AWS Cloud Practitioner Essentials

Module 5 introduction

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.

Video transcript

Well, we have quite a coffee operation going now. We've got customers, lots of happy customers. In fact, we have an elastic, scalable, disaster resistant, cost optimized architecture that now has a global, highly secured network that can be deployed entirely programmatically.

You know, now we need some way to show our appreciation to all our loyal customers. How about a frequent drinker loyalty program? Or we could just hand out punch cards, but let's be honest, we can't track those well and use them to get to know our customers better.

We need digital cards, which means we need to be able to keep track of our customers, what they order, how much they purchased. This will help them, the customers, get the best rewards they've earned and help us to know our customer base better.

This means we need databases. Databases, and storage. And not just any database. We need to make sure we choose the right database for the task and the right storage for data types.

Now you may have many different data usage needs and data types. Let's learn about the different services AWS has to help you build the perfect data solution.

Instance stores and Amazon Elastic Block Store (Amazon EBS):

Video Transcript

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 instances have hard drives as well. And there are a few different types.

When you launch an EC2 instance, depending on the type of the EC2 instance you launched, it might provide you with local storage called instance store volumes. These volumes are physically attached to the host, your EC2 instances running on top of. And you can write to it just like a normal hard drive. 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. 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.

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.

All right, I'm telling you not to write important data to the drives that come with EC2 instances. I'm sure that sounds a bit scary because obviously you'll need a place to write data that persists outside of the life cycle of an EC2 instance. You don't want your entire database getting deleted every time you stop an EC2 instance. Don't worry, this is where a service called Amazon Elastic Block Store, or EBS, comes into play.

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 you're easy to is running on. This means, that the data that you write to an EBS volume can persists between stops and starts of an EC2 instance.

EBS volumes come in all different sizes and types. How this works, is you define the size, type and configurations of the volume you need. Provision the volume, and then attach it to your EC2 instance. From there, you can configure your application to write to the volume and you're good to go. If you stop and then start the EC2 instance, the data in the volume remains.

Since the use case for EBS volumes is to have a hard drive that is persistent, that your applications can write to, it's probably important that you back that data up. EBS allows you to take incremental backups of your data called 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.

Instance stores:

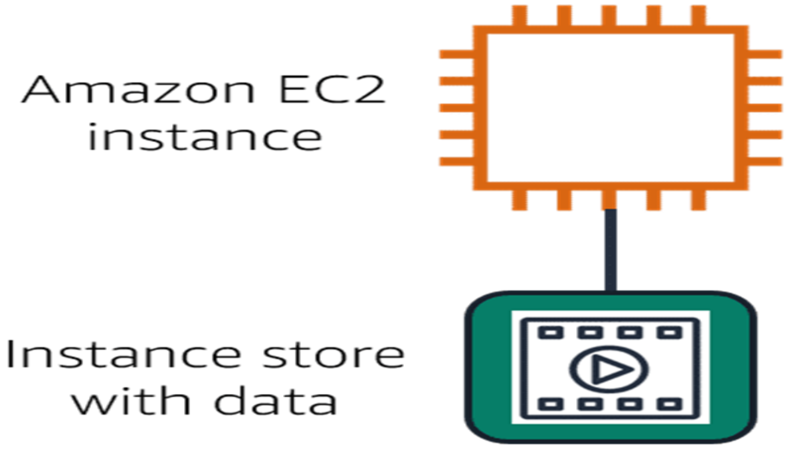
Block-level storage volumes behave like physical hard drives.

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

.

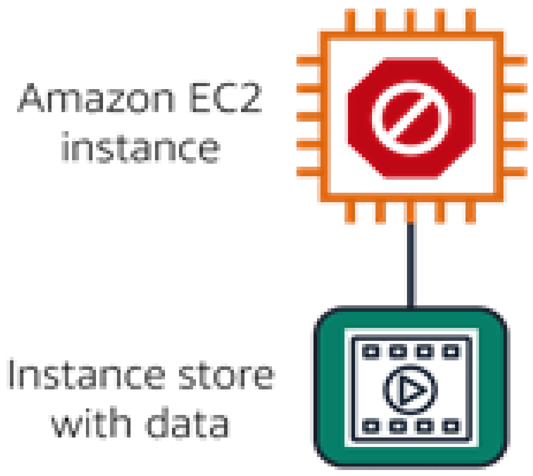
STEP 1:

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

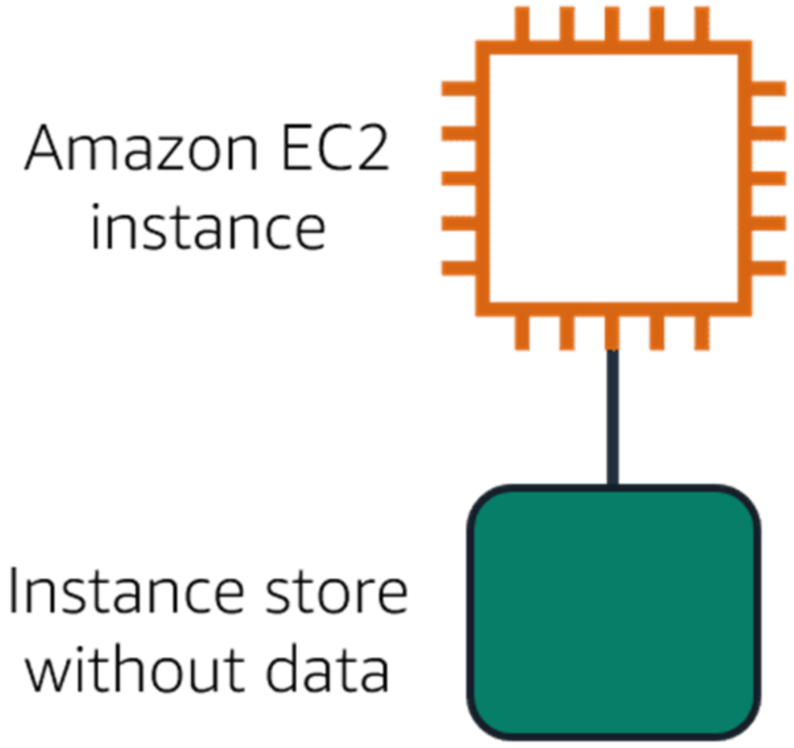


STEP 2:

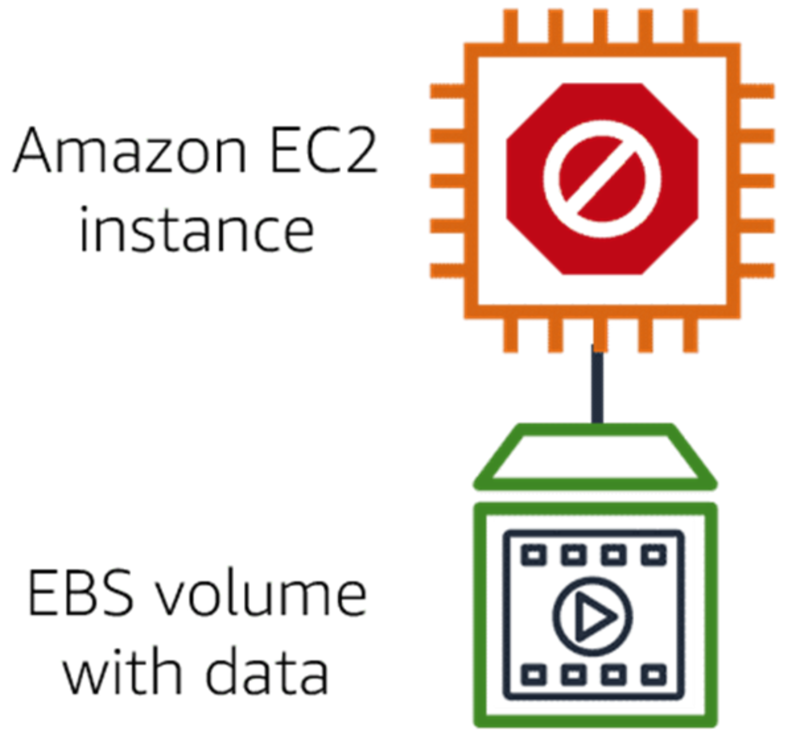
The instance is stopped or terminated.



STEP 3:
All data on the attached instance store is deleted.



Summary
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.

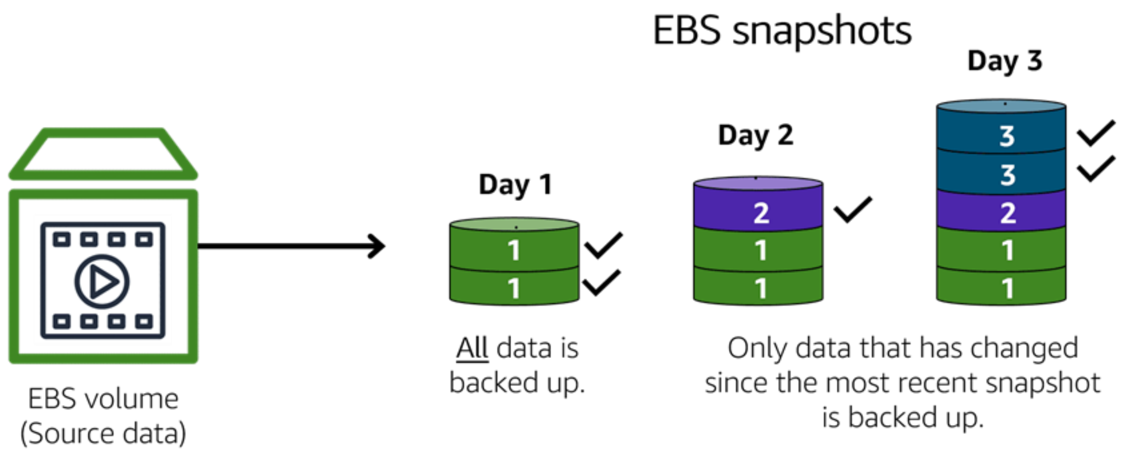


Amazon Elastic Block Store (Amazon EBS) is a service that provides block-level storage volumes that you can use with Amazon EC2 instances. If you stop or terminate an Amazon EC2 instance, all the data on the attached EBS volume remains available.

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

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

Amazon EBS snapshots



An **EBS snapshot** is an incremental backup. This means that the first backup taken of a volume copies all the data. For subsequent backups, only the blocks of data that have changed since the most recent snapshot are saved.
Incremental backups are different from full backups, in which all the data in a storage volume copies each time a backup occurs. The full backup includes data that has not changed since the most recent backup.

Amazon Simple Storage Service (Amazon S3):

Video transcript

Welcome to this video on Amazon Simple Storage Service, or S3. From the name, you've probably guessed that it is a storage service and it's, well, simple. Most businesses have data that needs to be stored somewhere. For the coffee shop, this could be receipts, images, 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, that is an object and think of a file directory, that is the bucket. The maximum object size that you can upload is five terabytes. You can also version objects to protect them from accidental deletion of an object. 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.

And you can even stage data between different tiers. These tiers offer mechanisms for different storage use cases such as data that needs to be accessed frequently, versus audit data that needs to be retained for several years. Let's go through the notable ones, shall we?

The first here is called S3 Standard and comes with 11 nines of durability. That means an object stored in S3 Standard has a 99.999999999 percentage – that's a lot of nines – 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. Pretty awesome way to start up that coffee blog.

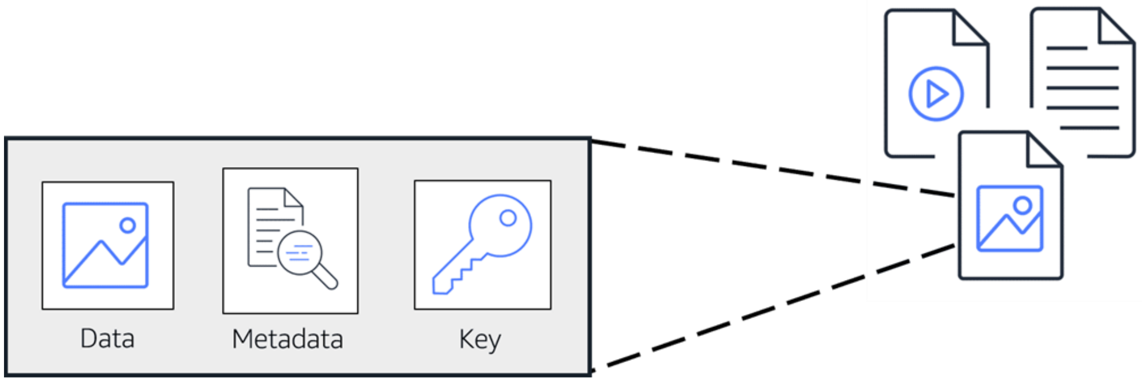
The next storage class is called S3 Infrequent Access or S3-IA. Which is used for data that is accessed less frequently but requires rapid access when needed. This means it's a perfect place to store backups, disaster recovery files, or any object that requires a long-term storage.

Another storage class or tier lends itself to that example we had earlier about audit data. 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. 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.

Fact, I don't think we mentioned lifecycle policies up till this point. But they 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. It's another example of a managed AWS service, helping take that burden off you, so you can focus on more of your business needs.

Just to note that there are other storage classes. Like S3 Infrequent Access One Zone and S3 Glacier Deep Archive that you can use. Happy storing.

Object storage



In **object storage**, each object consists of data, metadata, and a key.
The data might be an image, video, text document, or any other type of file. Metadata contains information about what the data is, how it is used, the object size, and so on. An object’s key is its unique identifier.
NOTE: Recall that when you modify a file in block storage, only the pieces that are changed are updated. When a file in object storage is modified, the entire object is updated.

Amazon Simple Storage Service (Amazon S3)

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

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

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

Amazon S3 storage classes:

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

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

To learn more about the Amazon S3 storage classes select the + symbol next to each category.

Amazon S3 Standard

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

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

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

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

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

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

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

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

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

Amazon S3 Intelligent-Tiering

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

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

Amazon S3 Glacier Instant Retrieval

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

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

Amazon S3 Glacier Flexible Retrieval

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

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

Amazon S3 Glacier Deep Archive

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

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

Amazon S3 Outposts

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

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

Knowledge Check:

You want to store data that is infrequently accessed but must be immediately available when needed.

Which Amazon S3 storage class should you use?

- Amazon S3 Intelligent-Tiering
- Amazon S3 Glacier Deep Archive
- Amazon S3 Standard-IA
- Amazon S3 Glacier Flexible Retrieval

The correct response option is **Amazon S3 Standard-IA**.

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

The other response options are incorrect because:

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

- Amazon S3 Glacier Flexible Retrieval and Amazon S3 Glacier Deep Archive are low-cost storage classes that are ideal for data archiving. They would not be the best choice for this scenario, which requires high availability. You can retrieve objects stored in the Amazon S3 Glacier Flexible Retrieval storage class within a few minutes to a few hours. By comparison, you can retrieve objects stored in the Amazon S3 Glacier Deep Archive storage class within 12 hours.

Comparing Amazon EBS and Amazon S3:

Video Transcript:

AWS Cloud Practitioners, welcome to the clash of the storage class! In the block storage corner, weighing in at sizes up to 16 tebibytes each, with a unique ability to survive the termination of their Amazon EC2 instances, they are solid state, they are spinning platters, they are Amazon Elastic Block Storage!

In the regional object storage corner, weighing in at unlimited storage, with individual objects at 5,000 gigabytes in size, they specialize in write once/read many, they are 99 .999 999 999% durable, they are Amazon Simple Storage Service!

Head-to-head, each storage class boasts the best dynamically distributed design for different storage demands. Which storage class will ultimately be victorious in this thunderdome slugfest? To understand who wins, you need to clarify a use case.

Round one. 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. It's 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!

Amazon Elastic File System (Amazon EFS):

Video Transcript:

Next up on the list of storage services is Amazon Elastic File System, or what we call EFS. EFS is a managed file system. 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. This data traditionally has been hosted on premises. In this on-premises data center, you would have to ensure that the storage you have can keep up with the amount of data that you are storing. Make sure backups are taken, and that the data is stored redundantly as well as manage all of the servers hosting that data.

Luckily with AWS, you don't need to worry about buying all of that hardware and keeping the whole file system running from an operational standpoint. With EFS, you can keep existing file systems in place but let AWS do all the heavy lifting of the scaling and the replication. 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. Super nice, right? Well, you might be thinking, Amazon EBS also lets me store files that I can access from EC2 instances. What exactly is the difference here?

[Blaine] AWS Cloud Practitioners, welcome back!

[Morgan] All right, we don't need to do all of that. The answer is really simple. Amazon EBS volumes attach to EC2 instances and are an Availability Zone-level resource. In order to attach EC2 to EBS, you need to be in the same AZ. 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 can have multiple instances reading and writing from it at the same time.

But it isn't just a blank hard drive that you can write to. It is a true file system for Linux. It is also a regional resource. Meaning any EC2 instance in the Region can write to the EFS file system. As you write more data to EFS, it automatically scales. No need to provision any more volumes.

File storage:

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

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

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

Comparing Amazon EBS and Amazon EFS

AMAZON EBS:
An Amazon EBS volume stores data in a **single** Availability Zone.
To attach an Amazon EC2 instance to an EBS volume, both the Amazon EC2 instance and the EBS volume must reside within the same Availability Zone.
AMAZON EFS:
Amazon EFS is a regional service. It stores data in and across **multiple** Availability Zones.
The duplicate storage enables you to access data concurrently from all the Availability Zones in the Region where a file system is located. Additionally, on-premises servers can access Amazon EFS using AWS Direct Connect

Amazon Relational Database Service (Amazon RDS):

Video Transcript:

So you're storing data about your coffee shop in various systems. But you're finding that you need to maintain relationships between various types of data. And by relationships, I mean, if say, a customer orders the same drink multiple times, maybe you want to offer them a promotional discount on their next purchase. And you need a way to keep track of this relationship somewhere. In this case, it's best to use a relational database management system, or RDBMS. Essentially, It means we store data in a way such that it relates to other pieces of data.

For example, if we had a customer entry or record, we store that in a customer table. We then could have an entry for the physical address, which we store on a corresponding address table. We then relate the two via a common attribute and can query the data that is housed in both tables.

The most common way to query the data is by writing queries in SQL. And this runs on a variety of database systems. Speaking of database systems, what are some of the more well known ones that AWS supports? Well, there's MySQL, PostgreSQL, Oracle, Microsoft SQL Server, and many more. If you have an on-premises environment, you're probably running one of those and they're most likely housed in your data center.

But is there a way to easily move them to the cloud? Well, the simple answer is yes, you can do what we call a Lift-and-Shift, and migrate your database to run on Amazon EC2. This means you have control over the same variables you do, in your on-premises environment, such as OS, memory, CPU, storage capacity, and so forth. It's a quick entry to the cloud, and you can migrate them using standard practices or using something like Database Migration Service, which we'll cover in a later video.

The other option for running your databases in the cloud is to use a more managed service called Amazon Relational Database Service, or RDS. It supports all the major database engines, like the ones we mentioned earlier, but this service comes with added benefits. These 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. Which if you're a database admin, can be pretty time consuming and difficult.

So how do we make it even easier for you to run database workloads on the cloud? Well, we go one further and have them migrate or deploy to Amazon Aurora. It's our 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. The other benefits are things like your data is replicated across facilities, so you have six copies at any given time. You can also deploy up to 15 read replicas, so you can offload your reads and scale performance. Additionally, there's continuous backups to S3, so you always have a backup ready to restore. You also get point in time recovery, so you can recover data from a specific period.

And there you have it, relational databases in a nutshell.

Relational databases

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

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

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

Example of data in a relational database:

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

Amazon Relational Database Service

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

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

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

Amazon RDS database engines

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

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

Amazon Aurora

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

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

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

Amazon DynamoDB:

Video Transcript:

Let's talk about Amazon DynamoDB. At its most basic level, DynamoDB is a database. It's a serverless database, meaning you don't need to manage the underlying instances or infrastructure powering it.

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. And you don't need to worry about the scaling of the system, up or down.

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.

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.

Now, DynamoDB isn't a normal database. In the sense that it doesn't use the very widely used query language, sequel, or SQL. Relational databases, like a standard MySQL Database, require that you have a well defined schema, in place. That might consist of one, or many tables that might relate to each other. You then use SQL to query the data.

This 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. DynamoDB is a non-relational database. Non-relational databases tend to have simple flexible schemas, not complex rigid schemas, laying out multiple tables that all relate to each other.

With DynamoDB, you can add and remove attributes from items in the table, at any time. Not every item in the table has to have the same attributes. This is great for datasets that have some variation from item to item. Because of this flexibility, you cannot run complex SQL queries on it. Instead, you would write queries based on a small subset of attributes that are designated as keys.

Because of this, the queries that you run are non-relational databases tend to be simpler, and focus on a collection of items from one table, not queries than span multiple tables. This query pattern, along with other factors, including the way that the underlying system is designed, allows DynamoDB to be very quick in response time, and highly scalable.

So, things to remember: DynamoDB is a non-relational, NoSQL database. It is purpose built. Meaning it has specific use cases, and it isn't the best fit for every workload out there. It has millisecond response time. It's fully managed, and it's highly scalable. One awesome example is on Prime Day in 2019, across the 48 hours of Prime Day, there were 7.11 trillion calls to the DynamoDB API, peaking at 45.4 million requests per second. That's insanely scalable, all without the underlying database management. That's pretty cool.

Nonrelational databases

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

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

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

Example of data in a nonrelational database:

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

Amazon DynamoDB

Amazon DynamoDB is a key-value database service. It delivers single-digit millisecond performance at any scale.
To learn about features of DynamoDB, select each flashcard to flip it.

Serverless:

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

Automatic Scaling:

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

Video Transcript:

AWS Cloud Practitioners, welcome back to the championship chase of the database! In the relational corner, engineered to remove undifferentiated heavy lifting from your database administrators with automatic high availability and recovery provided. You control the data, you control the schema, you control the network. You are running Amazon RDS. Yes, Yeah.

The NoSQL corner, using a key value pair that requires no advanced schema, able to operate as a global database at the touch of a button. It has massive throughput. It has petabyte scale potential. It has granular API access. It is Amazon DynamoDB.

Head to head. Each database class is engineered to exactly enhance exciting existential environments you envision. Which database will ultimately be victorious in this new world knock out night fight? Once again, the winner will depend on your use case.

Round one, 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 analyze 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.

Knowledge Check:

What are the scenarios in which you should use Amazon Relational Database Service (Amazon RDS)?

(Select TWO.)

- Running a serverless database
- Using SQL to organize data
- Storing data in a key-value database
- Scaling up to 10 trillion requests per day
- Storing data in an Amazon Aurora database

The two correct response options are:

- Using SQL to organize data
- Storing data in an Amazon Aurora database

The other three response options are scenarios in which you should use Amazon DynamoDB.

Amazon Redshift:

Video transcript

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.

Now there are many data warehouse solutions out on the market. If you already have a favorite 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. 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.

I'm not gonna go into details of how the magic works. 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.

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.

AWS Database Migration Service:

Video Transcript:

We've been talking about databases and the various database options on AWS. But what happens if you have a database that's on-premises or in the cloud already? Does that mean you have to start from scratch or does AWS have a magical way to help you migrate your existing database?

Thankfully, AWS offers a service called Amazon Database Magic. I mean, Amazon Database Migration Service, or DMS, to help customers do just that. DMS helps customers migrate existing databases onto AWS in a secure and easy fashion. You essentially migrate data between a source and a target database. The best part is that the source database remains fully operational during the migration, minimizing downtime to applications that rely on that database. Better yet is that the source and target databases don't have to be of the same type.

But let's start with databases that are of the same type. These migrations are known as homogenous and can be from MySQL to Amazon RDS for MySQL, Microsoft SQL Server to Amazon RDS for SQL Server or even Oracle to Amazon RDS for Oracle. The process is fairly straightforward since schema structures, data types, and database code is compatible between source and target.

As I mentioned, the source database can be located on-premises, running on Amazon EC2 Instances, or it can be an Amazon RDS database. The target itself can be a database in Amazon EC2 or Amazon RDS. In this case, you create a migration task with connections to the source and target databases. Then start the migration with a click of the button. AWS Database Migration Service takes care of the rest.

The second type of migration occurs when source and target databases are of different types. This is called heterogeneous migrations, and 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.

But these are not the only use cases for DMS. Others include development and test database migrations, database consolidation, and even continuous database replication.
Development and test migration is when you want to develop this to test against production data, but 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 is when you have several databases and want to consolidate them into one central database.

Finally, continuous replication is when you use DMS to perform continuous data replication. This could be for disaster recovery or because of geographic separation.

If you'd like to learn more about any of these, please check out our resources section. And there you have it, folks, DMS in a nutshell.

AWS Database Migration Service (AWS DMS):

AWS Database Migration Service (AWS DMS) enables you to migrate relational databases, nonrelational databases, and other types of data stores.
With AWS DMS, you move data between a source database and a target database. The source and target databases can be of the same type or different types. During the migration, your source database remains operational, reducing downtime for any applications that rely on the database.
For example, suppose that you have a MySQL database that is stored on premises in an Amazon EC2 instance or in Amazon RDS. Consider the MySQL database to be your source database. Using AWS DMS, you could migrate your data to a target database, such as an Amazon Aurora database.

Other use cases for AWS DMS

Development and test database migrations: Enabling developers to test applications against production data without affecting production users
Database consolidation: Combining several databases into a single database
Continuous replication: Sending ongoing copies of your data to other target sources instead of doing a one-time migration

Additional database services:

Video Transcript:

Before we wrap up databases and storage, I wanna loop back to the topic that we started all this with. Choosing the right database, choosing the right storage platform to fit your business needs, rather than forcing your data to fit your database's requirements. No matter what a database vendor might try to tell you, there is no one-size-fits-all database for all purposes. We've covered quite a few database flavors already, but there are even more databases AWS offers for special business requirements, that we don't have time to cover. But its worth just knowing they are there in case you need them.

For example, we talked about DynamoDB, and that's great for key-value pair databases. But what if you need more than just small attributes? What if you need a full content management system? Introducing Amazon DocumentDB, Great for content management, catalogs, user profiles.

What if you add a social network that you wanted to track for those kind of social webs, who is connected to who, is very clunky to manage in a traditional relational database so Amazon Neptune: a graph database, engineered for social networking and recommendation engines, also great for fraud detection needs.

Or perhaps you have a supply chain, that you have to track with assurances that nothing is lost. Or you have banking or financial records that require 100% immutability, or some people will tell you, oh, that's what blockchain is all about. Well, perhaps, I mean now, If you do need a blockchain solution, wouldn't you know it? We offer Amazon Managed Blockchain. But that's probably not what you really need here. It solves part of the equation, but adds a huge decentralization component, that's not what financial regulators wanna see. What you really need is an immutable ledger, so Amazon QLDB, or Quantum Ledger Database. An immutable system of record where any entry can never be removed from the audits.

Databases by themselves are great but if there is a way to make them faster, wouldn't that be greater? But you know I wouldn't be saying that, if there weren't some accelerator options that can be used in a number of unique scenarios. Starting with adding caching layers on top of your databases that can help improve read times of common requests from milliseconds to microseconds Amazon ElastiCache can provide those caching layers without your team needing to worry about the heavy lifting of launching, uplift, and maintenance. And it comes in both Memcached and Redis flavors.

Or if you are using DynamoDB, try using DAX, the DynamoDB Accelerator, a native caching layer designed to dramatically improve read times for your nonrelational data.

The key thing to understand is, AWS wants to make sure that you using the best tool for the job.

Additional database services

Amazon DocumentDB

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

Amazon Neptune

Amazon Neptune is a graph database service.

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

Amazon Quantum Ledger Database (Amazon QLDB)
Amazon Quantum Ledger Database (Amazon QLDB) is a ledger database service.
You can use Amazon QLDB to review a complete history of all the changes that have been made to your application data.
Amazon Managed Blockchain
Amazon Managed Blockchain is a service that you can use to create and manage blockchain networks with open-source frameworks.
Blockchain is a distributed ledger system that lets multiple parties run transactions and share data without a central authority.
Amazon ElastiCache
Amazon ElastiCache is a service that adds caching layers on top of your databases to help improve the read times of common requests.
It supports two types of data stores: Redis and Memcached.
Amazon DynamoDB Accelerator
Amazon DynamoDB Accelerator (DAX) is an in-memory cache for DynamoDB.

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

Module 5 summary:

In Module 5, you learned about the following concepts:

- Amazon EC2 instance store and Amazon EBS
- Amazon S3
- Amazon EFS
- Relational databases and Amazon RDS
- Nonrelational databases and DynamoDB
- Amazon Redshift
- AWS DMS
- Additional database services and accelerators

Video Transcript:

Another module completed. Awesome stuff. 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 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. 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 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.

Module 6 introduction

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 the 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.

Video Transcript:

Looks like we're getting deeper into our AWS account. Things are running well. And I want to let you in on the security measures we have in place. By this, I mean, we have to describe the various security mechanisms we offer on the AWS Cloud, like the shared responsibility model.

With the shared responsibility model, AWS controls security of the cloud and customers control security in the cloud. We, as AWS, control the data centers, security of our services, and all the layers in this section. The next part are the workloads that AWS customers run in the cloud and those are the customer's responsibility to secure. It's something we share with customers to ensure security in the cloud.

Let's take a look at the various other security services, mechanisms, and features that AWS has to offer in this module. Stay tuned.

Shared responsibility model

Video Transcript:

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

Now, if there's any security experts watching this right now you're probably shaking your head saying you can't have two different entities with the ultimate responsibility over a single object. That's not security, that's wishful thinking. At AWS, we agree completely. But for us, we don't look at your environment as a single object. Instead we see it as a collection of parts that build on each other. AWS is responsible for the security of some of the objects. Responsible 100% for those. The others, you are responsible 100% for their security. This is what's known as the shared responsibility model.

It's no different than securing a house. The builder constructed the house with four walls and a door. It's their responsibility to make sure the walls are strong and the doors are solid. It's your responsibility, the homeowner, to close and lock the doors.

It really is that simple on AWS as well. Take EC2, for instance. EC2 lives in a physical building, a data center that must be secured. It has a network and a hypervisor that supports your instances and their individual operating systems. On top of that operating system, you have your application, and that supports your data. So for EC2 and every service AWS offers, there's a similar stack of parts that build on top of each other. AWS is 100% responsible for some, you are responsible for the others.

So, starting with the physical layer. This is iron and concrete and fences and security guards. Someone has to own the concrete. Someone has to staff the physical perimeter, 24/7. This is AWS. On top of the physical layer we have our network and our hypervisor. Now, I'm not gonna go into details on how this is all secured, but basically we have reinvented those technologies to make them faster, stronger, tamper-proof.

But you don't have to just take our word for it. We have numerous third party auditors who have gone through the code and the way we build our infrastructure, and can provide the right documentation you need for your security compliance structures. Now, on top of all that, on EC2, you now get to pick what operating system you want to run.

This is the magic dividing line that separates our responsibility. AWS' responsibility and your responsibility. 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. I mean, no more than a construction company would keep copies of your front door key, AWS cannot enter your operating system. And here's a hint. If someone from AWS calls and asks you for your OS key, it is not AWS.

Now that means your operations team is 100% responsible for keeping the operating system patched. If AWS discovers there are some new vulnerabilities in your version of Windows, let's say, we can certainly notify your account owner but we cannot deploy a patch. This is a really good thing for your security. This means no one can deploy anything that might break your system without your team being the ones that do it. Now, on top of that OS, you can run whatever applications you want. You own them. You maintain them.

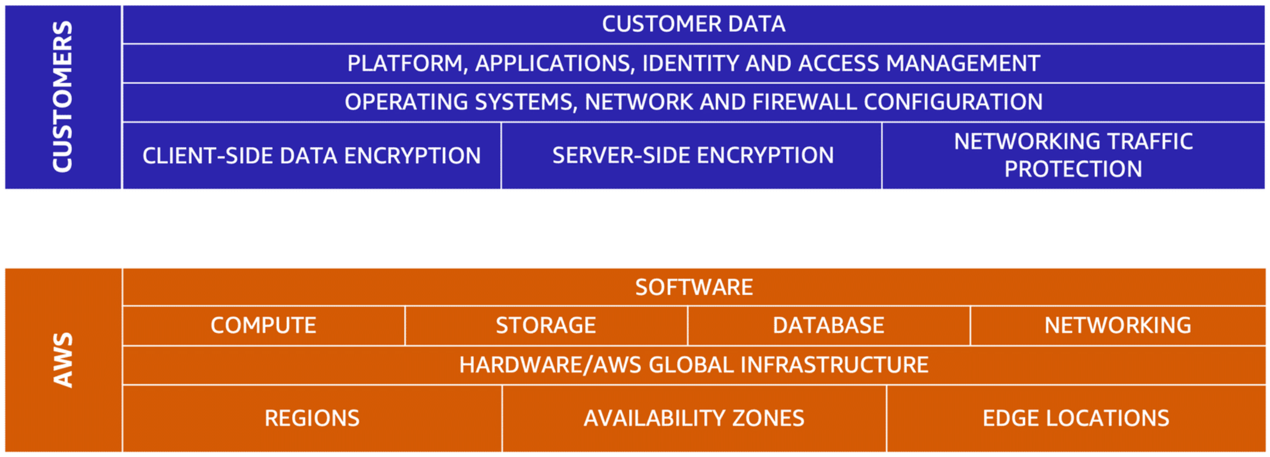
Which takes us to the most important part of the stack, your data. Data. This is always your domain to control. And sometimes you might want to have your data open for everyone to see, like pictures on a retail website. Other times like banking or healthcare, yeah, not so much, not so much. AWS provides everyone with the tool set they need for their data to open it up to some authorized individuals, to everyone, to just a single person under specific conditions, or even lock it down so no one can access it. Plus, the ability to have ubiquitous encryption. That way even if you accidentally left your front door open, all anyone would see is unreadable encrypted content.

The AWS shared responsibility model is about making sure both sides understand exactly what tasks are ours. Basically, AWS is responsible for the security of the cloud and you are responsible for the security in the cloud. Together, you have an environment you can trust.

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").


CUSTOMERS
CUSTOMER DATA
PLATFORM, APPLICATIONS, IDENTITY AND ACCESS MANAGEMENT
OPERATING SYSTEMS, NETWORK AND FIREWALL CONFIGURATION
CLIENT-SIDE DATA ENCRYPTION
SERVER-SIDE ENCRYPTION
NETWORKING TRAFFIC PROTECTION
AWS
SOFTWARE
COMPUTE
STORAGE
DATABASE
NETWORKING
HARDWARE/AWS GLOBAL INFRASTRUCTURE
REGIONS
AVAILABILITY ZONES
EDGE LOCATIONS

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.

To learn more, select the **+** symbol next to each category.

Customers: Security in the cloud

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

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

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

AWS: Security of the cloud

AWS is responsible for security *of* the cloud.

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

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

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

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

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

User permissions and access:

Video Transcript:

In the coffee shop, every employee has an identity. They come into work in the morning and they'd log into the system to clock in, use the registers and manage the systems, running the coffee shop, day to day. We have the cash registers, the computers helping run the whole operation. Each person has unique access to these systems based on who they are.

If I have Rudy at the register taking orders and Blaine in the back, checking on the inventory levels on the computer, they have two different logins and two different sets of permissions. Rudy can run the cash register, but if he went to log into the inventory system, he wouldn't be allowed to do that.

You will want to scope your users permissions in AWS in a similar way. When you create an AWS account, you are given what is called the root account user. This root user is the owner of the AWS account and has permission to do anything they want inside of that account. This is like being the owner of the coffee shop.

In this situation, let's say I am the owner of the coffee shop. I can come into the shop, use my credentials to work the register, work the inventory system or any other system in the coffee shop. I cannot be restricted. With the AWS root user, you can access and control any resource in the account. You can spin up databases, EC2 instances, blockchain services, or literally whatever you want. 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.

That is great. But even with MFA turned on, in reality you don't want to use the root user for everything. I, as the coffee shop owner, don't give my level of access to all employees. Rudy on the cash register cannot access the inventory system, remember? You control access in a granular way by using the AWS service, AWS Identity and Access Management, or IAM.

In IAM, you can create IAM users. When you create an IAM user, by default, it has no permissions. The user can't even log into the AWS account at first, it has absolutely zero permissions. It can not launch an EC2 instance. It can not create an S3 bucket. Nothing. You have to explicitly give the user permission to do anything in that account. Remember, by default, all actions are denied. You have to explicitly allow any action done by any user. You give people access only to what they need and nothing else. This idea is called the least privilege principle.

The way that you grant or deny permission is to associate what is called an IAM policy to an IAM user. An IAM policy is a JSON document that describes what API calls a user can or cannot make. Let's look at this quick example. In this example, you can see we have a permission statement that has the effect as Allow, the action as s3:ListBucket. And the resource is a unique ID for an S3 bucket. So if I attach this policy to a user and that user could view the bucket "coffee_shop_reports" but perform no other action in this account. 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. Now, as a business person, you wouldn't need to write these policies, but they are used all over in your AWS accounts.

One way to make it easier to manage your users and their permissions is to organize them into IAM groups. Groups are, well, they are groupings of users. You can attach a policy to a group and all of the users in that group will have those permissions. If you have a bunch of cashiers in the coffee shop, instead of individually granting them all access to the register. Instead, you can grant all cashiers access then just add each individual cashier to the group. Same idea with groups in IAM.

All right, so far with IAM, you have the root user, they can do anything. You have users which can be organized into groups. And you also have policies which are documents that describe permissions that you can then attach to users or groups. There is one other major identity in IAM, and it's called a role.

To understand the idea of roles, let's think about the coffee shop. As we know, Blaine works in the shop and depending on the staffing of the shop day to day, he might work the register or the inventory system or he might be the one cleaning up at the end of the day with access to no systems. I, as the owner, have the authority to assign these different roles to Blaine. His responsibilities and access are variable and change from day to day. Just because he worked on tracking inventory in the system yesterday, doesn't mean that he should be at any time. His role at work changes and is temporary in nature. The same type of idea exists in AWS. You can create identities in AWS that are called roles.

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.
You can actually avoid creating IAM users for every person in your organization by federating users into your account. This means that they could use their regular corporate credentials to log into AWS by mapping their corporate identities to IAM roles. AWS IAM authentication and authorization as a service.

AWS Identity and Access Management (IAM)

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

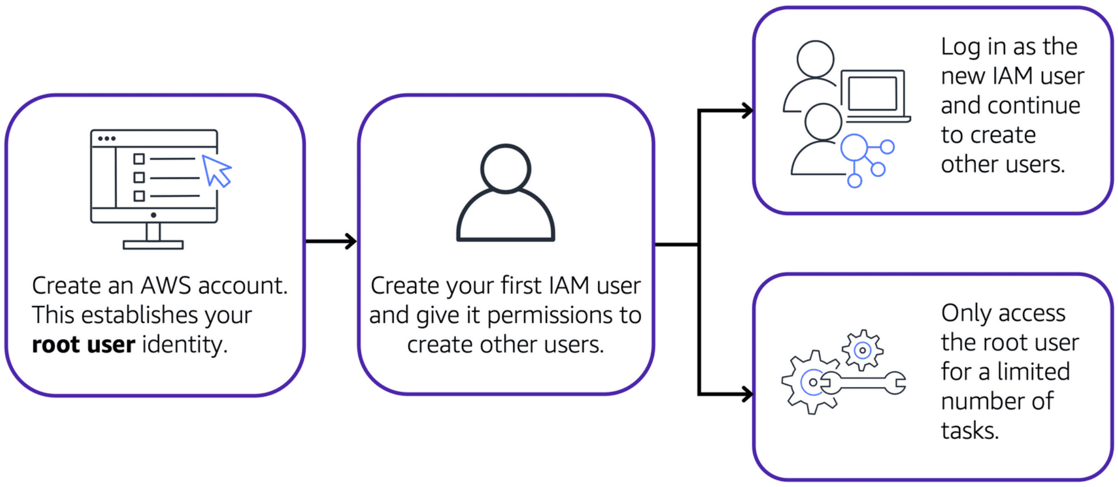
IAM gives you the flexibility to configure access based on your company's specific operational and security needs. You do this by using a combination of IAM features, which are explored in detail in this lesson:

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

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

AWS account root user

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


Create an AWS account. This establishes your **root user** identity.
Create your first IAM user and give it permissions to create other users.
Log in as the new IAM user and continue to create other users.
Only access the root user for a limited number of tasks.

Best practice:
Do **not** use the root user for everyday tasks.

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

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

IAM users

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

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

Best practice:

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

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

IAM policies

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

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

Best practice:

Follow the security principle of **least privilege** when granting permissions.

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

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

Example: IAM policy

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

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

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.

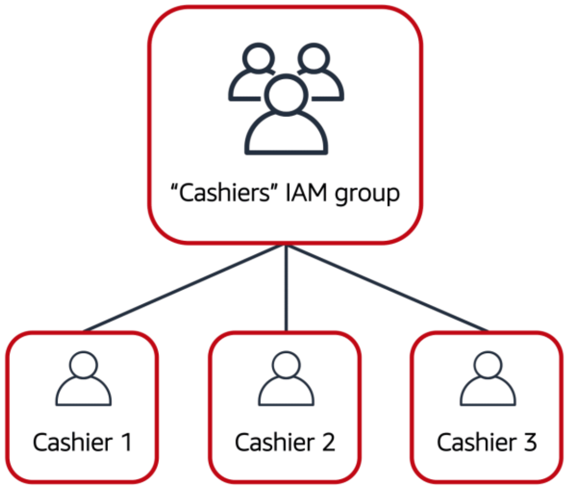
If the owner wants the cashier to be able to access other services and perform other actions in AWS, the owner must attach additional policies to specify these services and actions.

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

IAM groups

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

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



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

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

IAM roles

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

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

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

Before an IAM user, application, or service can assume an IAM role, they must be granted permissions to switch to the role. When someone assumes an IAM role, they abandon all previous permissions that they had under a previous role and assume the permissions of the new role.
Best practice:
IAM roles are ideal for situations in which access to services or resources needs to be granted temporarily, instead of long-term.
Example: IAM roles
To review an example of how IAM roles could be used in the coffee shop, select **Start**.

START

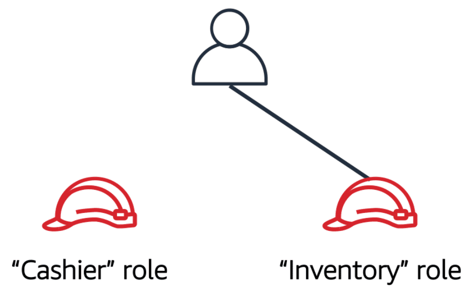
STEP-1

First, the owner grants the employee permissions to switch to a role for each workstation in the coffee shop.
Example: IAM roles
To review an example of how IAM roles could be used in the coffee shop, select **Start**.
START

First, the owner grants the employee permissions to switch to a role for each workstation in the coffee shop.

The employee begins their day by assuming the “Cashier” role. This grants them access to the cash register system.

STEP 3 :



Later in the day, the employee needs to update the inventory system. They assume the "Inventory" role.

This grants the employee access to the inventory system and also revokes their access to the cash register system.

START AGAIN

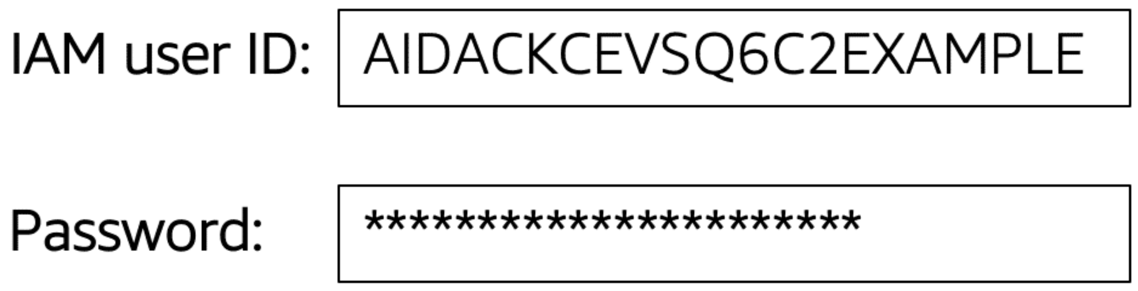
Multi-factor authentication

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

To review the steps involved with multi-factor authentication, select **Start**.



IAM user ID: AIDACKCEVSQ6C2EXAMPLE

Password: **********************

First, when a user signs in to an AWS website, they enter their IAM user ID and password.

Next, the user is prompted for an authentication response from their AWS MFA device. This device could be a hardware security key, a hardware device, or an MFA application on a device such as a smartphone.

When the user has been successfully authenticated, they are able to access the requested AWS services or resources.

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

AWS Organizations:

Video Transcript:

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. So you start to add more cards for each person, whoever needs to onboard. And before you know it, you end up with a tangled bowl of AWS account spaghetti, not as tasty as you might imagine.

For example, you'll then have to keep track of Account A, F, and G, or maybe Account B has the wrong permissions and Account C has billing and compliance info. One way to install order and to enforce who is allowed to perform certain functions in what account is to make use of an AWS service called AWS Organizations.

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. Let's outline some of the main features of AWS Organizations, shall we?

The first is centralized management of all your AWS accounts. Think of all those AWS accounts, we had: A, B, C, F, G. Now you can combine them into an organization that enables us to manage the accounts centrally, and wow. Now we've found Accounts D and E in the process. Next up is consolidated billing for all member accounts. 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. Cash money, indeed.

The next feature is that you can implement hierarchical groupings of your accounts to meet security, compliance, or budgetary needs. This means you can group accounts into organizational units, or OUs, kind of like business units, or BUs. For example, if you have accounts that must access only the AWS services that meet certain regulatory requirements, you can put those accounts into one OU, or if you have accounts that fall under the developer OU, you can group them accordingly.

One of the last main features we'll touch upon is that you have control over the AWS services and API actions that each account can access as an administrator of the primary account of an organization. You can use something called service control policies, or SCPs, to specify the maximum permissions for member accounts in the organization. In essence, with SCPs you can restrict which AWS services, resources, and individual API actions, the users and roles in each member account can access.

AWS Organizations:

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

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

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

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

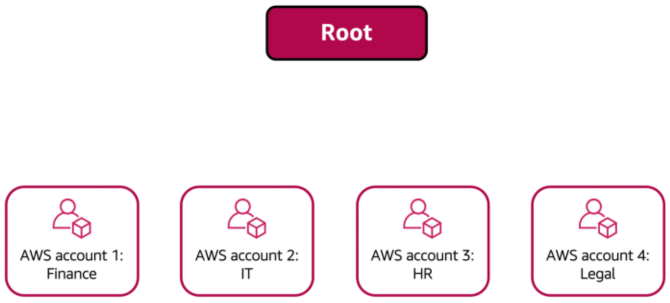
Organizational units:

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

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

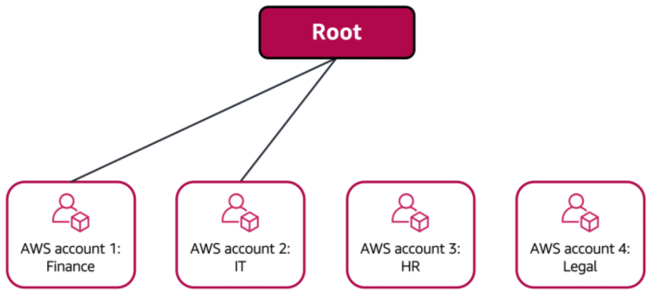
Example: AWS Organizations

To review an example of how a company might use AWS Organizations, select **Start**.

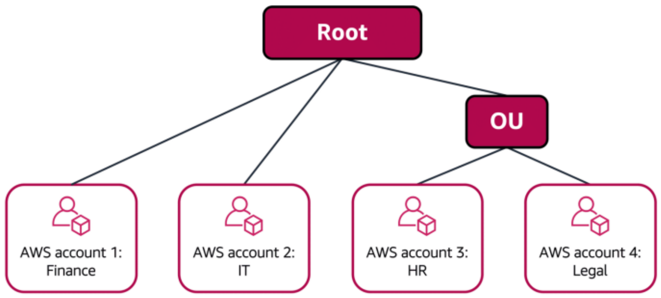


Imagine that your company has separate AWS accounts for the finance, information technology (IT), human resources (HR), and legal departments. You decide to consolidate these accounts into a single organization so that you can administer them from a central location. When you create the organization,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

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



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



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

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

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

Knowledge check

You are configuring service control policies (SCPs) in AWS Organizations. Which identities and resources can SCPs be applied to? (Select TWO.)

- IAM users
- IAM groups
- An individual member account
- IAM roles
- An organizational unit (OU)

The correct two response options are:

- **An individual member account**
- **An organizational unit (OU)**

In AWS Organizations, you can apply service control policies (SCPs) to the organization root, an individual member account, or an OU. An SCP affects all IAM users, groups, and roles within an account, including the AWS account root user.

You can apply IAM policies to IAM users, groups, or roles. You cannot apply an IAM policy to the AWS account root user.

Learn more:

- AWS Organizations
- Service control policies
- Attaching SCPs

Compliance:

Video Transcript:

For every industry, there are specific standards that need to be upheld, and you will be audited or inspected to ensure that you have met those standards. For example, for a coffee shop, the health inspector will come by and check that everything is up to code and sanitary. Similarly, 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. Depending on what types of solutions you host on AWS, you will need to ensure that you are up to compliance for whatever standards and regulations your business is specifically held to. If you run software that deals with consumer data in the EU, you would need to make sure that you're in compliance with GDPR, or if you run healthcare applications in the US you will need to design your architectures to meet HIPAA compliance requirements.

Whatever your compliance need is, you'll need some tools to be able to collect documents, records and inspect your AWS environment to check if you meet the compliance regulations that you're under. The first thing to note is, AWS has already built out data center infrastructure and networking following industry best practices for security, and as an AWS customer, you inherit all the best practices of AWS policies, architecture, and operational processes.

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

You also should be very aware of the fact that you own your data in AWS. As shown in the AWS shared responsibility model, you have complete control over the data that you store in AWS. You can employ multiple different encryption mechanisms to keep your data safe, and that varies from service to service. So, if you need specific standards for data storage, you can devise a way to either reach those requirements by building it yourself on top of AWS or using the features that already exist in many services. For a lot of services, enabling data protection is a configuration setting on the resource.

AWS also offers multiple whitepapers and documents that you can download and use for compliance reports. Since you aren't running the data center 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. With AWS Artifact, you can gain access to compliance reports done by third parties who have validated a wide range of compliance standards. Check out the AWS Compliance Center in order to find compliance information all in one place. It will show you compliance enabling services as well as documentation like the AWS Risk and Security Whitepaper, which you should read to ensure that you understand security and compliance with AWS.

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.

AWS Artifact

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

AWS Artifact is a service that provides on-demand access to AWS security and compliance reports and select online agreements. AWS Artifact consists of two main sections: AWS Artifact Agreements and AWS Artifact Reports.
To learn more, select the **+** symbol next to each section.

AWS Artifact Agreements

Suppose that your company needs to sign an agreement with AWS regarding your use of certain types of information throughout AWS services. You can do this through **AWS Artifact Agreements**.

In AWS Artifact Agreements, you can review, accept, and manage agreements for an individual account and for all your accounts in AWS Organizations. Different types of agreements are offered to address the needs of customers who are subject to specific regulations, such as the Health Insurance Portability and Accountability Act (HIPAA).

AWS Artifact Reports

Next, suppose that a member of your company's development team is building an application and needs more information about their responsibility for complying with certain regulatory standards. You can advise them to access this information in **AWS Artifact Reports**.

AWS Artifact Reports provide compliance reports from third-party auditors. These auditors have tested and verified that AWS is compliant with a variety of global, regional, and industry-specific security standards and regulations. AWS Artifact Reports remains up to date with the latest reports released. You can provide the AWS audit artifacts to your auditors or regulators as evidence of AWS security controls. The following are some of the compliance reports and regulations that you can find within AWS Artifact. Each report includes a description of its contents and the reporting period for which the document is valid.

Global
cloud security alliance
ISO 9001
ISO 27001
ISO 27017
ISO 27018
PCI Security Standards Council
PARTICIPATING ORGANIZATION
AICPA SOC
AICPA SOC
AICPA SOC
USA
FedRAMP
FFIEC
FIPS 140-2
FISMA
HIPAA
HITRUST CSF Certified
ITAR
Europe
C5
CYBER ESSENTIALS PLUS
ens
Asia Pacific
irap
ISMS
iDA SINGAPORE
TÜV AUSTRIA
TISAX

Customer Compliance Center

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

In the Customer Compliance Center, 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 Center 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.

Knowledge check

Which tasks can you complete in AWS Artifact? (Select TWO.)

- Access AWS compliance reports on-demand.
- Consolidate and manage multiple AWS accounts within a central location.

- Create users to enable people and applications to interact with AWS services and resources.
- Set permissions for accounts by configuring service control policies (SCPs).
- Review, accept, and manage agreements with AWS.

The correct two response options are:

- **Access AWS compliance reports on-demand.**
- **Review, accept, and manage agreements with AWS.**

The other response options are incorrect because:

- Consolidate and manage multiple AWS accounts within a central location- This task can be completed in *AWS Organizations*.
- Create users to enable people and applications to interact with AWS services and resources- This task can be completed in *AWS Identity and Access Management (IAM)*.
- Set permissions for accounts by configuring service control policies (SCPs)- This task can be completed in *AWS Organizations*.

Learn more:

- AWS Artifact

Denial-of-service attacks:

Video Transcript:

D-D-o-S, DDoS, the distributed denial-of-service. It's an attack on your enterprise's infrastructure, and you've heard of it. Your security team might have written a plan for it, and you know that many businesses have been devastated by it. But what exactly is it, and more importantly, how can you defend against it?

Now to be clear, this is a 14-hour discussion to really understand it all, but you need to at least know the fundamentals of how the attacks are carried out, and how AWS can automatically defend your infrastructure from these crippling assaults. Now we don't have a lot of time to cover all this, and the clock starts now. The objective of a DDoS attack is to shut down your application's ability to function by overwhelming the system to the point it can no longer operate.

In normal operations, your application takes requests from customers and returns results. In a DDoS attack, the bad actor tries to overwhelm the capacity of your application, basically to deny anyone your services. But a single machine attacking your application has no hope of providing enough of an attack by itself, so the distributed part is that the attack leverages other machines around the internet to unknowingly attack your infrastructure. The bad actor creates an army of zombie bots, brainlessly assaulting your enterprise. The key to a good attack, and I call it that when I should call it powerful. I mean, it's definitely chaotic evil, but the key is to have the assault commander do the smallest amount of work needed, and have the targeted victim receive an unbearable load of resulting work they must process through.

So let me cherry-pick a few specific attack examples that work really well. The 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. Now that is one example of half a dozen low-level, brute force attacks, all designed to exhaust your network.

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.

They even try horrible tricks like the Slowloris attack. Mm-hmm. Imagine standing in line at the coffee shop, when someone in front of you takes seven minutes to order their 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. I could go on monologuing for hours just talking about the elegantly evil architecture of these attacks, but we are on the clock here, and it is time to stop these attacks cold. And here's the cool solution: You already know the solution.

Everything we've been talking about over this entire course is not only good architecture, but it also helps solve almost all DDoS attack vectors with zero additional effort or cost. First attack, the low level network attacks like the UDP floods. Solution, security groups. The security groups only allow in proper request traffic. Things like weather reports use an entirely different protocol than the ones your customers use. Not on the list, you don't get to talk to the server. And what's more, security groups operate at the AWS network level, not at the EC2 instance level, like an operating system firewall might.

So massive attacks like UDP floods or reflection attacks just get shrugged off by the scale of the entire AWS Regions capacity, not your individual EC2's capacity. This is a case where our size is a huge advantage in your protection. I won't say it's impossible to overwhelm AWS, but the scale it would take, it would be too expensive for these bad actors. Slowloris attacks? Look at our elastic load balancer. Because the ELB handles the http traffic request first, so it waits until the entire message, no matter how fast or slow, is complete before sending it over to the front end web server. I mean, sure, you can try to overwhelm it, but remember how the ELB is scalable and how it runs at the region level?

To overwhelm ELB, you would once again have to overwhelm the entire AWS region. It's not theoretically impossible, but too massively expensive for anyone to pull off. For the sharpest, most sophisticated attacks, AWS also offers specialized defense 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.

All right, that's it, the clock is almost up. The takeaway is a well-architected system is already defended against most attacks. And by using AWS Shield Advanced, you can turn AWS into your partner against DDoS attacks. Oh, oh.

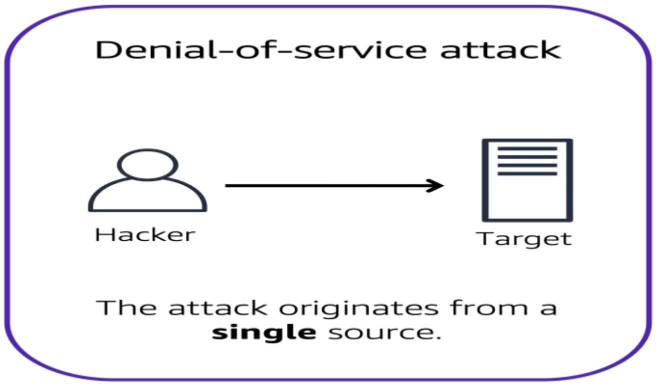
Customers can call the coffee shop to place their orders. After answering each call, a cashier takes the order and gives it to the barista.

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

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

Denial-of-service attacks

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

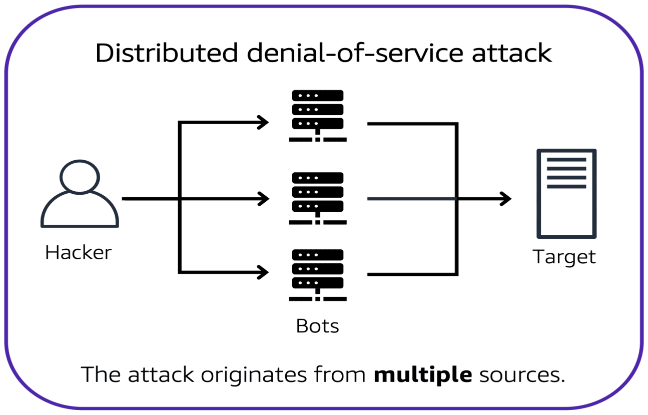


For example, an attacker might flood a website or application with excessive network traffic until the targeted website or application becomes overloaded and is no longer able to respond. If the website or application becomes unavailable, this denies service to users who are trying to make legitimate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

Distributed denial-of-service attacks

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

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


Distributed denial-of-service attack
Hacker
Bots
Target
The attack originates from multiple sources.

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

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

AWS Shield

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

To learn more, select the **+** symbol next to each service.

AWS Shield Standard

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

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

AWS Shield Advanced

AWS Shield Advanced is a paid service that provides detailed attack diagnostics and the ability to detect and mitigate sophisticated DDoS attacks.

It also integrates with other services such as Amazon CloudFront, Amazon Route 53, and Elastic Load Balancing. Additionally, you can integrate AWS Shield with AWS WAF by writing custom rules to mitigate complex DDoS attacks.

Additional security services:

Video Transcript:

With all the comings and goings on in your coffee shop, you'll want to increase security to your coffee beans, equipment, and even money in the till. For your beans, this could be when they're sitting in your store room, or even when you're transporting them between shops. After all, we don't want unwanted visitors with access to our coffee beans, or even running off with precious equipment.

To start off, let's chat about how you can secure your coffee beans, or your data whether it's at rest, or in transit. For our beans, the simple way to do it would be to lock the door when we'd leave at night. That's the notion of encryption, which is securing a message or data in a way that can only be accessed by authorized parties. Non-authorized parties are therefore less likely to be able to access the message. Or not able to access it at all. Think of it as that key and door example. If you have the key, you can unlock the door. But if you don't, then you cannot unlock that door.

At AWS, this comes in two variations. Encryption at rest and encryption in transit. By at rest, we mean when your data is idle. It's just being stored and not moving. 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. So make sure to keep it safe.

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.

But speaking of other services, the next service we want to highlight is called Amazon Inspector. Inspector helps to improve security, and compliance of your AWS deployed applications by running an automated security assessment against your infrastructure. Specifically, it helps to check on deviations of security best practices, exposure of EC2 instances, vulnerabilities, and so forth. The service consists of three parts a network configuration reachability piece, an Amazon agent, which can be installed an EC2 instances, and a security assessment service that brings them all together. To use it, you configure Inspector options, run the service, out pops a list of potential security issues. The resulting findings are displayed in the Amazon Inspector console, and they are presented with a detailed description of the security issue, and a recommendation on how to fix it. Additionally, you can retrieve findings through an API. So as to go towards the best practice of performing remediation to fix issues.

Another service dimension is our threat detection offering known as Amazon GuardDuty. It analyzes continuous streams of metadata generated from your account, and network activity found on AWS CloudTrail events, Amazon 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.

They are so many other security services like Advanced Shield and Security Hub. So please check out the Resources section to learn more. Thanks for following along.

AWS Key Management Service (AWS KMS)

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

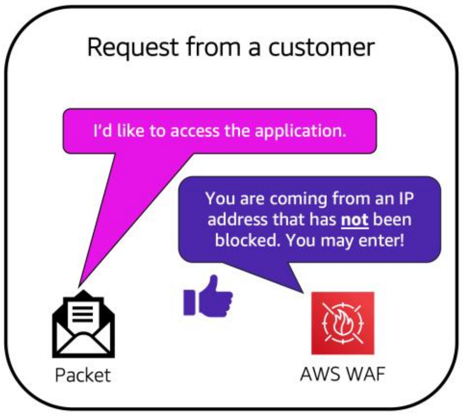
In the same way, you must ensure that your applications' data is secure while in storage **(encryption at rest)** and while it is transmitted, known as **encryption in transit**.

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

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

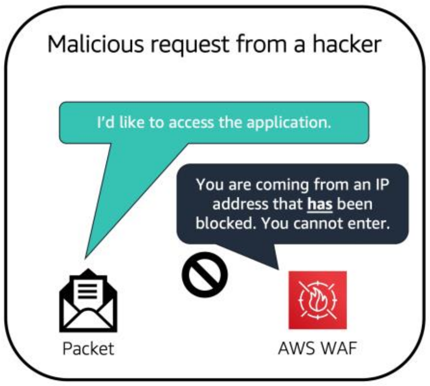
AWS WAF

AWS WAF is a web application firewall that lets you monitor network requests that come into your web applications.
AWS WAF works together with Amazon CloudFront and an Application Load Balancer. Recall the network access control lists that you learned about in an earlier module. AWS WAF works in a similar way to block or allow traffic. However, it does this by using a **web access control list (ACL)** to protect your AWS resources.
Here's an example of how you can use AWS WAF to allow and block specific requests.


Request from a customer
I'd like to access the application.
You are coming from an IP address that has **not** been blocked. You may enter!
Packet
AWS WAF

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.


Malicious request from a hacker
I'd like to access the application.
You are coming from an IP address that **has** been blocked. You cannot enter.
Packet
AWS WAF

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

Amazon Inspector

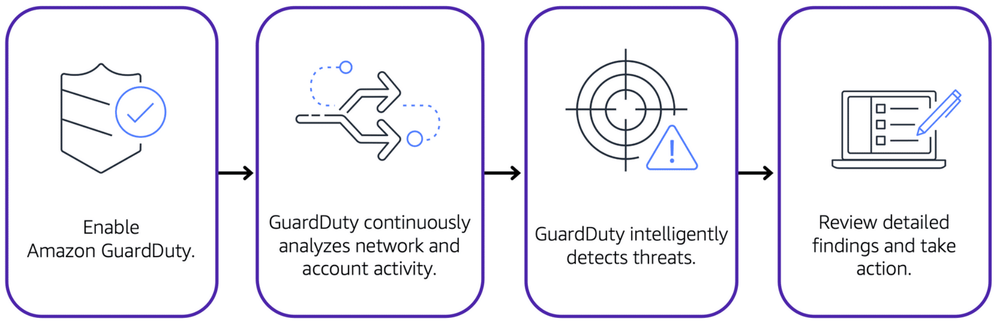
Suppose that the developers at the coffee shop are developing and testing a new ordering application. They want to make sure that they are designing the application in accordance with security best practices. However, they have several other applications to develop, so they cannot spend much time conducting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

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

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

Amazon GuardDuty

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


Enable Amazon GuardDuty.
GuardDuty continuously analyzes network and account activity.
GuardDuty intelligently detects threats.
Review detailed findings and take action.

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

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

Module 6 summary:

In Module 6, you learned about the following concepts:

- The shared responsibility model
- Features of AWS Identity and Access Management

- Methods of managing multiple accounts in AWS Organizations
- AWS compliance resources
- AWS services for application security and encryption

Video Transcript:

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.
