SWINBURNE UNIVERSITY OF TECHNOLOGY

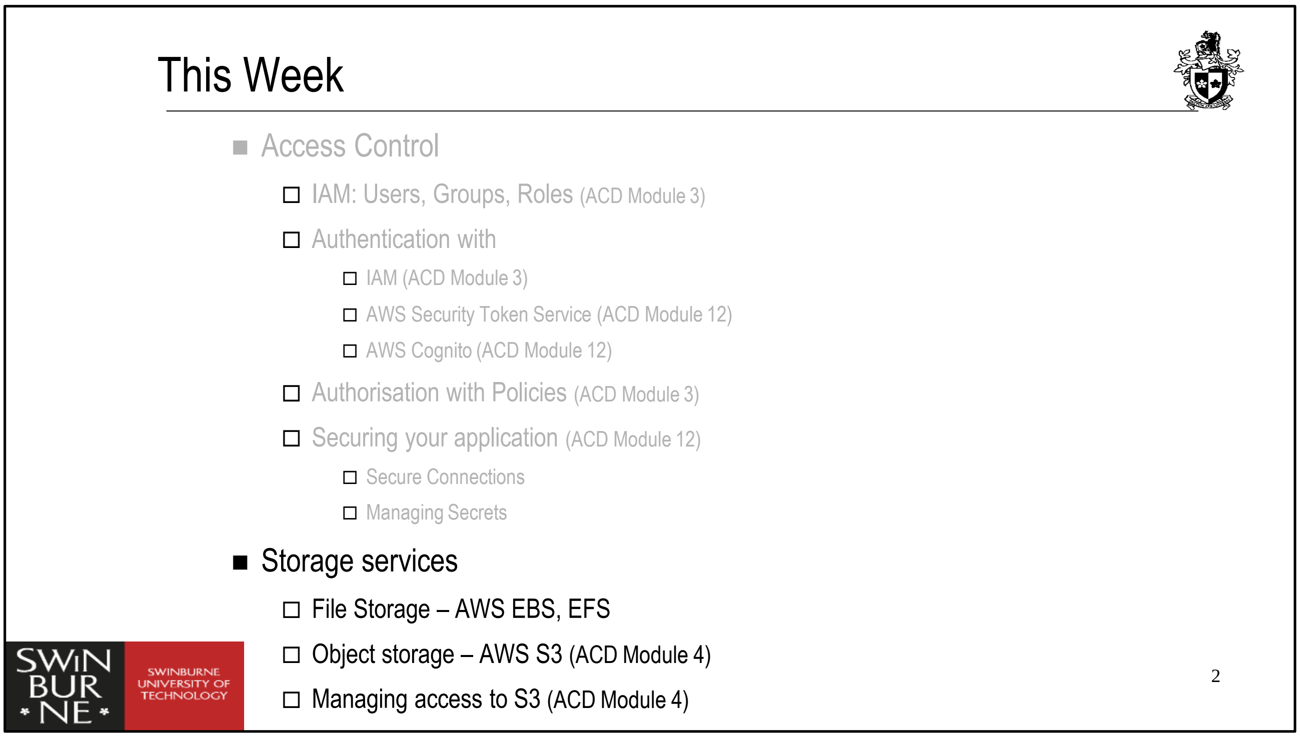
COS80011
Cloud-Native Web Applications

Lecture 03

Access Control

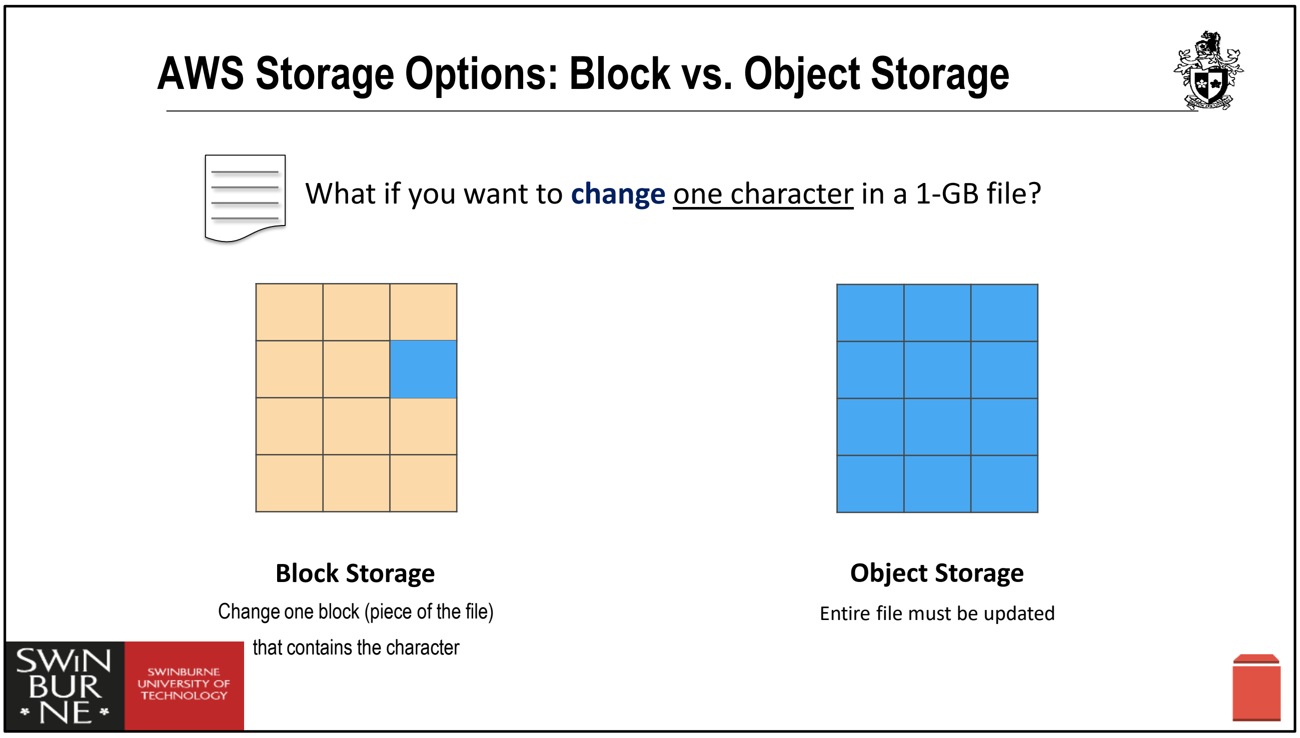
Storage Services

Phu Lai



This Week

- Access Control
  - IAM: Users, Groups, Roles (ACD Module 3)
  - Authentication with
    - IAM (ACD Module 3)
    - AWS Security Token Service (ACD Module 12)
    - AWS Cognito (ACD Module 12)
  - Authorisation with Policies (ACD Module 3)
  - Securing your application (ACD Module 12)
    - Secure Connections
    - Managing Secrets
- Storage services
  - File Storage – AWS EBS, EFS
  - Object storage – AWS S3 (ACD Module 4)
  - Managing access to S3 (ACD Module 4)



AWS Storage Options: Block vs. Object Storage

What if you want to **change** one character in a 1-GB file?

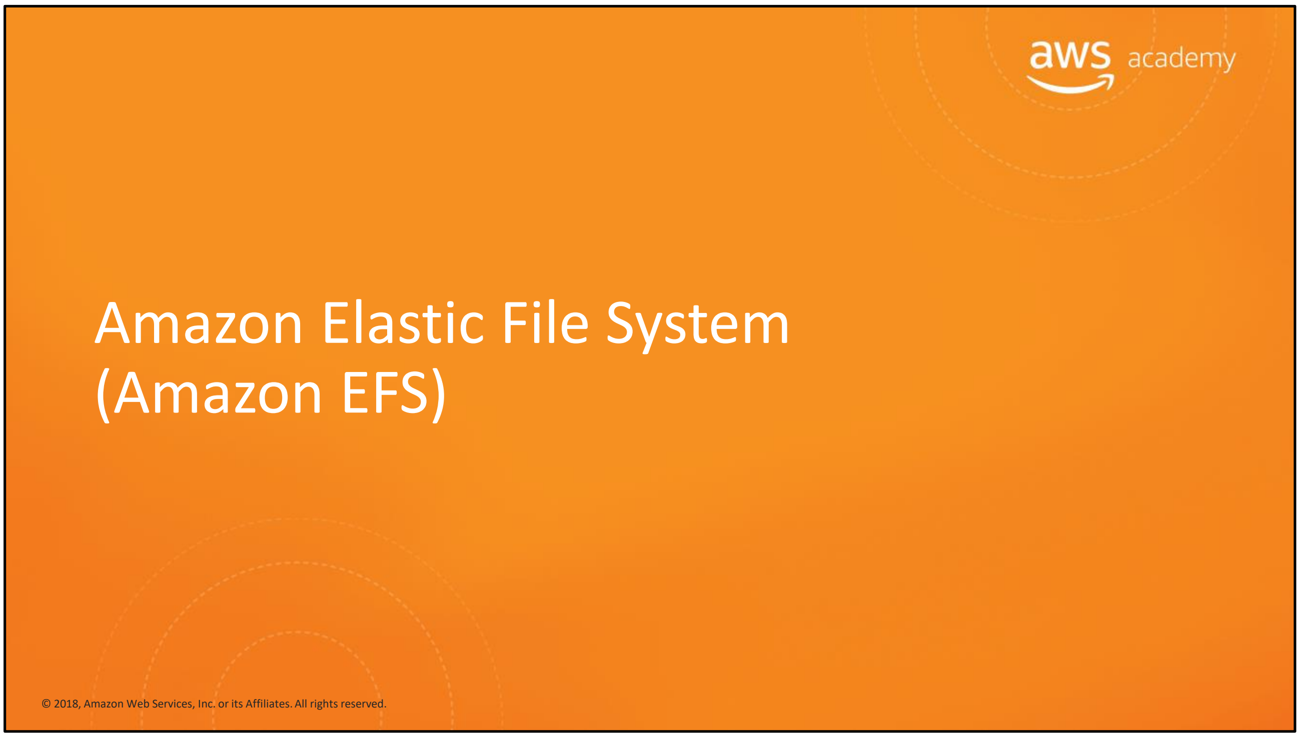
Block Storage

Change one block (piece of the file) that contains the character

Object Storage

Entire file must be updated

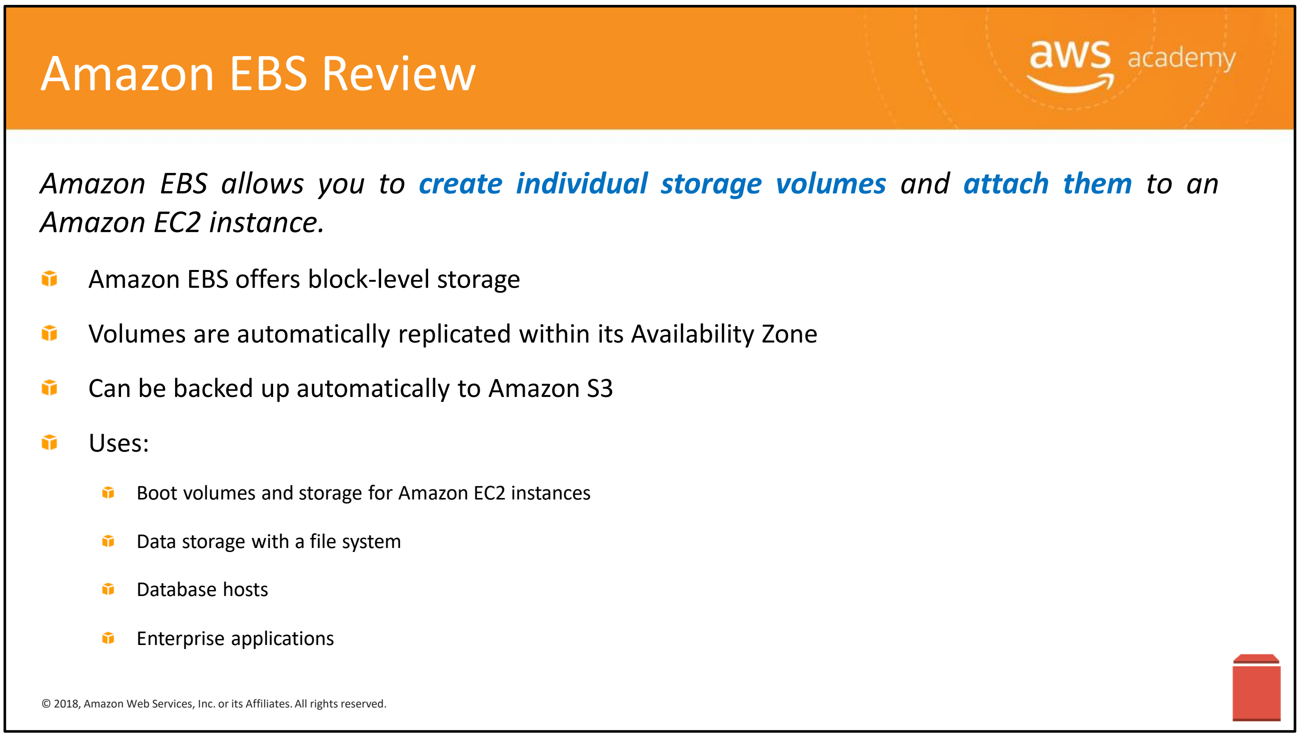
One of the critical concepts to understanding the differences between some storage types is whether they offer "block-level" storage or "object-level" storage. This difference has a major impact on the throughput, latency, and cost of your storage solution: block storage solutions are typically faster and use less bandwidth but cost more than object-level storage.



aws academy

Amazon Elastic File System (Amazon EFS)

Amazon Elastic File System (Amazon EFS) provides simple, scalable file storage for use with Amazon EC2 instances in the AWS Cloud. Amazon EFS is easy to use and offers a simple interface that allows you to create and configure file systems quickly and easily.

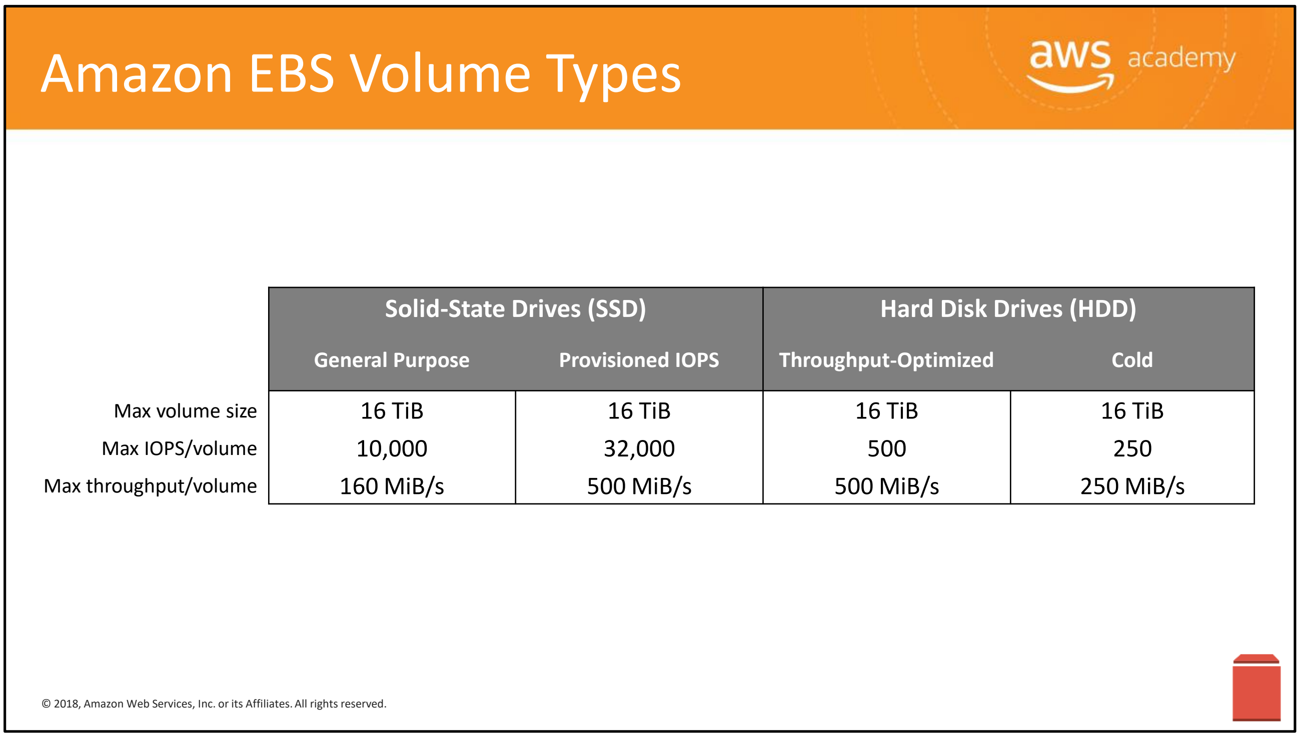


Amazon EBS Review

*Amazon EBS allows you to **create individual storage volumes** and **attach them** to an Amazon EC2 instance.*

- Amazon EBS offers block-level storage
- Volumes are automatically replicated within its Availability Zone
- Can be backed up automatically to Amazon S3
- Uses:
  - Boot volumes and storage for Amazon EC2 instances
  - Data storage with a file system
  - Database hosts
  - Enterprise applications

Amazon EBS volumes provide durable, detachable, block-level storage (like an external hard drive) for your Amazon EC2 instances. Because they are directly attached to the instances, they can provide extremely low latency between where the data is stored and where it might be used on the instance. For this reason, they can be used to run a database with an Amazon EC2 instance. Amazon EBS volumes can also be used to back up your instances into Amazon Machine Images (AMI), which are stored in Amazon S3 and can be reused to create new Amazon EC2 instances later.

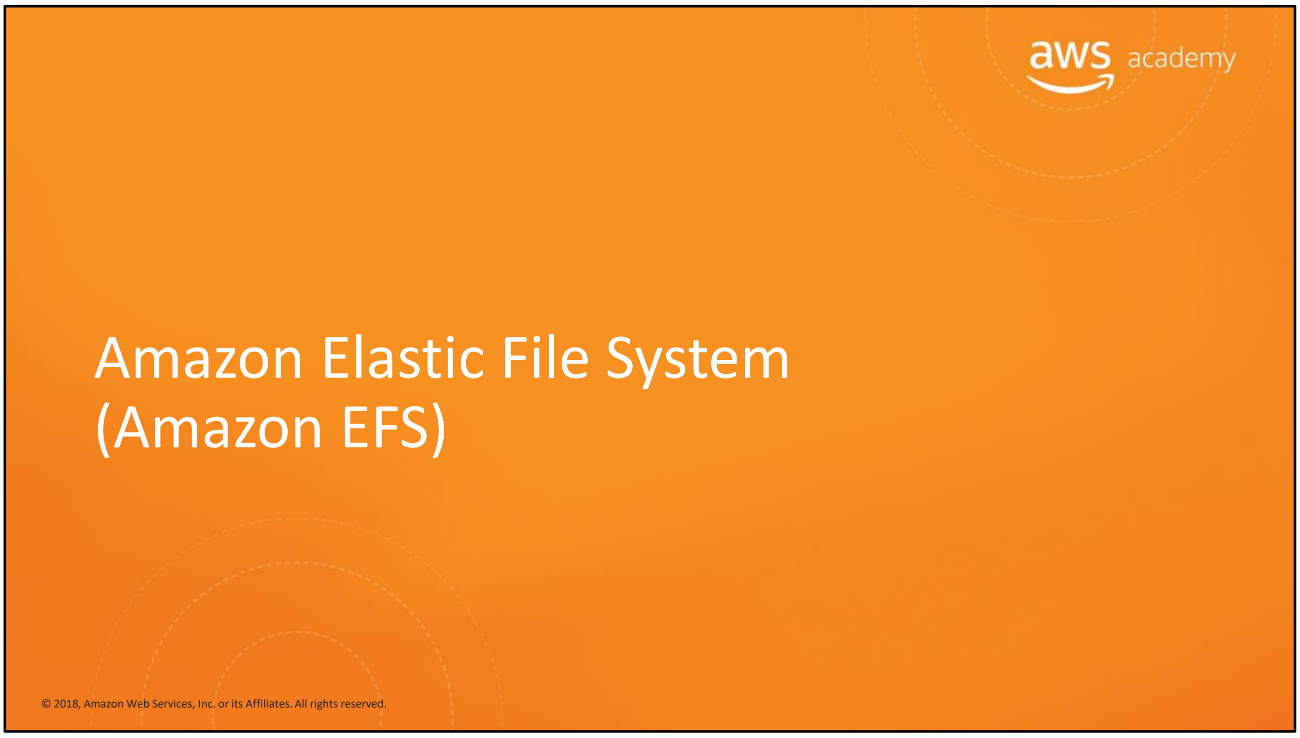


Amazon EBS Volume Types

	Solid-State Drives (SSD)		Hard Disk Drives (HDD)	
	General Purpose	Provisioned IOPS	Throughput-Optimized	Cold
Max volume size	16 TiB	16 TiB	16 TiB	16 TiB
Max IOPS/volume	10,000	32,000	500	250
Max throughput/volume	160 MiB/s	500 MiB/s	500 MiB/s	250 MiB/s

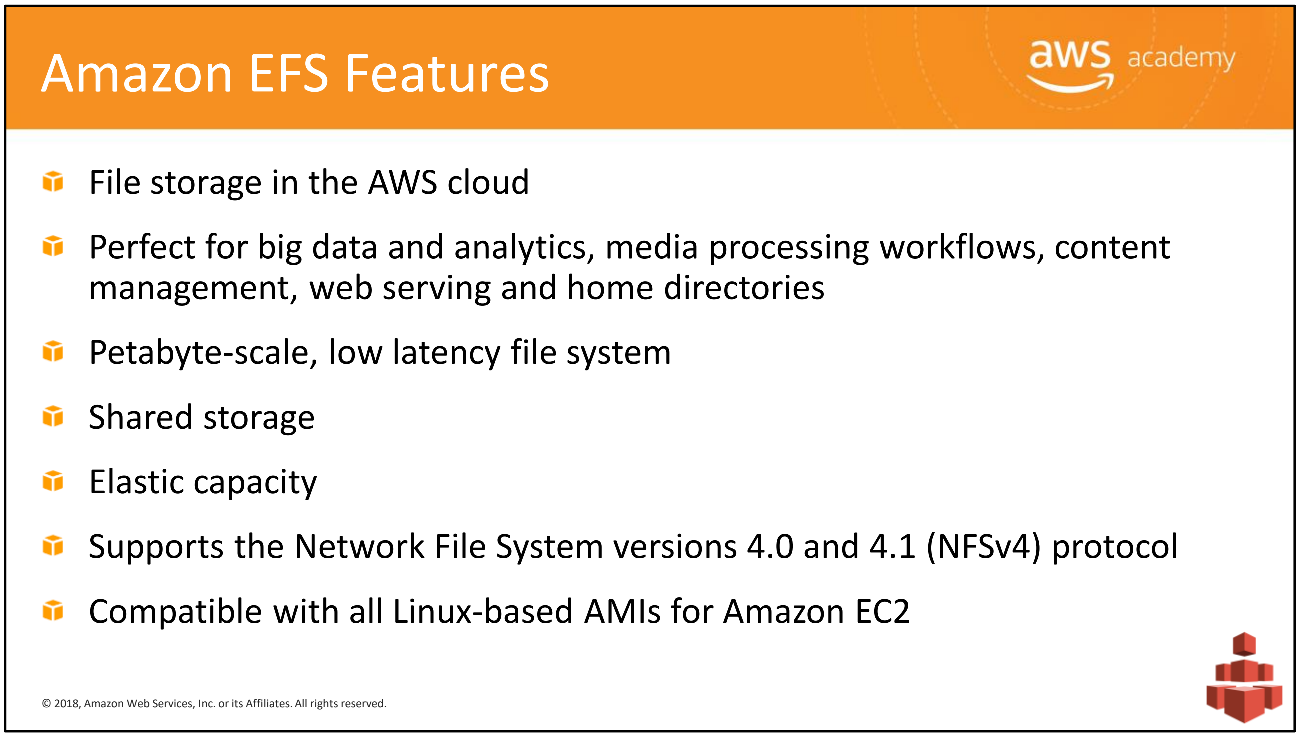
Matching the correct technology to your workload is a key best practice for reducing storage costs. Provisioned IOPS SSD-backed Amazon EBS volumes can give you the highest performance, but if your application doesn't require or won't use performance that high, one of the lower-cost options might be a better solution.

For more information, see http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSVolumeTypes.html.



Amazon Elastic File System (Amazon EFS)

Amazon Elastic File System (Amazon EFS) provides simple, scalable file storage for use with Amazon EC2 instances in the AWS Cloud. Amazon EFS is easy to use and offers a simple interface that allows you to create and configure file systems quickly and easily.

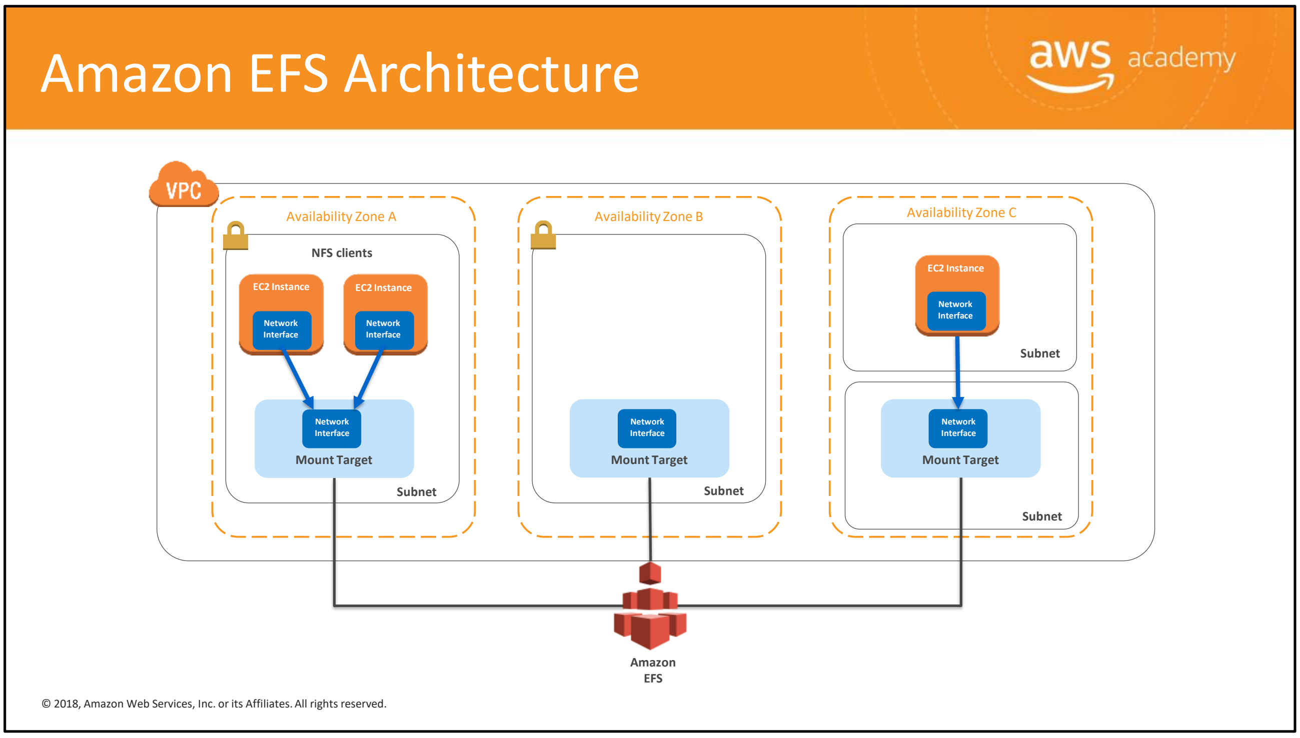


Amazon EFS Features

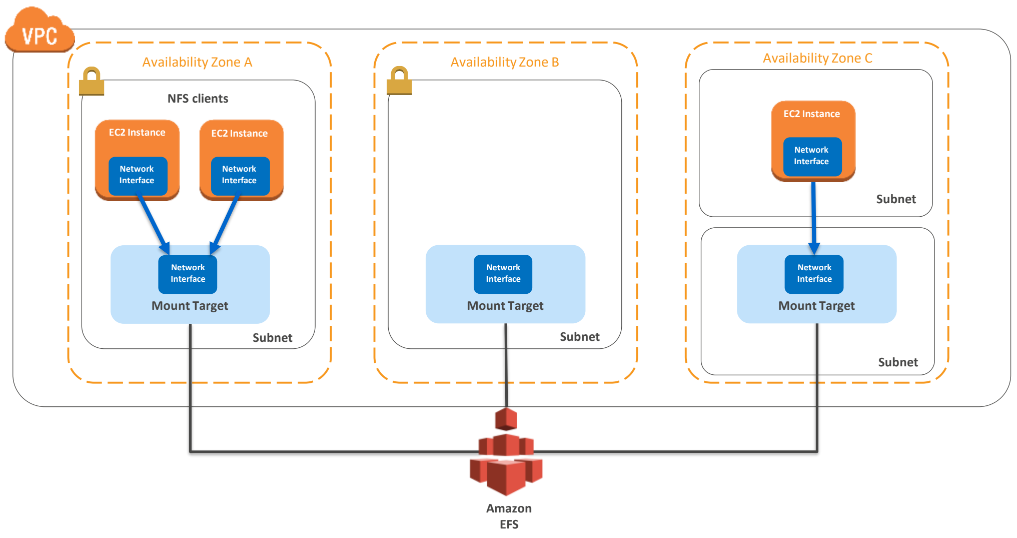
- File storage in the AWS cloud
- Perfect for big data and analytics, media processing workflows, content management, web serving and home directories
- Petabyte-scale, low latency file system
- Shared storage
- Elastic capacity
- Supports the Network File System versions 4.0 and 4.1 (NFSv4) protocol
- Compatible with all Linux-based AMIs for Amazon EC2

Amazon EFS is a fully managed service that makes it easy to set up and scale file storage in the AWS cloud. It is the easiest way to build a file system for big data and analytics, media processing workflows, content management, web serving and home directories. You can create file systems that are accessible to Amazon EC2 instances via a file system interface (using standard operating system file I/O APIs) and that support full file system access semantics (such as strong consistency and file locking).

Amazon EFS file systems can automatically scale from gigabytes to petabytes of data without needing to provision storage. Thousands of Amazon EC2 instances can access an Amazon EFS file system at the same time, and Amazon EFS provides consistent performance to each Amazon EC2 instance. Amazon EFS is designed to be highly durable and highly available. With Amazon EFS, there is no minimum fee or setup costs, and you pay only for the storage you use.



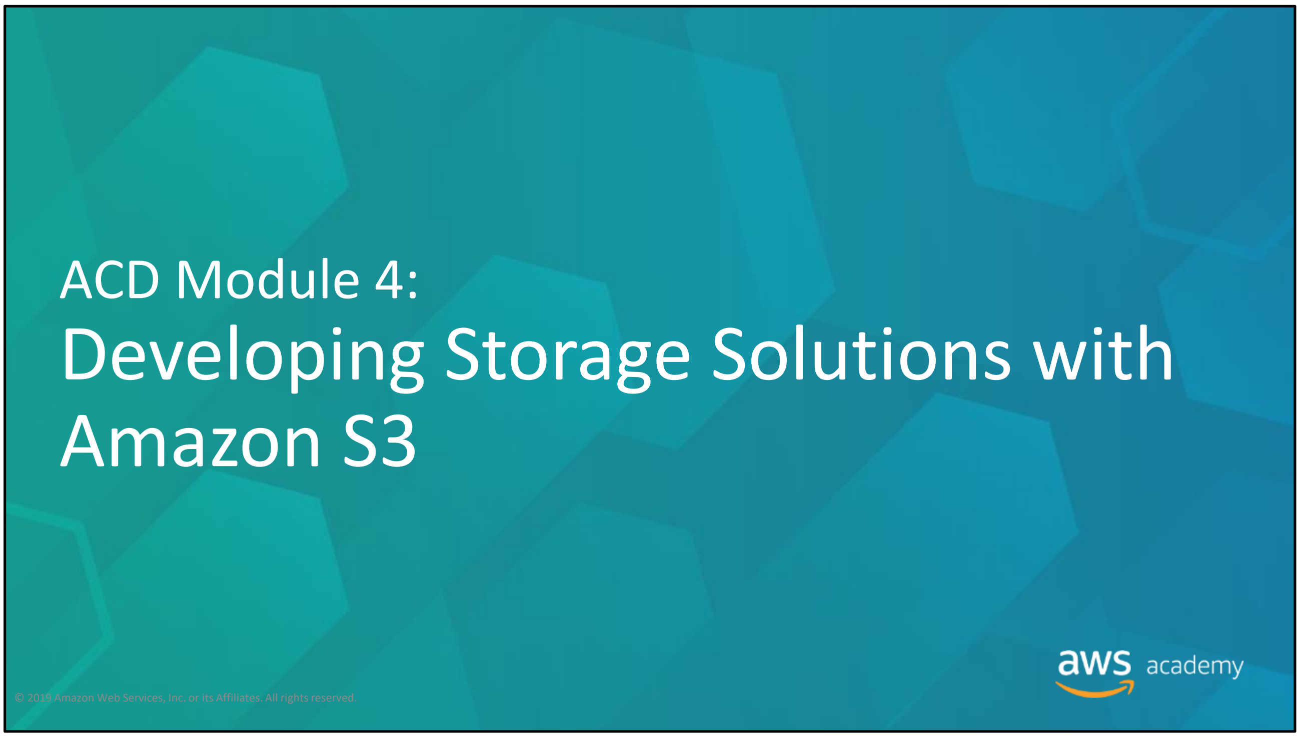
Amazon EFS Architecture



Amazon EFS provides file storage in the cloud. With Amazon EFS, you can create a file system, mount the file system on an Amazon EC2 instance, and then read and write data from to and from your file system. You can mount an Amazon EFS file system in your VPC, through the Network File System versions 4.0 and 4.1 (NFSv4) protocol.

You can access your Amazon EFS file system concurrently from Amazon EC2 instances in your Amazon VPC, so applications that scale beyond a single connection can access a file system. Amazon EC2 instances running in multiple Availability Zones within the same AWS Region can access the file system, so that many users can access and share a common data source.

In this illustration, the VPC has three Availability Zones, and each has one mount target created in it. We recommend that you access the file system from a mount target within the same Availability Zone. Note that one of the Availability Zones has two subnets. However, a mount target is created in only one of the subnets.

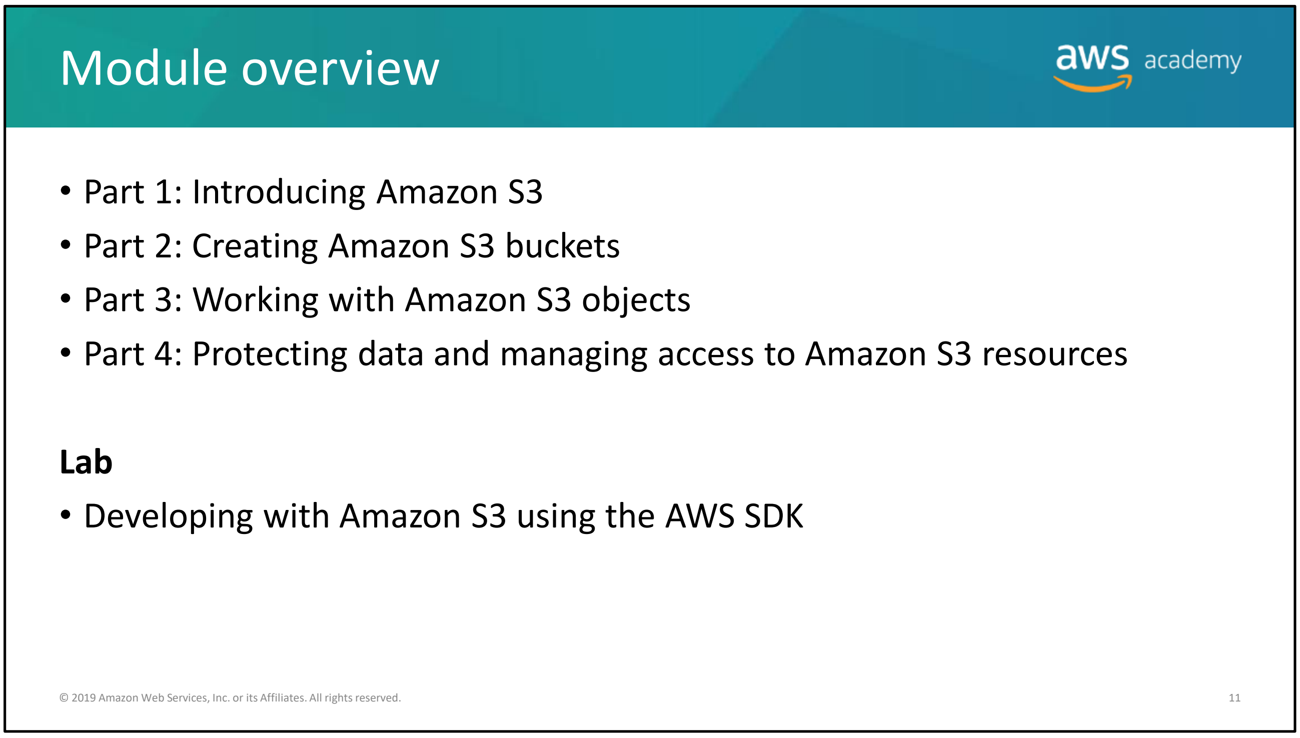


ACD Module 4:

Developing Storage Solutions with Amazon S3

aws academy

Welcome to Module 4: Developing Storage Solutions with Amazon Simple Storage Service (Amazon S3)



Module overview

- Part 1: Introducing Amazon S3
- Part 2: Creating Amazon S3 buckets
- Part 3: Working with Amazon S3 objects
- Part 4: Protecting data and managing access to Amazon S3 resources

Lab

- Developing with Amazon S3 using the AWS SDK

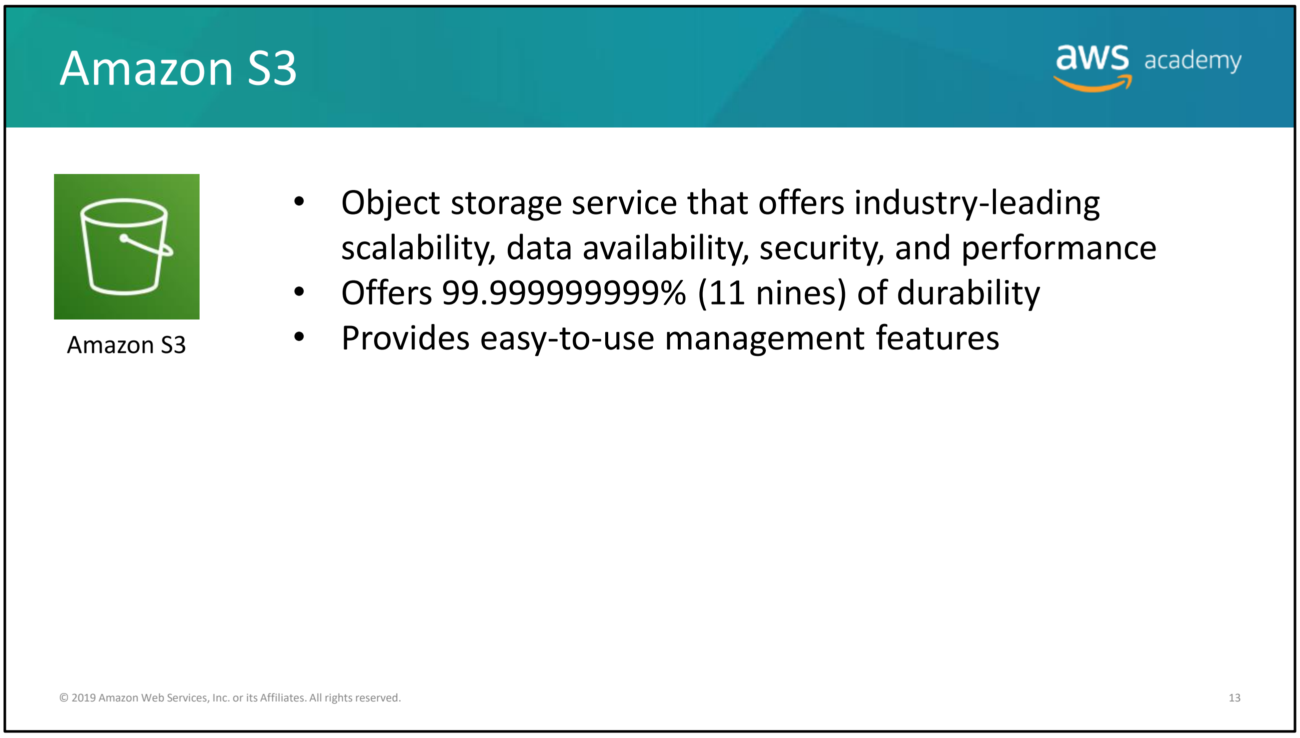
This module covers:

- Introducing Amazon S3
- Creating Amazon S3 buckets
- Working with Amazon S3 objects
- Protecting data and managing access to Amazon S3 resources

At the end of this module, you will complete the lab *Developing with Amazon S3 using the AWS SDK*.

Part 1: Introduction to Amazon S3

Part1: Introduction to Amazon S3



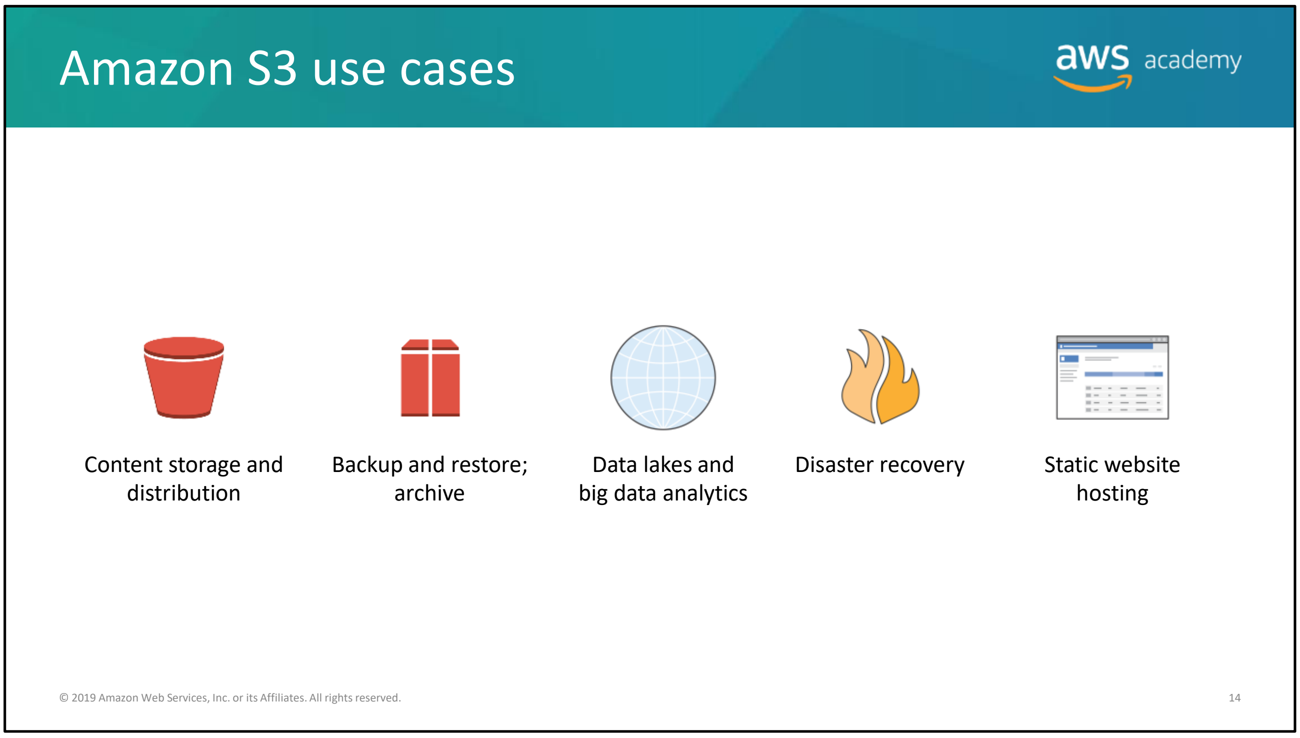
Amazon S3

Amazon S3

- Object storage service that offers industry-leading scalability, data availability, security, and performance
- Offers 99.999999999% (11 nines) of durability
- Provides easy-to-use management features

Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. This means customers of all sizes and industries can use it to store and protect any amount of data for a range of use cases, such as websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics. Amazon S3 provides easy-to-use management features so you can organize your data and configure finely-tuned access controls to meet your specific business, organizational, and compliance requirements. Amazon S3 is designed for 99.999999999% (11 9's) of durability, and stores data for millions of applications for companies all around the world.

For more information on Amazon S3, see the product page: https://aws.amazon.com/s3/



Amazon S3 use cases

aws academy

Content storage and distribution

Backup and restore; archive

Data lakes and big data analytics

Disaster recovery

Static website hosting

There are many use cases for Amazon S3, such as content storage and distribution, backup and restore, archiving, data lakes and big data analytics, disaster recovery, and static website hosting.

- You can securely store static content—such as images, videos, and music—to Amazon S3, and then distribute the content directly from Amazon S3 or through Amazon CloudFront edge locations.
    - Case Study: Classle is a cloud-based social learning platform that allows students to connect with other students and professionals from various areas. Amazon S3, with the Reduced Redundancy Storage (RRS) feature, serves the dual function of providing Classle's content downloads and acting as an origin server for Amazon CloudFront. Read the case study: https://aws.amazon.com/solutions/case-studies/classle/.

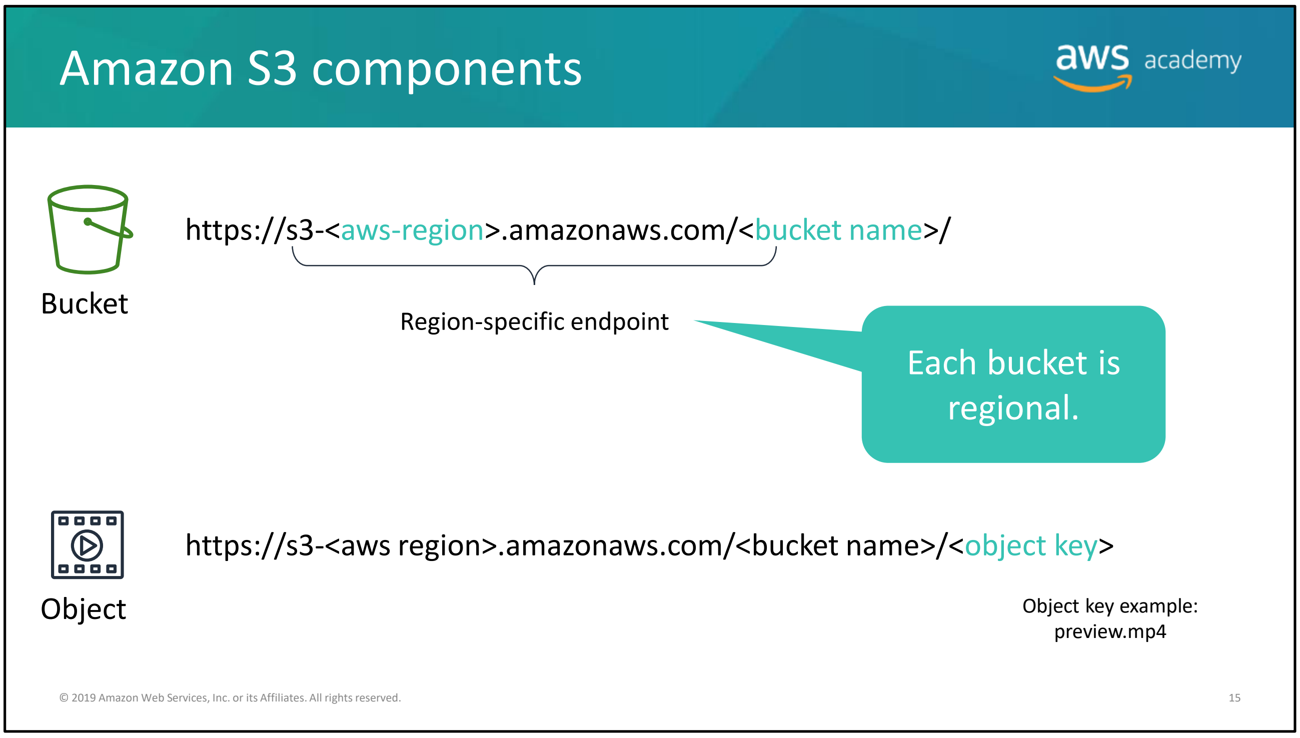
- You can back up your data and use storage lifecycle policies to archive rarely accessed data to Amazon S3 Glacier.
    - Case Study: Celgene uses Amazon S3 and Amazon Glacier to store hundreds of terabytes of genomic data. "Some of our genomic files are 200 gigabytes each in size, after compression, so we need the robust storage capabilities of Amazon S3 and Amazon Glacier," says Lance Smith, associate director of IT at Celgene. Read the case study: https://aws.amazon.com/solutions/case-studies/celgene/

- You can create a data lake in Amazon S3. A data lake is a centralized repository that allows you to store all your structured and unstructured data at any scale. With a data lake, you can extract valuable insights using query-in-place, analytics, and machine learning tools.

    - Case Study: The UK Data Service is responsible for the United Kingdom's largest collection of economic and social data, used by researchers, policy makers, and private companies. It uses Amazon S3 to store the cloud-based portion of its data lake. Read the case study: https://aws.amazon.com/solutions/case-studies/uk-data-service/

- The highly durable, secure, global infrastructure provided by Amazon S3 offers a robust disaster recovery solution. You can use Amazon S3 to protect critical data, applications, and IT systems that are running in the AWS Cloud or in your on-premises environment without incurring the expense of a second physical site. Cross-Region replication (CRR) enables automatic, asynchronous copying of objects across buckets in different AWS Regions.
    - Case Study: 6 Waves Limited is a leading publisher and developer of gaming applications on the Facebook platform. 6 Waves uses Amazon S3 and Amazon Elastic Compute Cloud (Amazon EC2) services to host over 50 million players monthly on their online games. The decoupling of storage from server instances largely simplifies disaster recovery and sizing. Read the case study: https://aws.amazon.com/solutions/case-studies/6waves/.

- You can use Amazon S3 to host a static website. On a static website, individual webpages include static content. They might also contain client-side scripts. By contrast, a dynamic website relies on server-side processing, including server-side scripts such as PHP, JSP, or ASP.NET. Amazon S3 does not support server-side scripting.
    - For more information, see Hosting a Static Website on Amazon S3: https://docs.aws.amazon.com/AmazonS3/latest/dev/WebsiteHosting.html



Amazon S3 components

Bucket

https://s3-<aws-region>.amazonaws.com/<bucket name>/

Region-specific endpoint

Each bucket is regional.

Object

https://s3-<aws region>.amazonaws.com/<bucket name>/<object key>

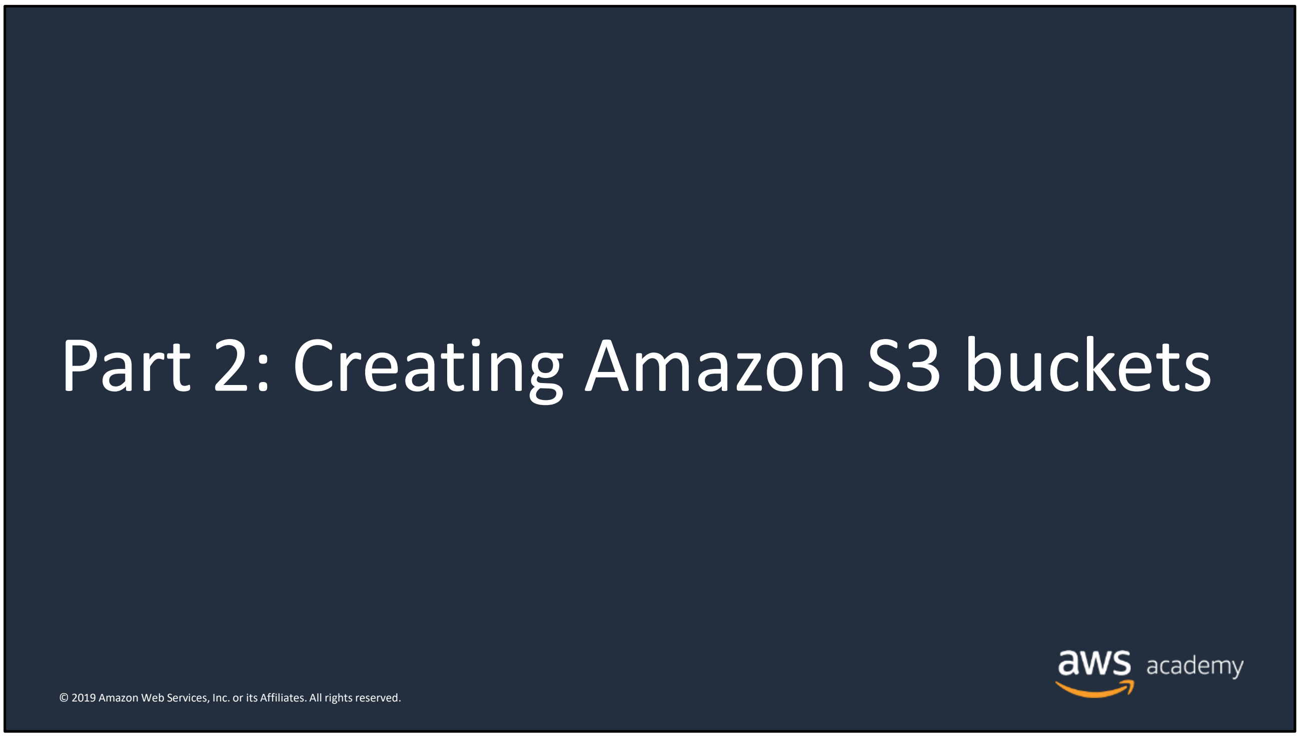
Object key example: preview.mp4

The basic components of Amazon S3 are buckets, objects, and keys:

- A *bucket* is a container for objects stored in Amazon S3. Buckets serve several purposes. They organize the Amazon S3 namespace at the highest level, they identify the account responsible for storage and data transfer charges, they play a role in access control, and they serve as the unit of aggregation for usage reporting.
- An *object* is the fundamental entity stored in Amazon S3. An object can be any kind of file such as text, video, photo, or another binary format. Objects consist of object data and metadata. The data portion is opaque to Amazon S3. The metadata is a set of name-value pairs that describe the object.
- An object *key* uniquely identifies the object in a bucket. Every object in a bucket has exactly one key. When you highlight a bucket in the Amazon S3 console, a list of objects in your bucket appears. These names are the object keys.

Note that *each bucket is regional*. You can choose the geographical region where Amazon S3 will store the buckets you create. Objects stored in an AWS Region never leave the Region unless you explicitly transfer them to another Region. The Region is indicated by the Region code. For example, the Region code for a bucket created in the US Oregon region is *us-west-2*.

For additional information on Amazon S3 core concepts, click the link to the Documentation: https://docs.aws.amazon.com/AmazonS3/latest/dev/Introduction.html#CoreConcepts



Part 2: Creating Amazon S3 buckets

Part 2: Creating Amazon S3 buckets

In this section, you will learn about creating Amazon S3 buckets.

Amazon S3 bucket names

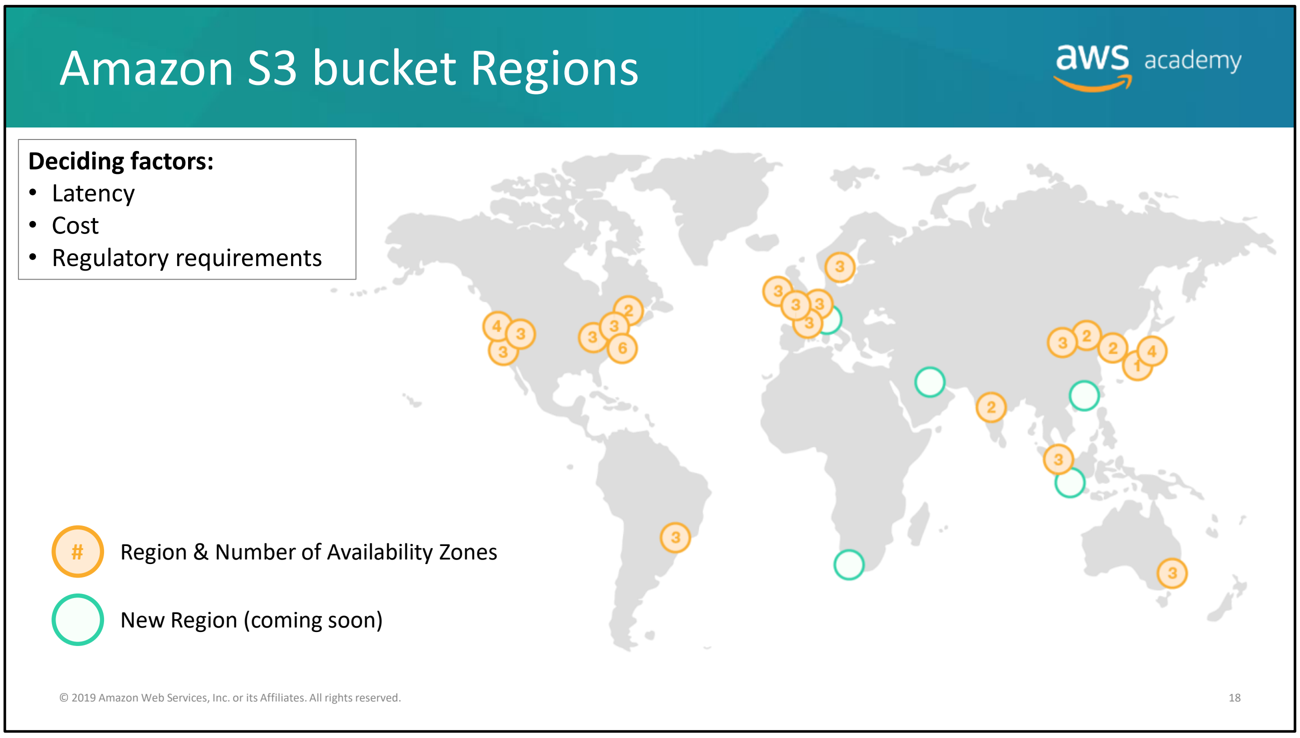
- Bucket names must be **globally unique**.
- Additional rules to follow when choosing bucket names:
  - Use 3 – 63 characters.
  - Use only lowercase letters, numbers, and hyphens (-).
  - Do not use uppercase characters or underscore (_) character.
  - Do not use a period (.) when using virtual hosted-style buckets with Secure Sockets Layer (SSL).

When you create an S3 bucket, you specify a name for it. There are certain rules that you must follow when choosing a bucket name:

- An Amazon S3 bucket name must be *globally unique*, and the namespace is shared by all AWS accounts. This means that after a bucket is created, the name of that bucket cannot be used by another AWS account in any AWS Region until the bucket is deleted. To ensure uniqueness, you might prefix the bucket name with the name of your organization.
- Additionally, bucket names:
  - Must be at least 3 and no more than 63 characters long.
  - Can contain lowercase letters, numbers, and hyphens.
  - Must not contain uppercase characters or underscores.
- Bucket names that have a period can cause certificate exceptions when they are accessed with HTTPS-based URLs. To support HTTPS access to your bucket, avoid using a period in the bucket name.

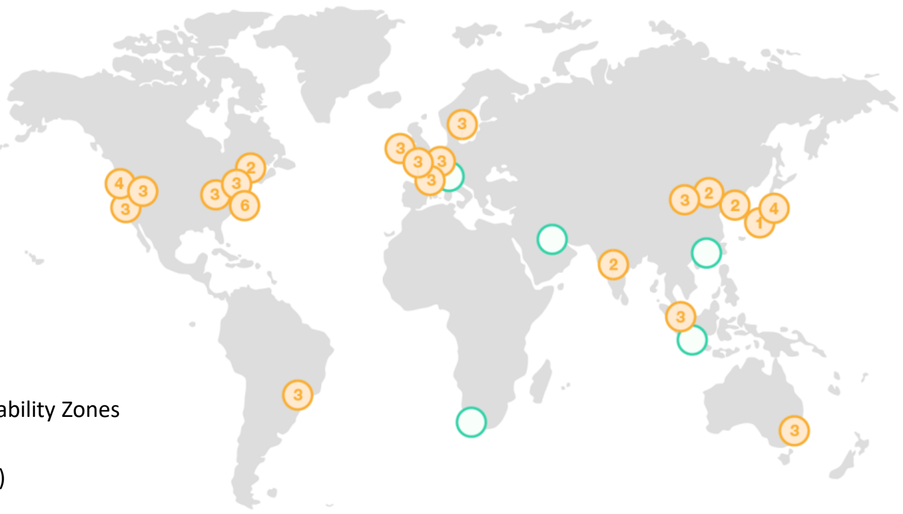
After you create an S3 bucket, you can't change the bucket name, so choose the name wisely.

For more information about S3 bucket restrictions, including rules for naming buckets, see the AWS Documentation:
https://docs.aws.amazon.com/AmazonS3/latest/dev/BucketRestrictions.html



Amazon S3 bucket Regions

Deciding factors:
- Latency
- Cost
- Regulatory requirements



● # Region & Number of Availability Zones

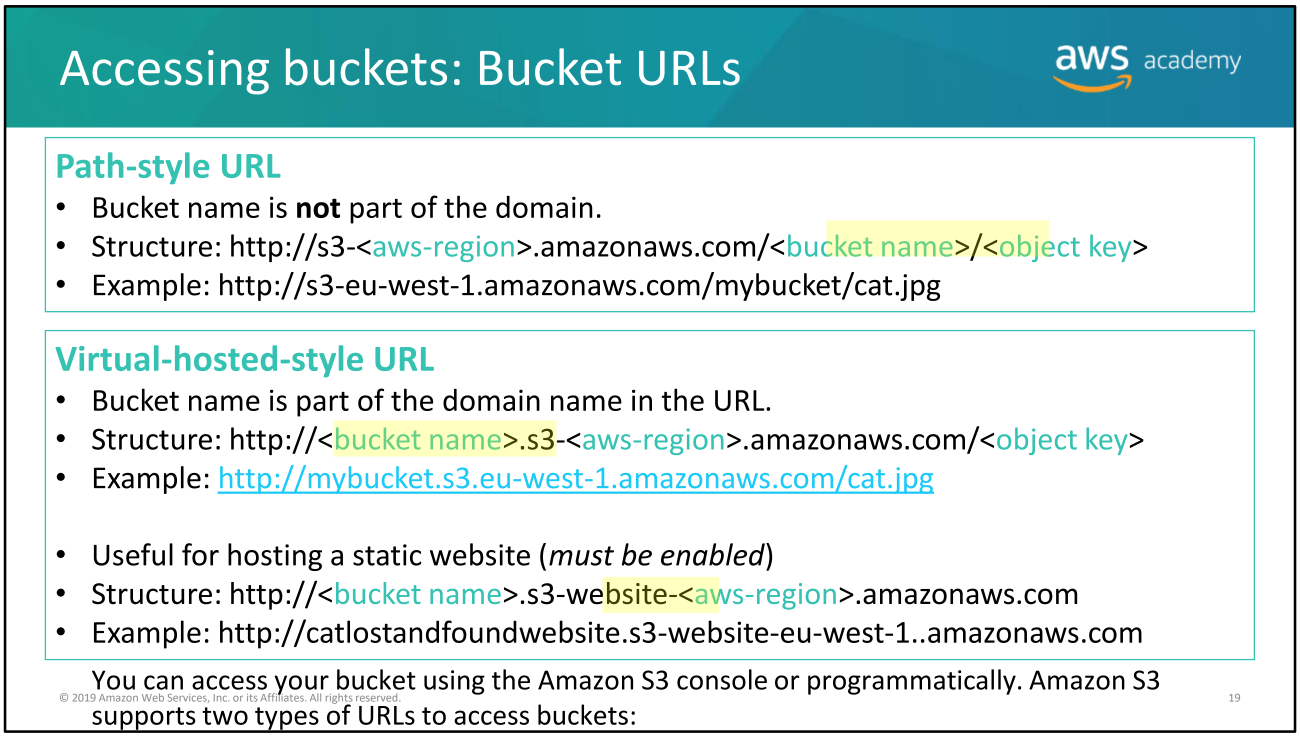
○ New Region (coming soon)

Amazon S3 creates buckets in a Region you specify. You might choose a Region to optimize latency, minimize costs, or address regulatory requirements.

After you have created a bucket, you can't change its Region.

For information about Amazon S3 Regions, see the AWS Documentation:
https://docs.aws.amazon.com/general/latest/gr/rande.html#s3_region

To see a map of the current AWS global infrastructure, see this AWS webpage:
https://aws.amazon.com/about-aws/global-infrastructure/



Accessing buckets: Bucket URLs

aws academy

Path-style URL

- Bucket name is **not** part of the domain.
- Structure: http://s3-<aws-region>.amazonaws.com/<bucket name>/<object key>
- Example: http://s3-eu-west-1.amazonaws.com/mybucket/cat.jpg

Virtual-hosted-style URL

- Bucket name is part of the domain name in the URL.
- Structure: http://<bucket name>.s3-<aws-region>.amazonaws.com/<object key>
- Example: http://mybucket.s3.eu-west-1.amazonaws.com/cat.jpg

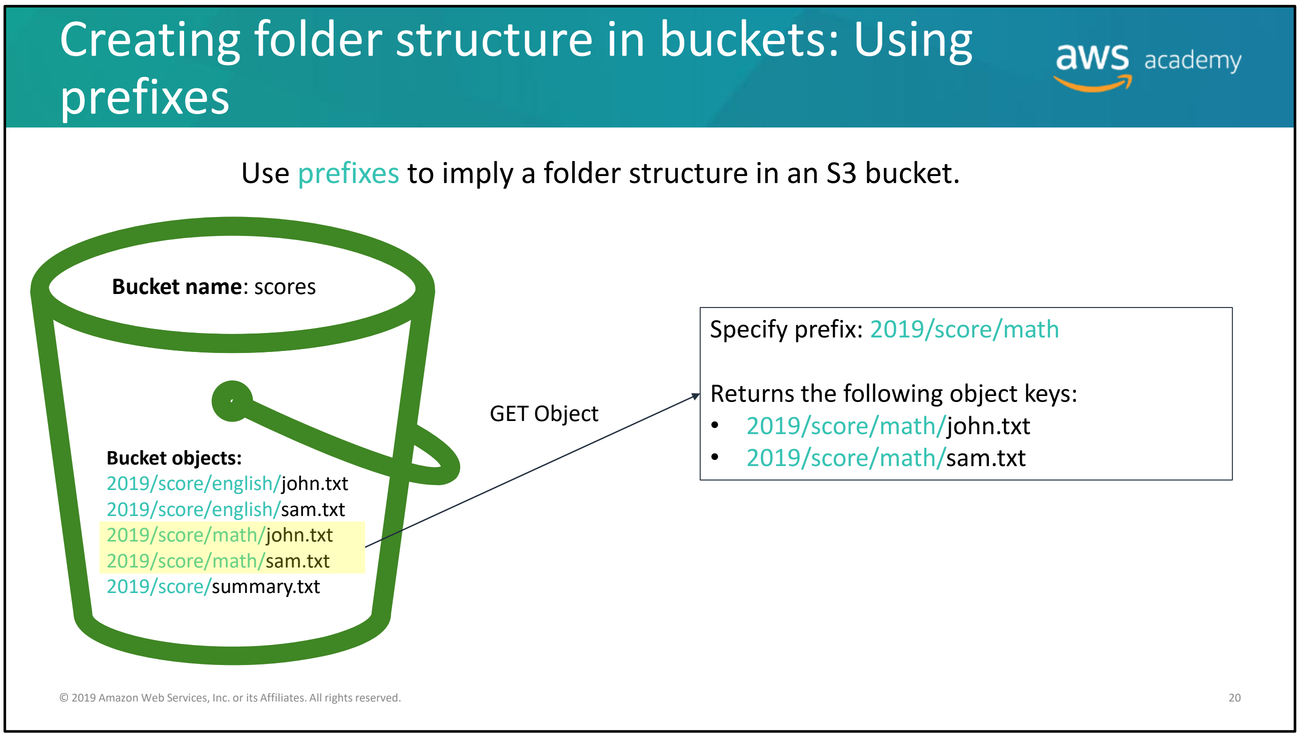
- Useful for hosting a static website (*must be enabled*)
- Structure: http://<bucket name>.s3-website-<aws-region>.amazonaws.com
- Example: http://catlostandfoundwebsite.s3-website-eu-west-1..amazonaws.com

You can access your bucket using the Amazon S3 console or programmatically. Amazon S3 supports two types of URLs to access buckets:

- In a *path-style URL*, the bucket name is not part of the domain. The example shows a path-style URL for a bucket called *mybucket* that resides in the eu-west-1 (Ireland) Region and contains an object called *cat.jpg*. Path-style URLs require you to specify a Region-specific endpoint when you attempt to access a bucket.
- In a *virtual-hosted–style URL*, the bucket name is part of the domain name in the URL. Virtual-hosted-style URLs are more user-friendly because the URLs begin with your custom bucket name.
- The virtual-hosted-style URL is useful if you are using your Amazon S3 bucket to host a static website. To host a static website, you configure an Amazon S3 bucket for website hosting, and then upload your website content to the bucket. This bucket must have public read access.

See the AWS Documentation for more information about:

- Accessing buckets: https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingBucket.html
- Virtual hosting of buckets: https://docs.aws.amazon.com/AmazonS3/latest/dev/VirtualHosting.html
- Hosting a static website: https://docs.aws.amazon.com/AmazonS3/latest/dev/WebsiteHosting.html



Creating folder structure in buckets: Using prefixes

Use prefixes to imply a folder structure in an S3 bucket.

Specify prefix: 2019/score/math

Returns the following object keys:

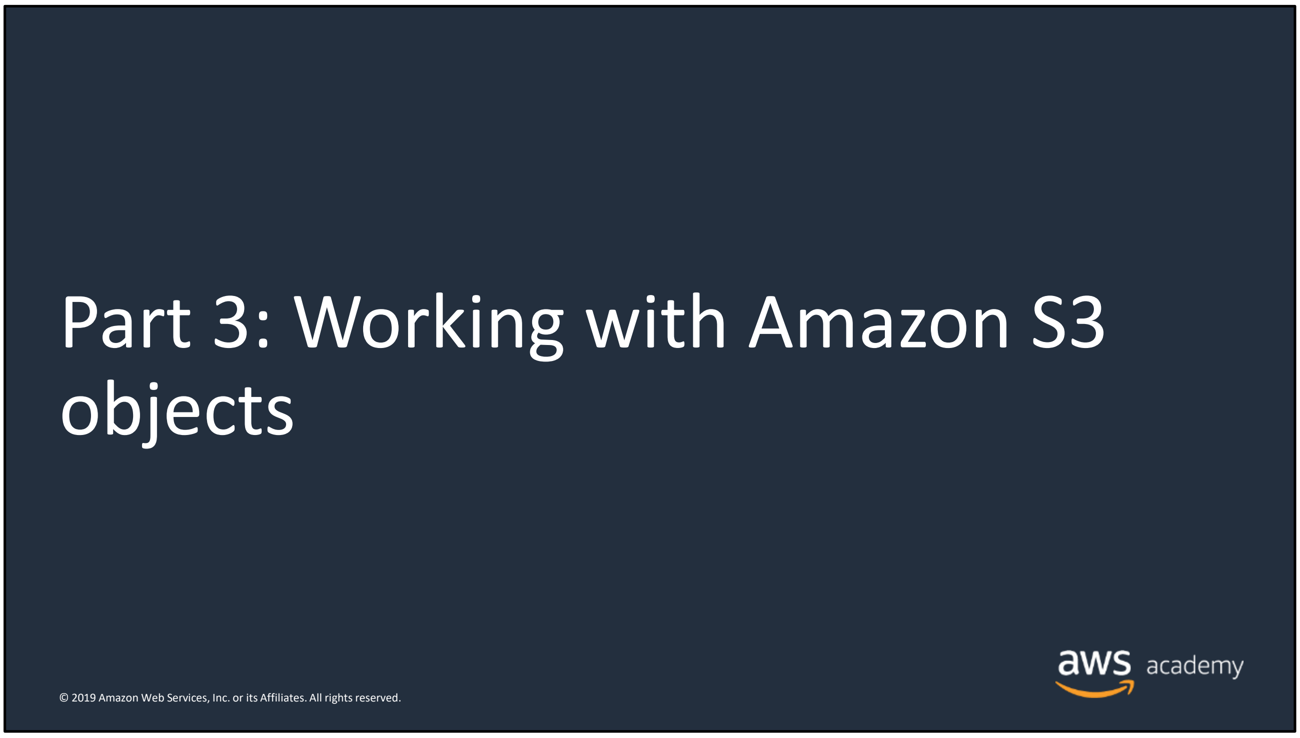
- 2019/score/math/john.txt
- 2019/score/math/sam.txt

In Amazon S3, buckets and objects are the primary resources, where objects are stored in buckets. Amazon S3 has a flat structure with no hierarchy like you would see in a file system. However, for the sake of organizational simplicity, Amazon S3 supports the folder concept as a means of grouping objects. Amazon S3 does this by using a shared name prefix for objects (that is, objects that have names that begin with a common string). When you create a folder, S3 console creates an object with the above name appended by suffix "/" and that object is displayed as a folder in the S3 console.

A prefix will limit results to only those keys that begin with the specified prefix. In the example, the bucket `scores` contains objects with student English and math scores for the year 2019. To list all the math score object keys for 2019, specify the prefix `2019/score/math`.

For more information about working with prefixes, see the AWS Documentation:

- https://docs.aws.amazon.com/AmazonS3/latest/user-guide/using-folders.html
- https://docs.aws.amazon.com/AmazonS3/latest/dev/ListingKeysHierarchy.html



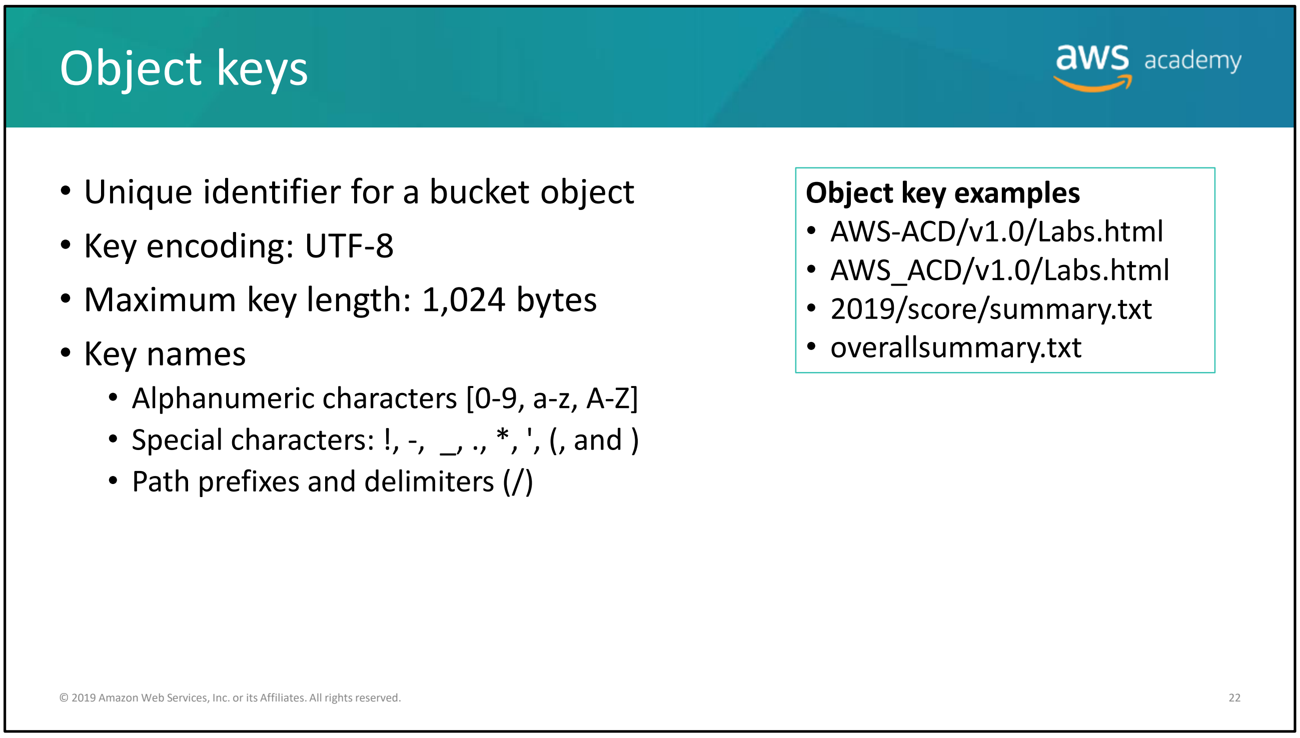
Part 3: Working with Amazon S3 objects

Part 3: Working with Amazon S3 objects

In this section, you will learn about object keys and metadata, versioning, and some of the common operations that you can use to work with Amazon S3 objects.

See the AWS Documentation for more information on the following:

- Operations on S3 objects: https://docs.aws.amazon.com/AmazonS3/latest/API/RESTObjectOps.html
- Operations on S3 buckets: https://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketOps.html



Object keys

aws academy

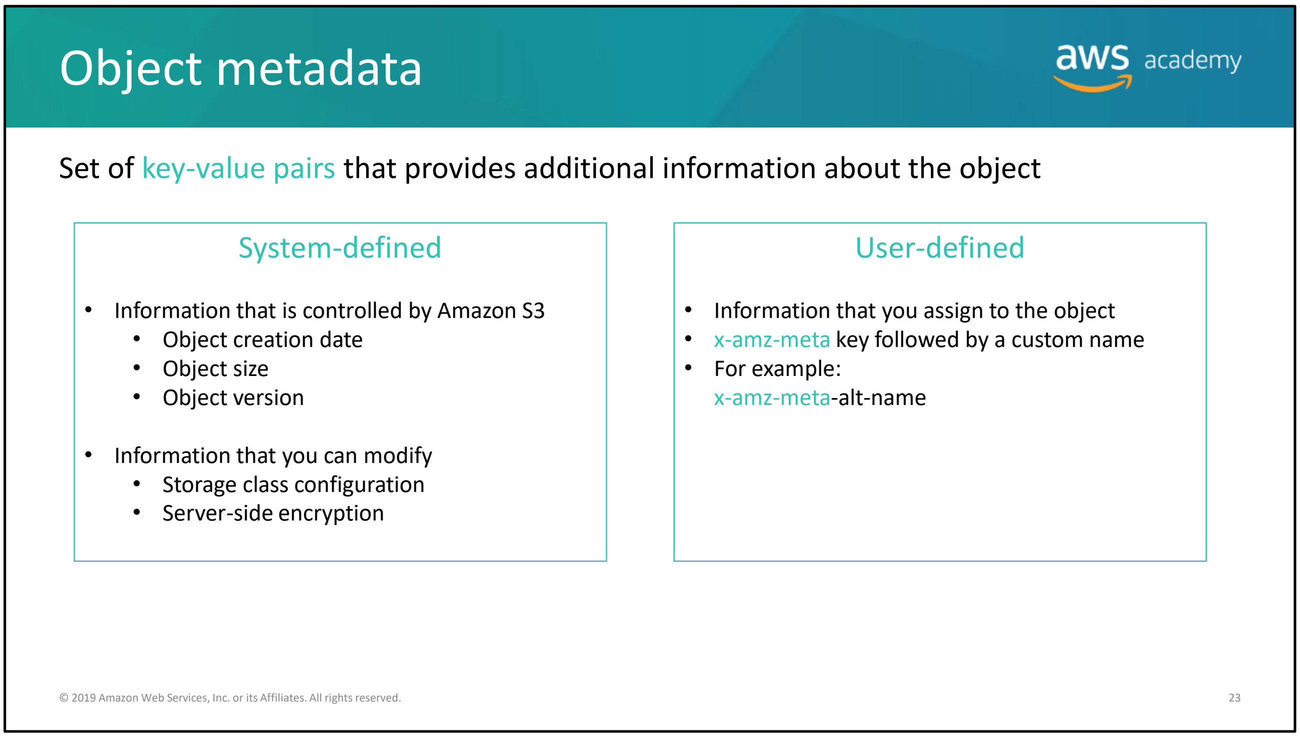
- Unique identifier for a bucket object
- Key encoding: UTF-8
- Maximum key length: 1,024 bytes
- Key names
  - Alphanumeric characters [0-9, a-z, A-Z]
  - Special characters: !, -, _, ., *, ', (, and)
  - Path prefixes and delimiters (/)

Object key examples
- AWS-ACD/v1.0/Labs.html
- AWS_ACD/v1.0/Labs.html
- 2019/score/summary.txt
- overallsummary.txt

An *object* is the fundamental entity stored in Amazon S3. When you create an object, you specify the key name.

Object keys:
- Uniquely identify bucket objects
- Are encoded using UTF-8 encoding.
- Have a maximum length of 1,024 bytes.
- Can contain alphanumeric characters and some special characters (such as an exclamation mark, hyphen, underscore, period, asterisk, single quote, and forward slash). They can also include path prefixes and delimiters to logically group your objects.

For more information about object keys, see
https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingMetadata.html



Object metadata

Set of key-value pairs that provides additional information about the object

System-defined

- Information that is controlled by Amazon S3
  - Object creation date
  - Object size
  - Object version
- Information that you can modify
  - Storage class configuration
  - Server-side encryption

User-defined

- Information that you assign to the object
- x-amz-meta key followed by a custom name
- For example:
  x-amz-meta-alt-name

Objects consist of object data and metadata. Object metadata is a set of key-value pairs that provides additional information about the object. There are two kinds of object metadata:

- *System-defined metadata* – For each object stored in a bucket, Amazon S3 maintains a set of system metadata. Amazon S3 processes this system metadata as needed. System-defined metadata includes information such as object creation date, object size, and object version. Some types of system-defined metadata are controlled by Amazon S3, e.g., object creation date, size, and version. You can update some types of system-defined metadata, e.g., storage configuration and server-side encryption.

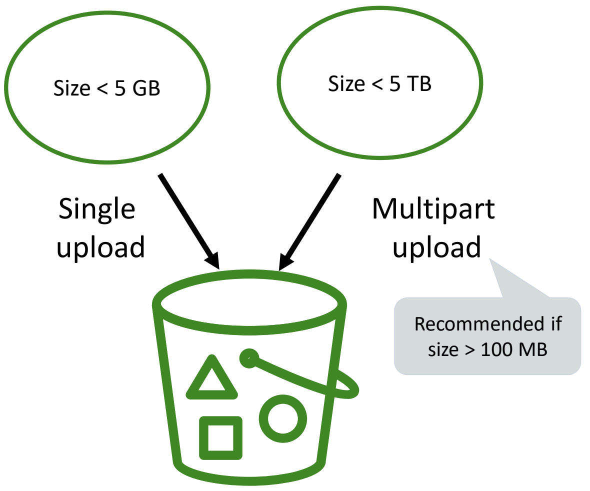
- *User-defined metadata* – You can also assign metadata to an object. You can assign user-defined metadata to an object when you upload the object or after the object has been uploaded. User-defined metadata is stored with the object and is returned when you download the object. Amazon S3 does not process user-defined metadata. User-defined metadata must begin with the prefix "x-amz-meta-", otherwise Amazon S3 will not set the key value pair as you define it. To define custom metadata, add a custom name to the *x-amz-meta-* key. This creates a custom key. For example, if you add the custom name alt-name, the metadata key would be x-amz-meta-alt-name.

For more information about object metadata, see the AWS Documentation:

- https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingMetadata.html#object-metadata
- https://docs.aws.amazon.com/AmazonS3/latest/user-guide/add-object-metadata.html

PUT Object

- Upload entire objects to a bucket
- Should use single upload for objects up to 5 GB in a single PUT operation
- Must use multipart upload for larger objects up to 5 TB (max individual object size)
- Multipart upload resumes where you left off.



The *PUT Object operation* allows you to add an object to an S3 bucket. You must have WRITE permissions on a bucket to add an object to it. Amazon S3 never adds partial objects; if you receive a success response, Amazon S3 added the entire object to the bucket.

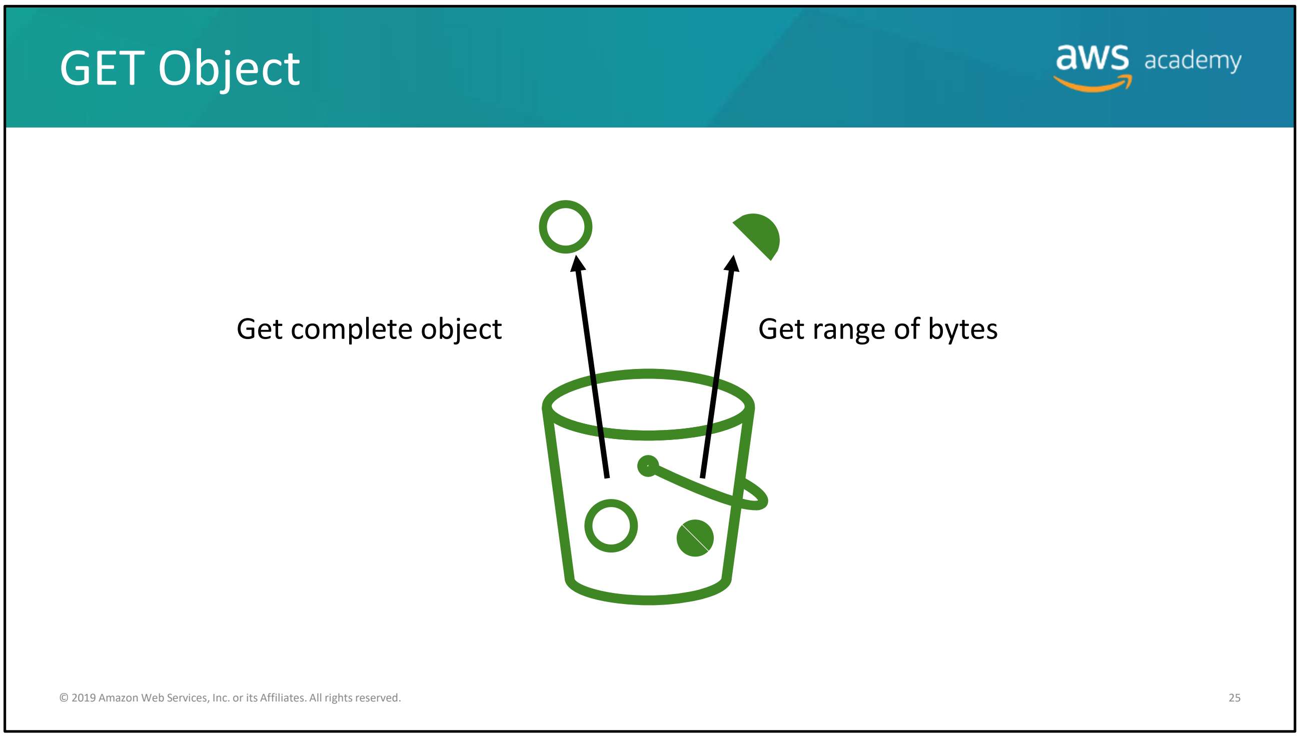
- *Single upload for objects up to 5 GB* – You can upload or copy objects of up to 5 GB in a single PUT operation.
- *Multipart upload for objects up to 5 TB* – For larger objects up to 5 TB, you should use the multipart upload application programming interface (API). Multipart upload allows you to upload a single object as a set of parts. Each part is a contiguous portion of the object's data. You can upload these object parts independently and in any order. If transmission of any part fails, you can retransmit that part without affecting other parts. After all parts of your object are uploaded, Amazon S3 assembles these parts and creates the object.

In general, **when your object size reaches 100 MB, you should consider using multipart uploads** instead of uploading the object in a single operation. To understand why, imagine an upload getting to just under 5 GB and your internet connection died. You would have to start the upload over. Whereas with multipart upload, the upload would resume where it left off when you are back online.

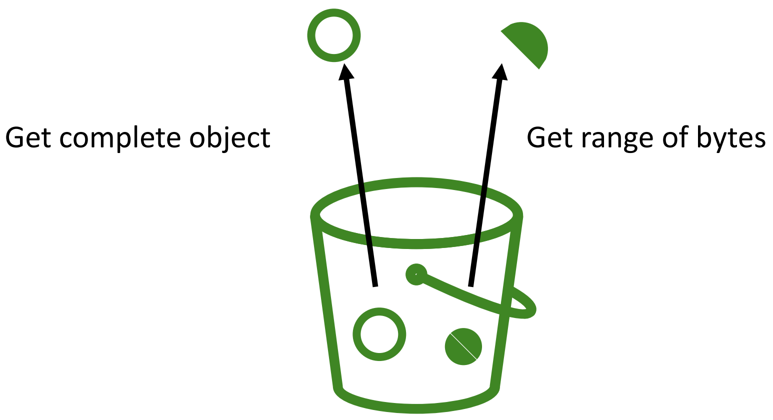
You can also use the *PUT Object - Copy* operation to create copies of an object that is already stored in Amazon S3.

See the AWS Documentation for more information on:
- PUT Object operation: https://docs.aws.amazon.com/AmazonS3/latest/API/RESTObjectPUT.html
- Put Object – Copy operation: https://docs.aws.amazon.com/AmazonS3/latest/API/RESTObjectCOPY.html

- Multipart upload: https://docs.aws.amazon.com/AmazonS3/latest/dev/mpuoverview.html



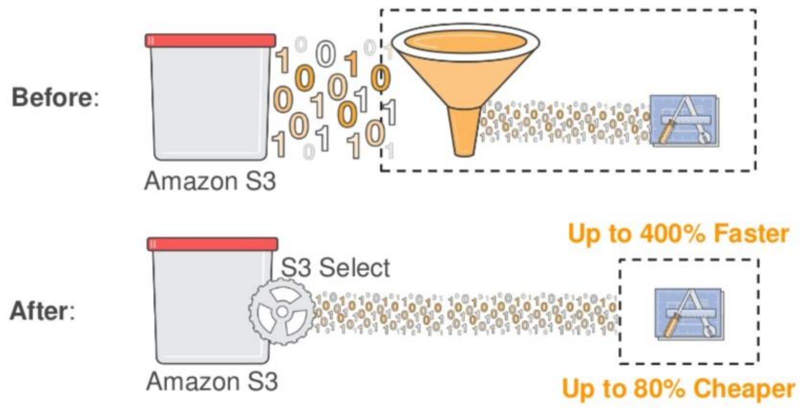
GET Object



You use the *GET Object operation* to retrieve objects from Amazon S3. To use GET, you must have READ access to the object. You can retrieve a complete object in a single GET request. You can also retrieve an object in parts by specifying the range of bytes needed. This is useful in scenarios where network connectivity is poor, or if your application can only process subsets of object data.

For more information on the GET Object operation, see the Amazon S3 API Reference: https://docs.aws.amazon.com/AmazonS3/latest/API/RESTObjectGET.html

SELECT Object



You can use the *SELECT Object Content operation* to filter the contents of an Amazon S3 object based on a simple structured query language (SQL) statement. You must send a SQL expression in the request, and you must also specify a data serialization format for the S3 object, such as JavaScript Object Notation (JSON), comma-separated values (CSV), or Apache Parquet. Amazon S3 uses this format to parse object data into records, and it returns only records that match the specified SQL expression. You must have `s3:GetObject` permission for this operation.

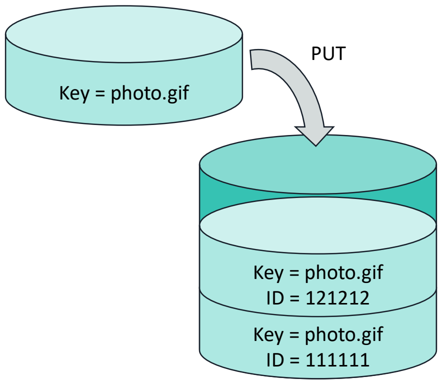
For example, you can use the select request to retrieve all records from an S3 object by using *`<Expression>Select * from S3Object</Expression>`* or to identify the column headers in your file by name using
`SELECT s.Id, s.FirstName, s.SSN FROM S3Object s`

Because your applications do not have to use compute resources to scan and filter the data from an object, you can potentially increase query performance by up to 400 percent and reduce query costs by as much as 80 percent.

For more information on the SELECT Object Content operation, see the Amazon S3 API Reference: https://docs.aws.amazon.com/AmazonS3/latest/API/RESTObjectSELECTContent.html

Versioning

- Way of keeping multiple variants of an object in the same bucket.
- Use to recover from unintended user actions and application failures.
- In versioning-enabled S3 buckets, each object has a version ID.
- Once enabled, you can only *suspend* versioning (you cannot disable it)
- Object locking is supported on versioned buckets.



S3 bucket with versioning enabled

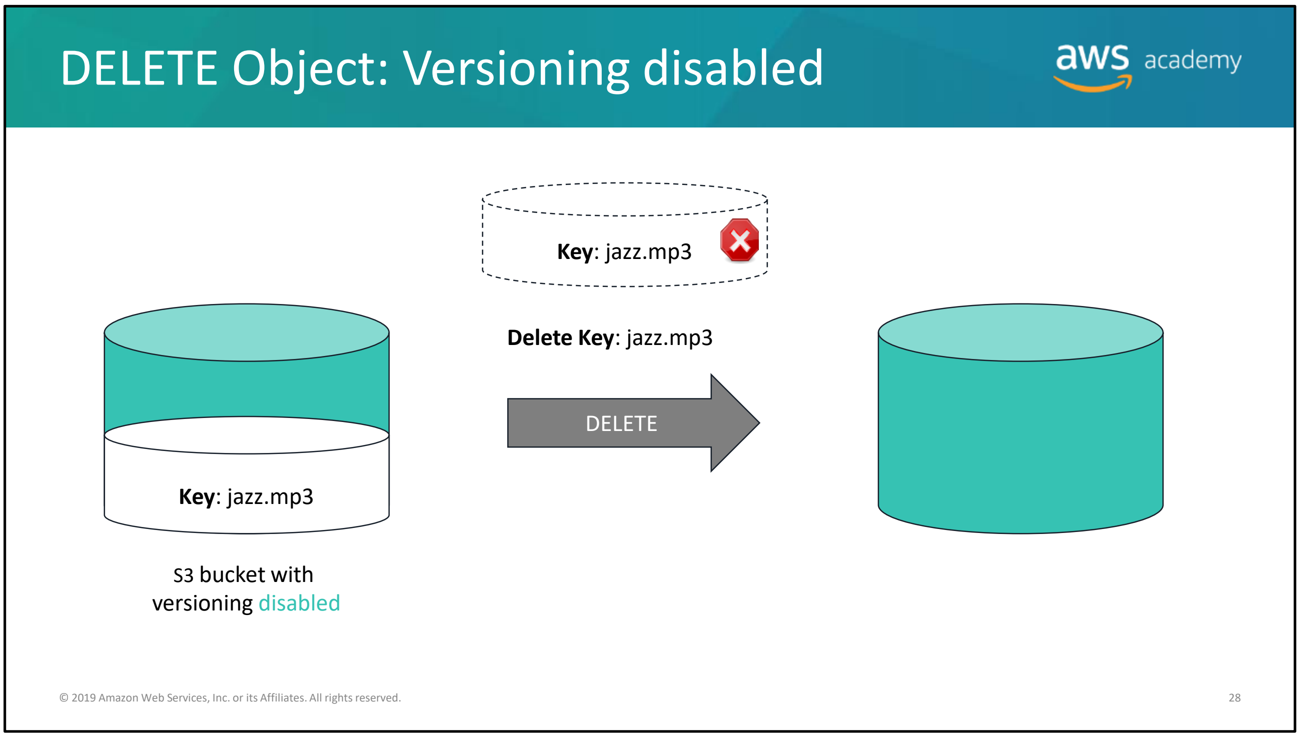
Versioning is another feature of Amazon S3 that you can use to protect your data. Versioning is a means of keeping multiple variants of an object in the same bucket. In one bucket, you can have two objects that have the same key but different version IDs, e.g., photo.gif (version 111111) and photo.gif (version 121212).

You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions (such as overwrites and deletions) and application failures. You can also use versioning to archive objects so you have access to previous versions.

By default, versioning is disabled in S3 buckets. Once you enable versioning, you can only suspend it (you cannot disable it). Note that normal Amazon S3 rates apply for every version of an object that is stored or requested. (For more information, see *How am I charged for using Versioning*: https://aws.amazon.com/s3/faqs/)

Amazon S3 Object Lock is another option you can use to prevent data from being changed, overwritten, or deleted.

See the AWS Documentation for more information on:
- How to use versioning: https://docs.aws.amazon.com/AmazonS3/latest/dev/Versioning.html
- Amazon S3 Object Lock: https://docs.aws.amazon.com/AmazonS3/latest/dev/object-lock.html

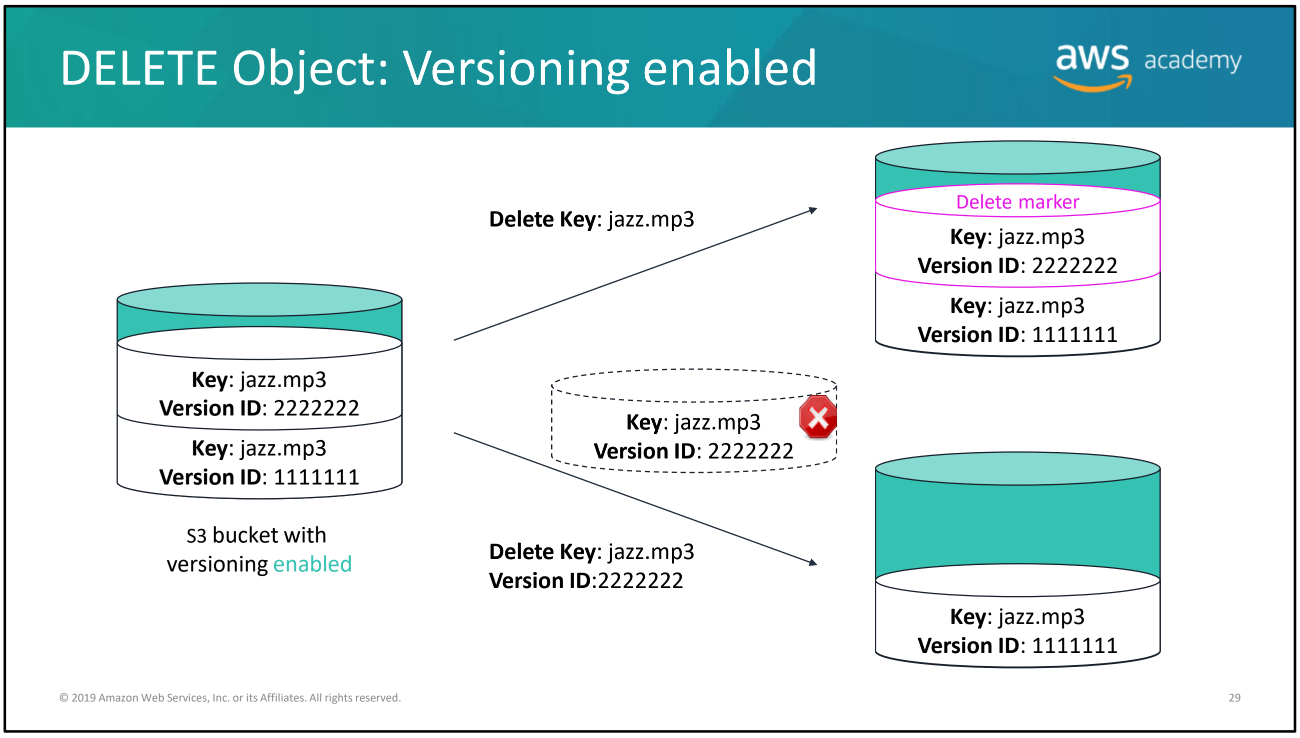


DELETE Object: Versioning disabled

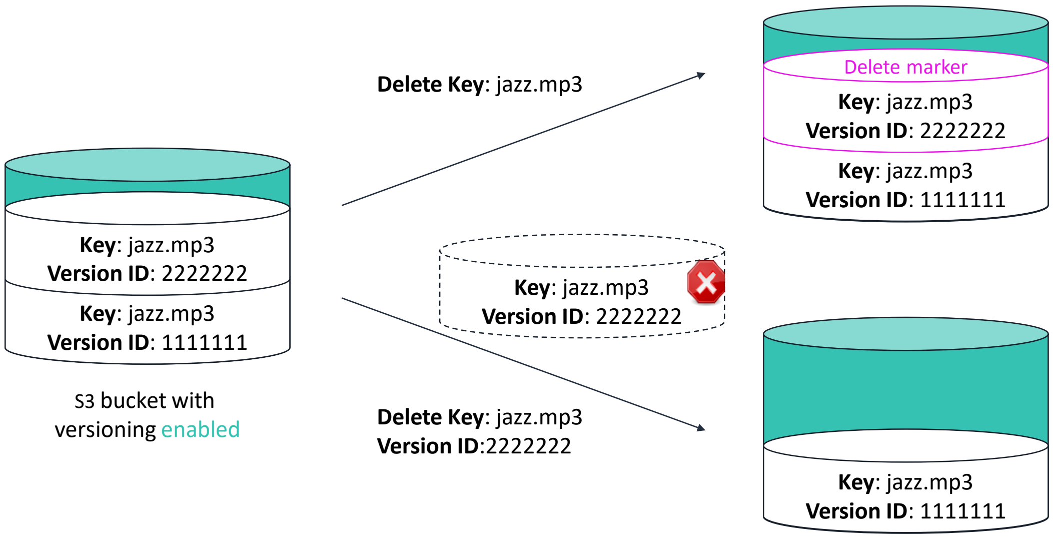
With the *DELETE Object operation*, you can delete a single object or multiple objects.

In a bucket that is not versioning-enabled, you can permanently delete an object by specifying the key of the object that you want to delete.

For more information about deleting object versions, see the AWS Documentation: https://docs.aws.amazon.com/AmazonS3/latest/dev/DeletingObjectVersions.html



DELETE Object: Versioning enabled



In a bucket that is versioning-enabled, you can permanently delete an object by invoking a delete request with a delete key and version ID. You must delete each individual version to completely remove an object.

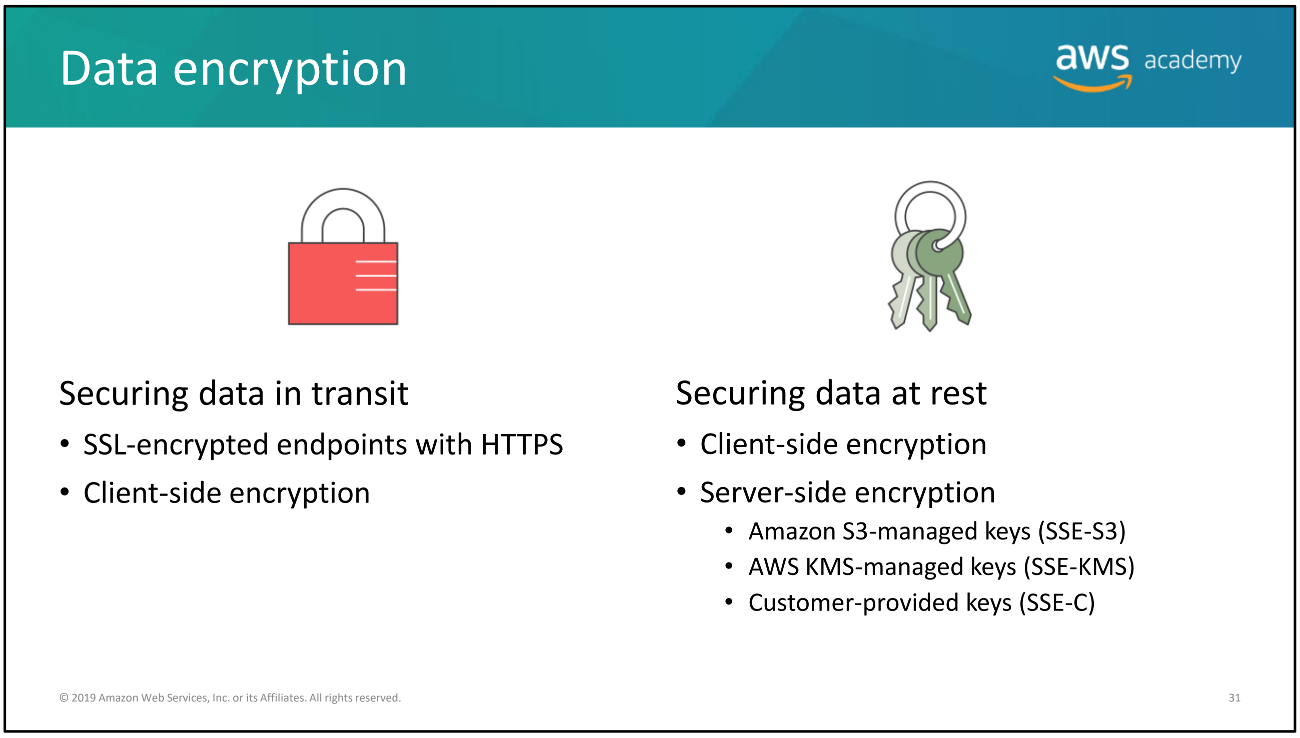
If you specify only a delete key, Amazon S3 adds a delete marker that becomes the current version of the object. If you try to retrieve an object that has a delete marker, Amazon S3 returns a 404 Not Found error.

For more information about deleting object versions, see the AWS Documentation:
https://docs.aws.amazon.com/AmazonS3/latest/dev/DeletingObjectVersions.html

Part 4: Protecting data and managing access to Amazon S3 resources

Part 4: Protecting data and managing access to Amazon S3 resources

In this section, you will learn about how to protect data in Amazon S3 and manage access control.



Data encryption

aws academy

Securing data in transit

- SSL-encrypted endpoints with HTTPS
- Client-side encryption

Securing data at rest

- Client-side encryption
- Server-side encryption
  - Amazon S3-managed keys (SSE-S3)
  - AWS KMS-managed keys (SSE-KMS)
  - Customer-provided keys (SSE-C)

Data protection refers to protecting data while *in transit* (as it travels to and from Amazon S3) and *at rest* (while it is stored on disks in Amazon S3 data centers).

You can protect data in transit by using SSL or by using client-side encryption.

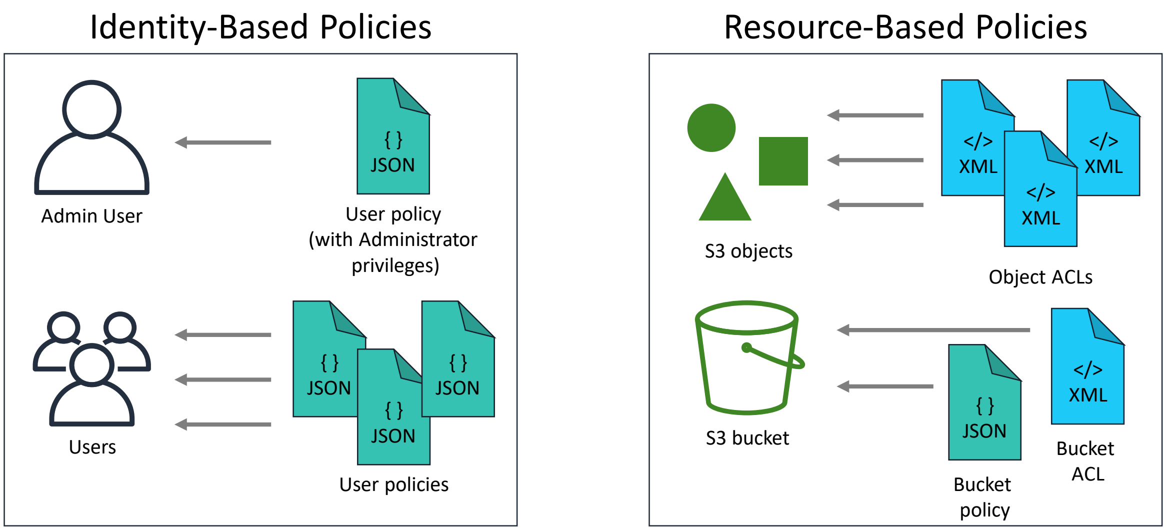
To protect data at rest in Amazon S3, you have the following options:

- With *client-side encryption*, you can encrypt data on the client-side and upload the encrypted data to Amazon S3. In this case, you manage the encryption process, the encryption keys, and related tools. Click the link for information about which software development kits (SDKs) support client-side encryption – https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingClientSideEncryption.html.
- With *server-side encryption*, Amazon S3 encrypts your data at the object level as it writes it to disks in its data centers, and then decrypts it for you when you access it. You have three mutually exclusive options depending on how you choose to manage the encryption keys:
  - *Server-Side Encryption with Amazon S3-Managed Keys (SSE-S3)* – Each object is encrypted with a unique key that employs strong multi-factor encryption. It also encrypts the key itself with a master key that it regularly rotates. Amazon S3 server-side encryption uses one of the strongest block ciphers available, 256-bit Advanced Encryption Standard (AES-256), to encrypt your data.
  - *Server-Side Encryption with AWS KMS-Managed Keys (SSE-KMS)* – This is similar to SSE-S3. However, it includes some other benefits and has some additional charges. There are separate permissions for the use of an envelope key (that is, a key that protects your data's encryption key), which provides added protection against unauthorized access of your Amazon S3 objects. SSE-KMS also provides you with an audit trail of when your key was used and by whom. Additionally, you have the option of creating and managing encryption keys yourself, or you can use a default key that is unique to you, the service you're using, and the Region you're working in.

- *Server-Side Encryption with Customer-Provided Keys (SSE-C)* – You manage the encryption keys. Amazon S3 manages the encryption as it writes to disks, and it also manages the decryption when you access your objects.

For more information on server-side encryption with Amazon S3, see the AWS Documentation: https://docs.aws.amazon.com/AmazonS3/latest/dev/serv-side-encryption.html

User policies and resource-based policies

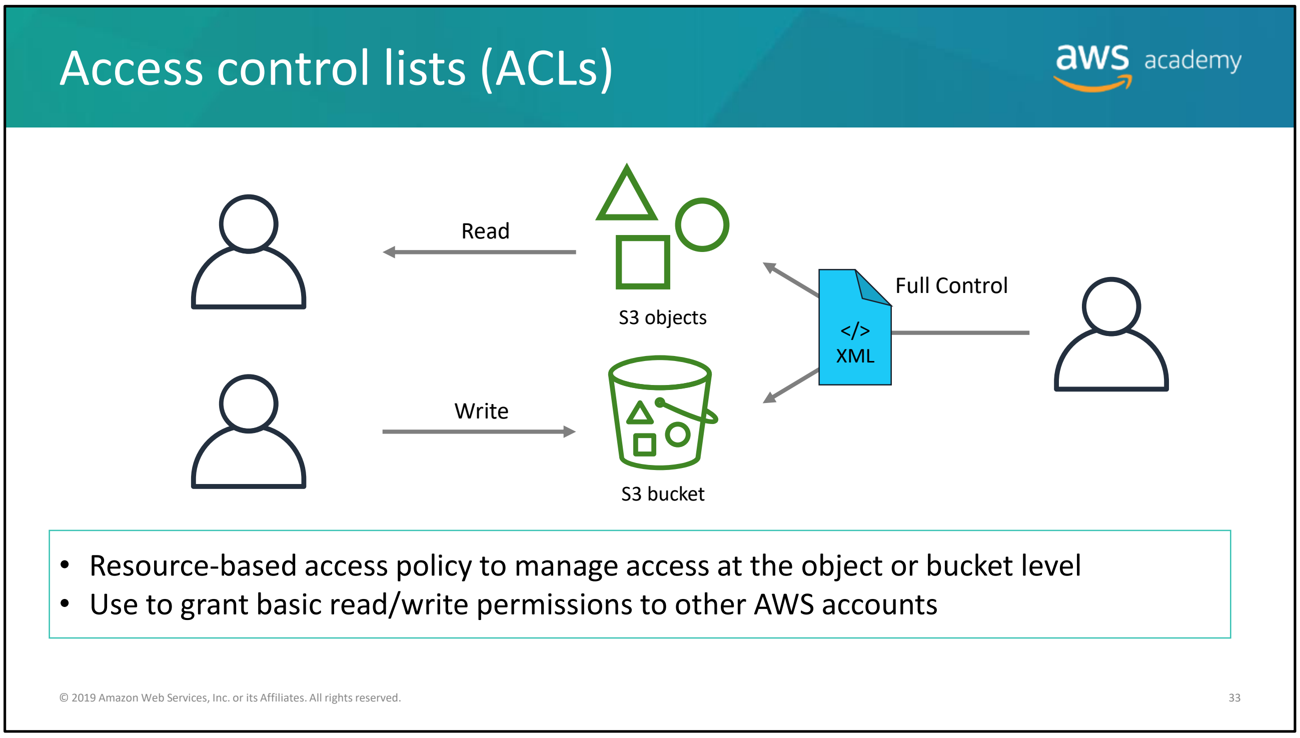


You can manage access to your Amazon S3 resources by writing *access policies* that grant others permission to perform the resource operations. For example, you can grant PUT Object permission to a user in an AWS account so the user can upload objects to your bucket. By default, all Amazon S3 resources, such as buckets and objects, are private. Only the AWS account that created the resources can access them.

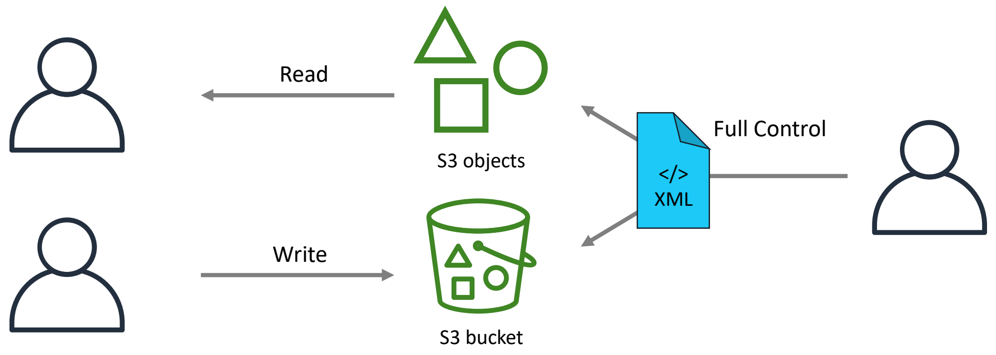
There are two types of access policies:

- *Identity-based policies* – You attach these policies to AWS Identity and Access Management (IAM) users, groups, and roles in your account to grant them access to AWS resources.
- *Resource-based policies* – You attach these policies to your Amazon S3 resources. Bucket policies and access control lists (ACLs) are resource-based policies.

For an overview of managing access to Amazon S3 resources, see the AWS Documentation: https://docs.aws.amazon.com/AmazonS3/latest/dev/access-control-overview.html



Access control lists (ACLs)



- Resource-based access policy to manage access at the object or bucket level
- Use to grant basic read/write permissions to other AWS accounts

Access control lists (ACLs) are one of the resource-based access policy options that you can use to manage access to your buckets and objects. You can use ACLs to grant basic read/write permissions to other AWS accounts.

There are limits to managing permissions using ACLs. For example, you can grant permissions only to other AWS accounts; you cannot grant permissions to users in your account. You cannot grant conditional permissions, nor can you explicitly deny permissions.

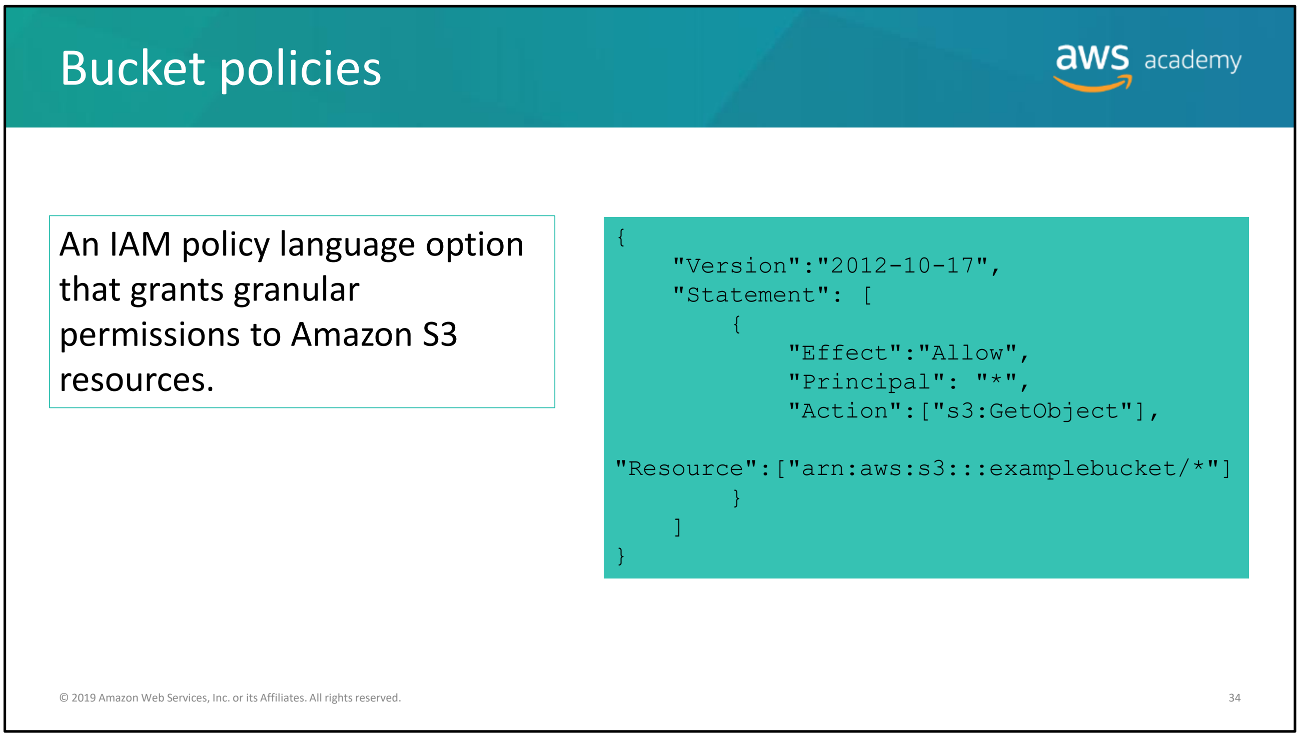
ACLs are suitable for specific scenarios. For example, if a bucket owner allows other AWS accounts to upload objects, permissions to these objects can only be managed using object ACLs *by the AWS account that owns the object*.

You use an *object ACL* for the following scenarios:

- An object ACL is the only way to manage access to objects that are not owned by the bucket owner – An AWS account that owns the bucket can grant another AWS account permission to upload objects. The bucket owner does not own these objects. The AWS account that created the object must grant permissions using object ACLs.
- Permissions vary by object, and you need to manage permissions at the object level – You can write a single policy statement that grants an AWS account read permission on millions of objects with a specific key name prefix.
- Object ACLs control only object-level permissions – The entire bucket has a single bucket policy, but object ACLs are specified per object.

The only recommended use case for the *bucket ACL* is to grant write permission to the Amazon S3 Log Delivery group to write access log objects to your bucket.

For more information on managing access with ACLs, see the AWS Documentation: https://docs.aws.amazon.com/AmazonS3/latest/dev/S3_ACLs_UsingACLs.html



Bucket policies

An IAM policy language option that grants granular permissions to Amazon S3 resources.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect":"Allow",
            "Principal": "*",
            "Action":["s3:GetObject"],

"Resource":["arn:aws:s3:::examplebucket/*"]
        }
    ]
}
```

A *bucket policy* is another type of resource-based IAM policy. You can add a bucket policy to an S3 bucket to grant other AWS accounts or IAM users permissions for the bucket and the objects in it. Object permissions apply only to the objects that the bucket owner creates. Bucket policies supplement—and in many cases, replace—ACL-based access policies.

The example bucket policy grants anonymous read permission on all objects in a bucket. The bucket policy has one statement, which allows the `s3:GetObject` action (read permission) on objects in a bucket named examplebucket. By specifying the principal with a wild card (*), the policy grants anonymous access.

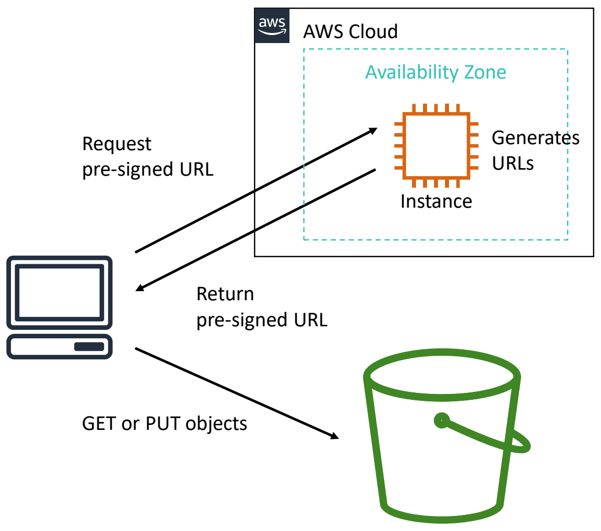
You use a bucket policy when you want to manage cross-account permissions for all Amazon S3 permissions: https://docs.aws.amazon.com/AmazonS3/latest/dev/access-policy-alternatives-guidelines.html

For more information on when to use bucket policies, see the Guidelines for Using the Available Access Policy Options.

Pre-signed URLs

aws academy

- Provide access to PUT or GET objects without opening permissions to do anything else.
- Use permissions of the user who creates the URL.
- Provide security credentials, a bucket name, an object key, HTTP method, and expiration date and time.
- Are only valid until expiration time (maximum 1 week).



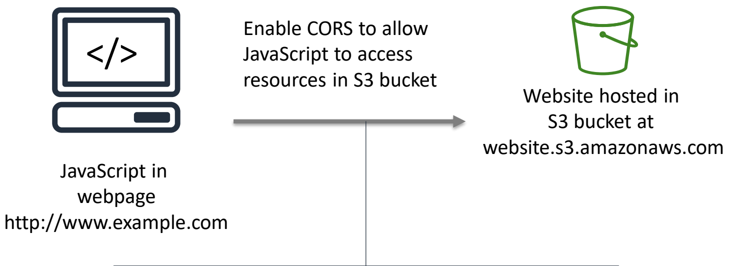
All objects and buckets are private by default. *Pre-signed URLs* are useful if you want your user to be able to upload a specific object to your bucket without needing AWS security credentials or permissions. When you create a pre-signed URL, you must provide your security credentials, bucket name, an object key, an HTTP method (PUT for uploading objects, GET for retrieving objects), and an expiration date and time. The pre-signed URLs are valid only for the specified duration.

Share the pre-signed URL with users who need to access your S3 bucket to put or retrieve objects.

For more information on pre-signed URLs, see Uploading Objects Using Pre-Signed URLs: https://docs.aws.amazon.com/AmazonS3/latest/dev/PresignedUrlUploadObject.html

Cross-origin resource sharing (CORS)

Cross-origin resource sharing (CORS) defines a way for client web applications that are loaded in one domain to interact with resources in a different domain.


Enable CORS to allow JavaScript to access resources in S3 bucket
JavaScript in webpage http://www.example.com
Website hosted in S3 bucket at website.s3.amazonaws.com

```
<CORSConfiguration>
 <CORSRule>
   <AllowedOrigin>*</AllowedOrigin>
   <AllowedMethod>GET</AllowedMethod>
 </CORSRule>
</CORSConfiguration>
```

Sometimes websites have to load resources from other places. Cross-origin resource sharing (CORS) defines a way for client web applications that are loaded in one domain to interact with resources in a different domain.

Consider the following examples of when to enable CORS:

- JavaScript in one domain's webpage (http://www.example.com) wants to use resources from your S3 bucket by using the endpoint *website.s3.amazonaws.com*. The browser will allow such cross-domain access only if CORS is enabled on your bucket.
- You want to host a web font in your S3 bucket. A webpage in a different domain might try to use this web font. Before the browser loads this webpage, it will perform a CORS check to make sure that the domain where the page is being loaded is allowed to access resources from your S3 bucket.

With CORS support in Amazon S3, you can build web applications with Amazon S3 and selectively allow cross-origin access to your Amazon S3 resources.

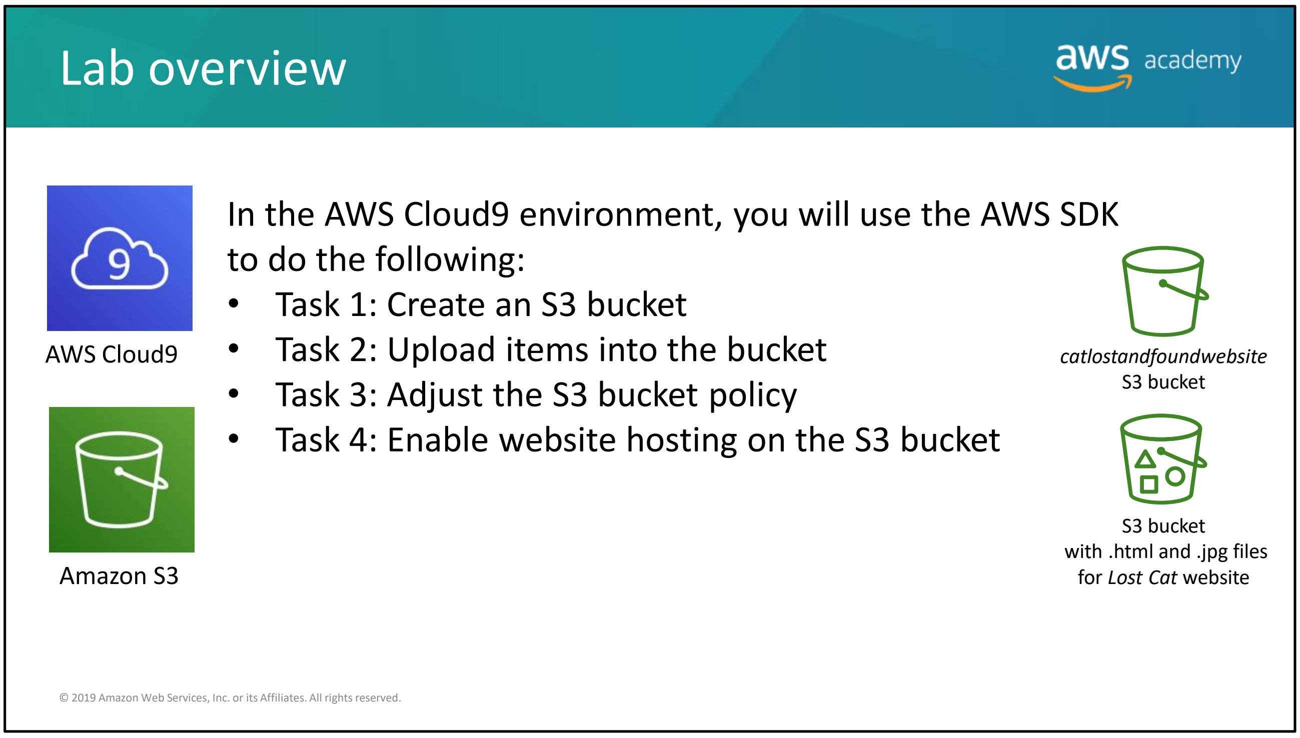
To enable CORS, create a CORS configuration XML file with rules that identify the origins you will allow to access your bucket, the operations (HTTP methods) that you will support for each origin, and other operation-specific information. You can add up to 100 rules to the configuration. You can apply the CORS configuration to the S3 bucket by using the AWS SDK.

See the AWS Documentation for more information about:

- CORS: https://docs.aws.amazon.com/AmazonS3/latest/dev/cors.html
- How to enable CORS:

https://docs.aws.amazon.com/AmazonS3/latest/dev/ManageCorsUsing.html

Lab: Developing with Amazon S3 using the AWS SDK



Lab overview

aws academy

In the AWS Cloud9 environment, you will use the AWS SDK to do the following:

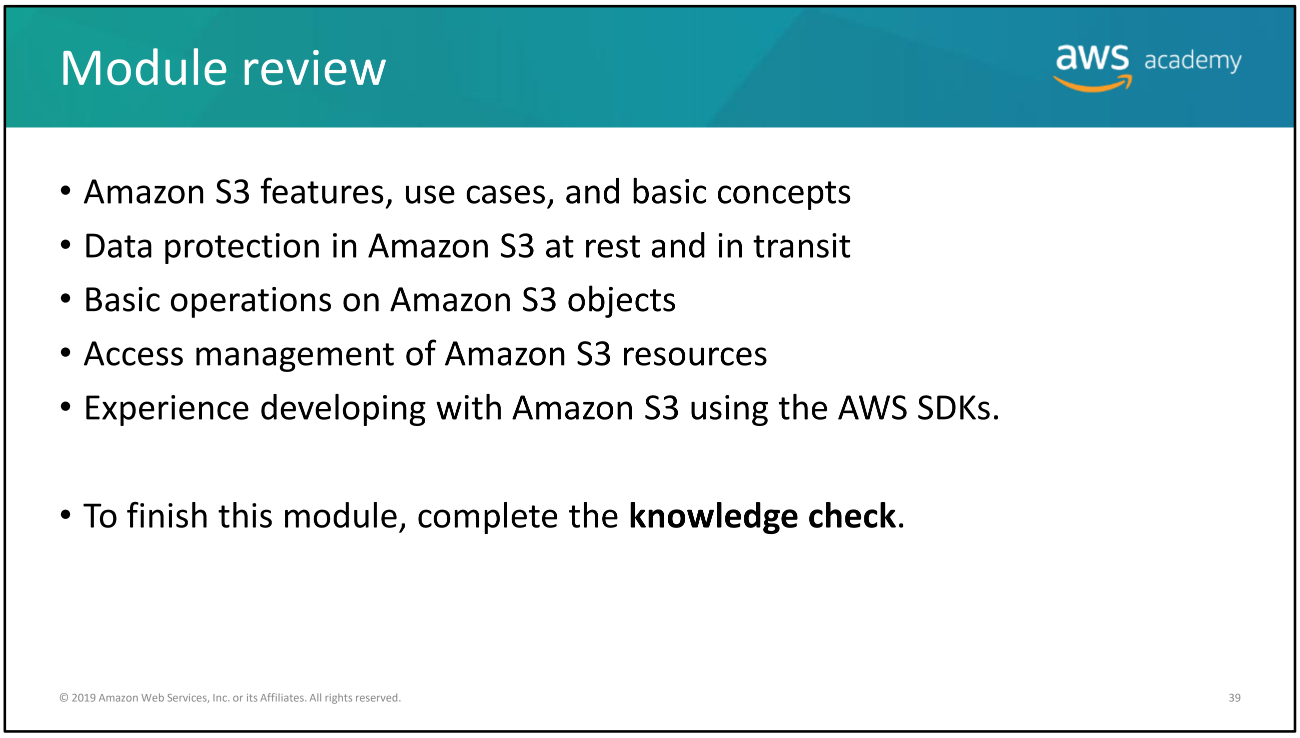
- Task 1: Create an S3 bucket
- Task 2: Upload items into the bucket
- Task 3: Adjust the S3 bucket policy
- Task 4: Enable website hosting on the S3 bucket

AWS Cloud9
Amazon S3
catlostandfoundwebsite S3 bucket
S3 bucket with .html and .jpg files for *Lost Cat* website

In this lab, you will use the AWS SDKs to create an S3 bucket and use the bucket to store .html and .jpg files for a Lost Cat static website that you're creating. You will also adjust the S3 bucket policy to make the bucket publicly viewable and configure the bucket to host your static website.

After completing this lab, you will be able to use the AWS SDK to do the following:

- Create an S3 bucket
- Upload items to the bucket
- Make the website publicly viewable
- Enable website hosting



Module review

- Amazon S3 features, use cases, and basic concepts
- Data protection in Amazon S3 at rest and in transit
- Basic operations on Amazon S3 objects
- Access management of Amazon S3 resources
- Experience developing with Amazon S3 using the AWS SDKs.

- To finish this module, complete the **knowledge check**.

In this module, we addressed the following topics:

- Amazon S3 features, use cases, and basic concepts
- Data protection in Amazon S3 at rest and in transit
- Basic operations on Amazon S3 objects
- Access management of Amazon S3 resources

You also gained experience developing with Amazon S3 using the AWS SDKs.

To finish this module, please complete the corresponding knowledge check.