗ Amazon CloudFront
- ✗ Virtual private gateway
- ✓ AWS Direct Connect

Question

25/30

Which Perspective of the AWS Cloud Adoption Framework focuses on recovering IT workloads to meet the requirements of your business stakeholders?

- ✗ Business Perspective
- ✗ Governance Perspective
- ✓ Operations Perspective
- ✗ People Perspective

Question

26/30

Which statement best describes AWS Marketplace?

- ✓ A digital catalog that includes thousands of software listings from independent software vendors
- ✗ A resource that can answer questions about best practices and assist with troubleshooting issues
- ✗ A resource that provides guidance, architectural reviews, and ongoing communication with your company as you plan, deploy, and optimize your applications
- ✗ An online tool that inspects your AWS environment and provides real-time guidance in accordance with AWS best practices

Question

27/30

Which tasks are the responsibilities of AWS? (Select TWO.)

- [ ] Configuring security groups on Amazon EC2 instances
- [ ] Creating IAM users and groups
- [x] Configuring AWS infrastructure devices
- [ ] Training company employees on how to use AWS services
- [x] Maintaining virtualization infrastructure

Question

28/30

In the S3 Intelligent-Tiering storage class, Amazon S3 moves objects between a frequent access tier and an infrequent access tier. Which storage classes are used for these tiers? (Select TWO.)

- [ ] S3 Glacier Deep Archive
- [ ] S3 Glacier
- [ ] S3 One Zone-IA
- [x] S3 Standard-IA
- [x] S3 Standard

Question

29/30

Which service enables you to review details for user activities and API calls that have occurred within your AWS environment?

- Amazon CloudWatch
- AWS CloudTrail
- AWS Trusted Advisor
- Amazon Inspector

Question

30/30

Which service enables you to consolidate and manage multiple AWS accounts from a central location?

- AWS Organizations
- AWS Identity and Access Management (IAM)
- AWS Artifact
- AWS Key Management Service (AWS KMS)