AWS Academy Cloud Developing
Module 03 Student Guide
Version 2.0.0

200-ACCDEV-20-EN-SG

Contents

AWS Academy Cloud Developing

Module 3: Developing Storage Solutions

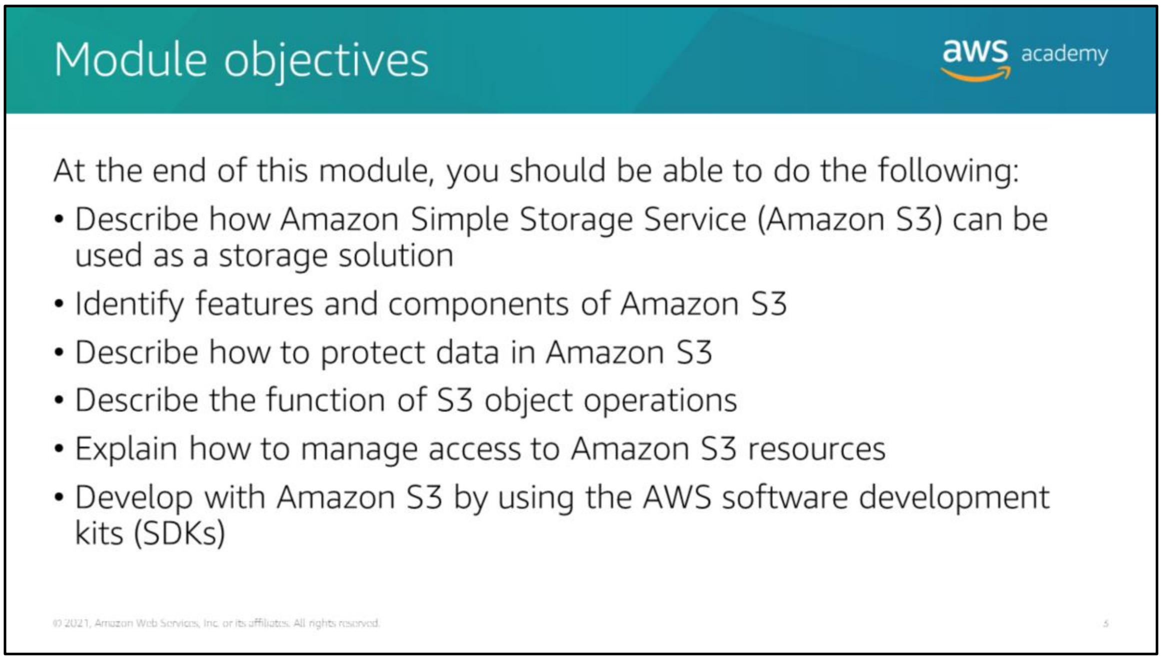
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved

Welcome to Module 3: Developing Storage Solutions.

Module 3: Developing Storage Solutions

Section 1: Introduction

Section 1: Introduction.



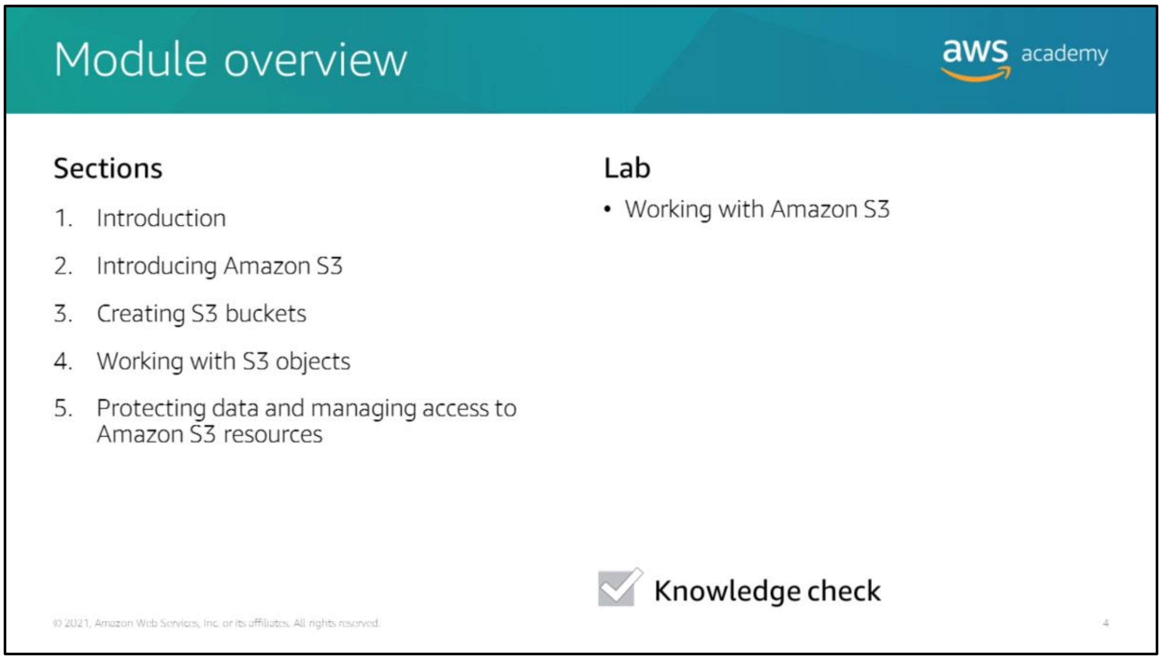
Module objectives

At the end of this module, you should be able to do the following:
- Describe how Amazon Simple Storage Service (Amazon S3) can be used as a storage solution
- Identify features and components of Amazon S3
- Describe how to protect data in Amazon S3
- Describe the function of S3 object operations
- Explain how to manage access to Amazon S3 resources
- Develop with Amazon S3 by using the AWS software development kits (SDKs)

5

At the end of this module, you should be able to do the following:

- Describe how Amazon Simple Storage Service (Amazon S3) can be used as a storage solution
- Identify features and components of Amazon S3
- Describe two ways to protect data in Amazon S3
- Describe the function of S3 object operations
- Explain how to manage access to Amazon S3 resources
- Develop with Amazon S3 by using the AWS software development kits (SDKs)



Module overview

aws academy

Sections

1. Introduction
2. Introducing Amazon S3
3. Creating S3 buckets
4. Working with S3 objects
5. Protecting data and managing access to Amazon S3 resources

Lab

- Working with Amazon S3

✓ Knowledge check

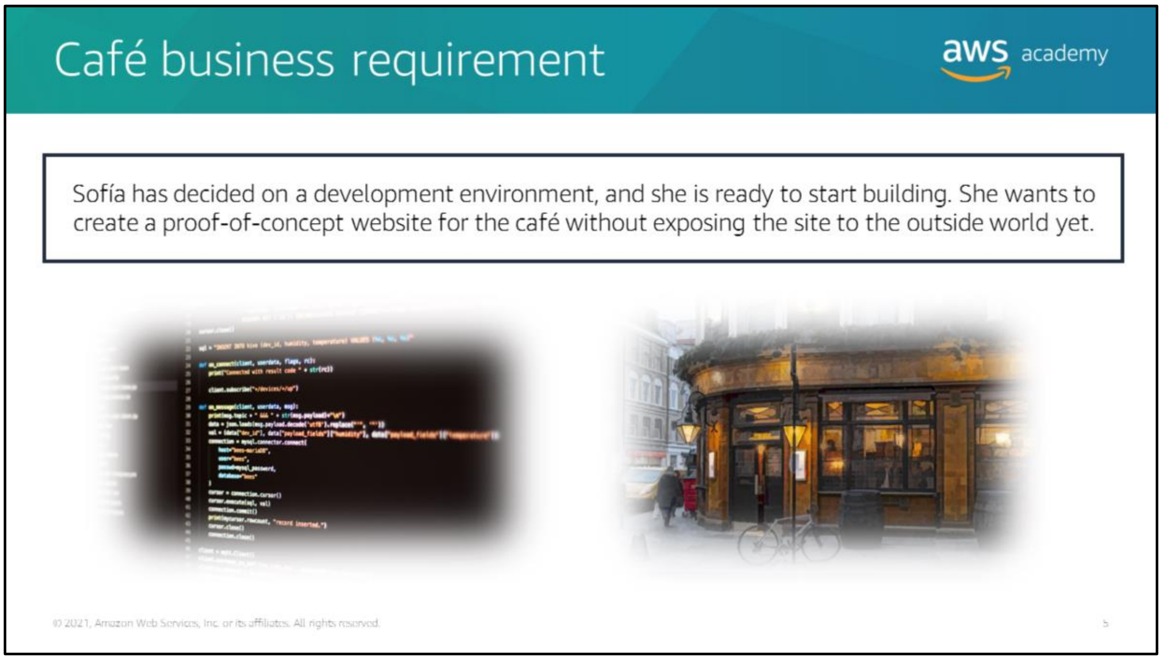
4

This module includes the following sections:

1. Introduction
2. Introducing Amazon S3
3. Creating S3 buckets
4. Working with S3 objects
5. Protecting data and managing access to Amazon S3 resources

This module also includes a lab about working with Amazon S3.

Finally, you will complete a knowledge check to test your understanding of key concepts covered in this module.

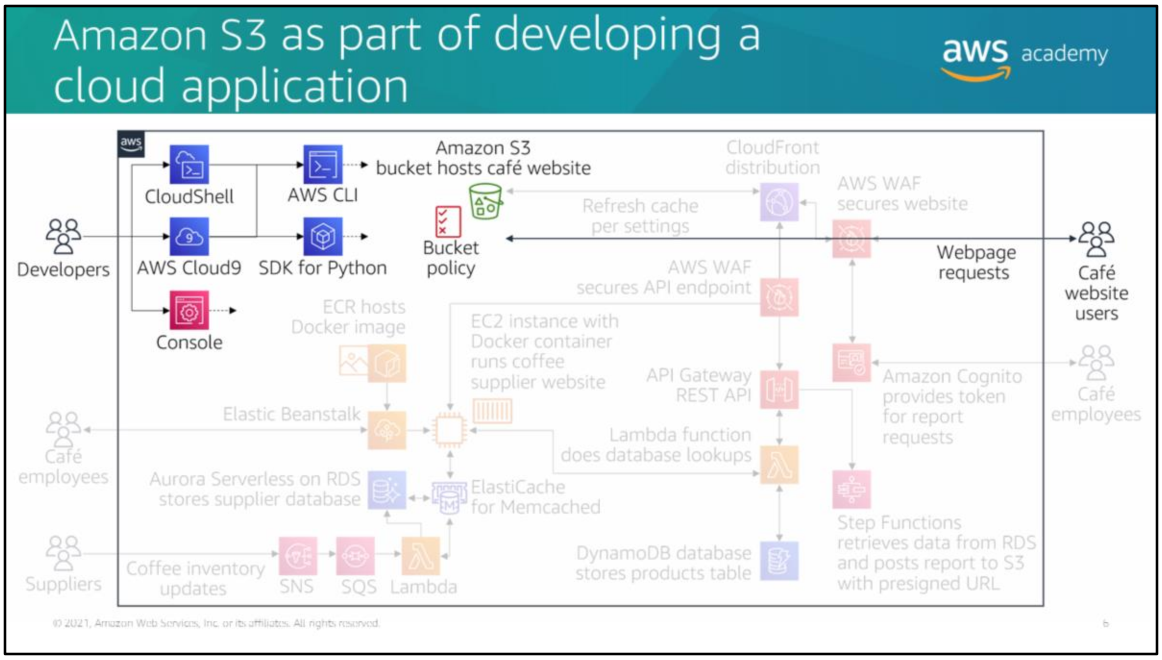


Café business requirement

Sofía has decided on a development environment, and she is ready to start building. She wants to create a proof-of-concept website for the café without exposing the site to the outside world yet.

5

Sofía has decided on a development environment, and she is ready to start building. She wants to create a proof-of-concept website for the café without exposing the site to the outside world yet. Sofía plans to use the proof-of-concept website to demonstrate to Frank and Martha how a website could visually showcase the café's offerings.



Amazon S3 as part of developing a cloud application

aws academy
Developers
CloudShell
AWS Cloud9
Console
AWS CLI
SDK for Python
Amazon S3 bucket hosts café website
Bucket policy
CloudFront distribution
Refresh cache per settings
AWS WAF secures website
Webpage requests
Café website users
AWS WAF secures API endpoint
ECR hosts Docker image
EC2 instance with Docker container runs coffee supplier website
API Gateway REST API
Amazon Cognito provides token for report requests
Café employees
Elastic Beanstalk
Café employees
Lambda function does database lookups
Aurora Serverless on RDS stores supplier database
ElastiCache for Memcached
Suppliers
Coffee inventory updates
SNS
SQS
Lambda
DynamoDB database stores products table
Step Functions retrieves data from RDS and posts report to S3 with presigned URL
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.
6

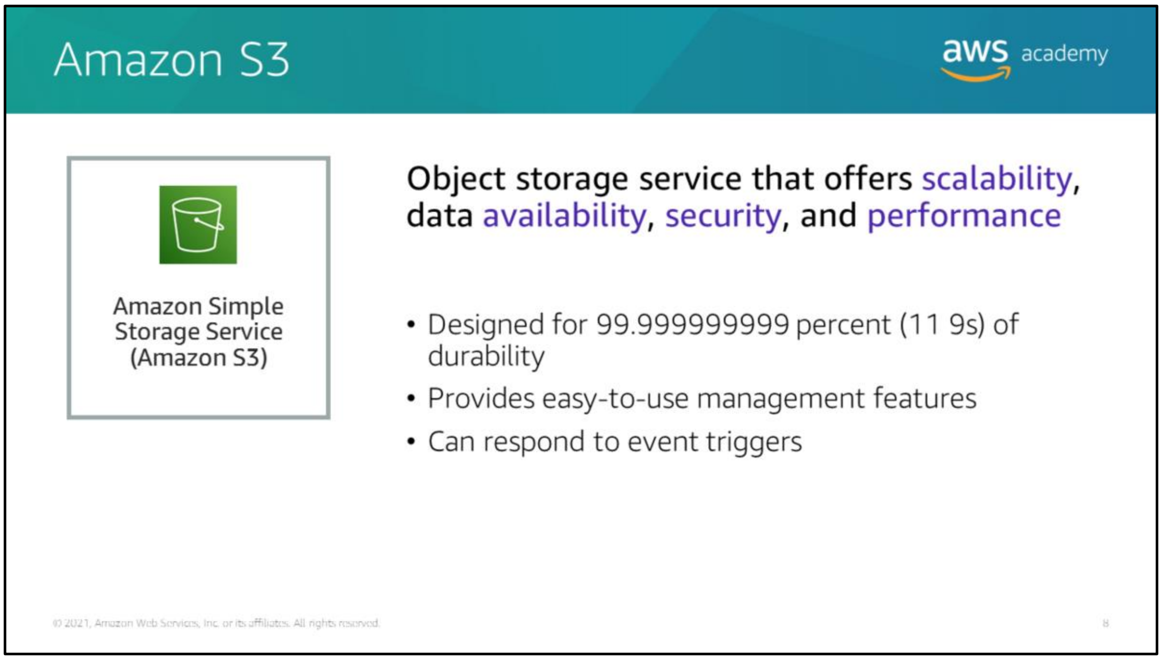
The diagram on this slide gives an overview of the application that you will build through the labs in this course. The highlighted portions are relevant to this module.

As highlighted in the diagram, you will host a static website with Amazon S3. The bucket will be protected with a bucket policy to limit the source of requests.

Module 3: Developing Storage Solutions

Section 2: Introducing Amazon S3

Section 2: Introducing Amazon S3.



Amazon S3

Amazon Simple Storage Service (Amazon S3)

Object storage service that offers scalability, data availability, security, and performance

- Designed for 99.999999999 percent (11 9s) of durability
- Provides easy-to-use management features
- Can respond to event triggers

8

When applications generate large amounts of data, developers need storage services that are designed to deliver virtually unlimited storage, durability, availability, performance, and security.

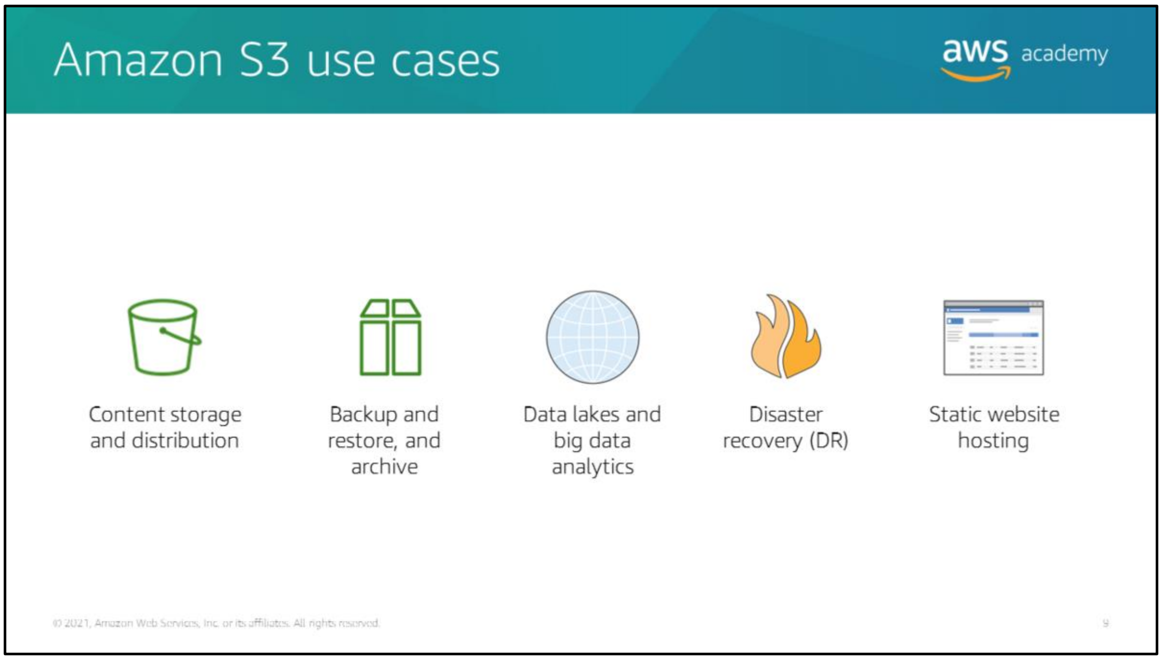
Amazon Simple Storage Service (Amazon S3) is an object storage service that offers the ability to store large amounts of data. Developers can use Amazon S3 as an object storage solution for backup files, Internet of Things (IoT) device data, big data repositories, and more.

Amazon S3 is designed for 99.999999999 percent (11 9s) of durability. This percentage means that if you store 10,000,00 objects in Amazon S3, you might lose one object every 10,000 years, on average.

You can manage Amazon S3 in the console or programmatically. Amazon S3 can be configured with fine-tuned access controls for specific business goals or compliance requirements.

You can configure Amazon S3 to generate event notifications. For example, you can configure notifications when an object is created, deleted, or restored; or lost from Reduced Redundancy Storage (RRS). You can also configure a notification to be sent when a replication event occurs. Notifications can be used with AWS Lambda, Amazon Simple Notification Service (Amazon SNS), or Amazon Simple Queue Service (Amazon SQS).

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



Amazon S3 use cases

Content storage and distribution

Backup and restore, and archive

Data lakes and big data analytics

Disaster recovery (DR)

Static website hosting

Amazon S3 has many use cases. You can use Amazon S3 for content storage and distribution, backup and restore, archiving, data lakes, big data analytics, disaster recovery (DR), and static website hosting.

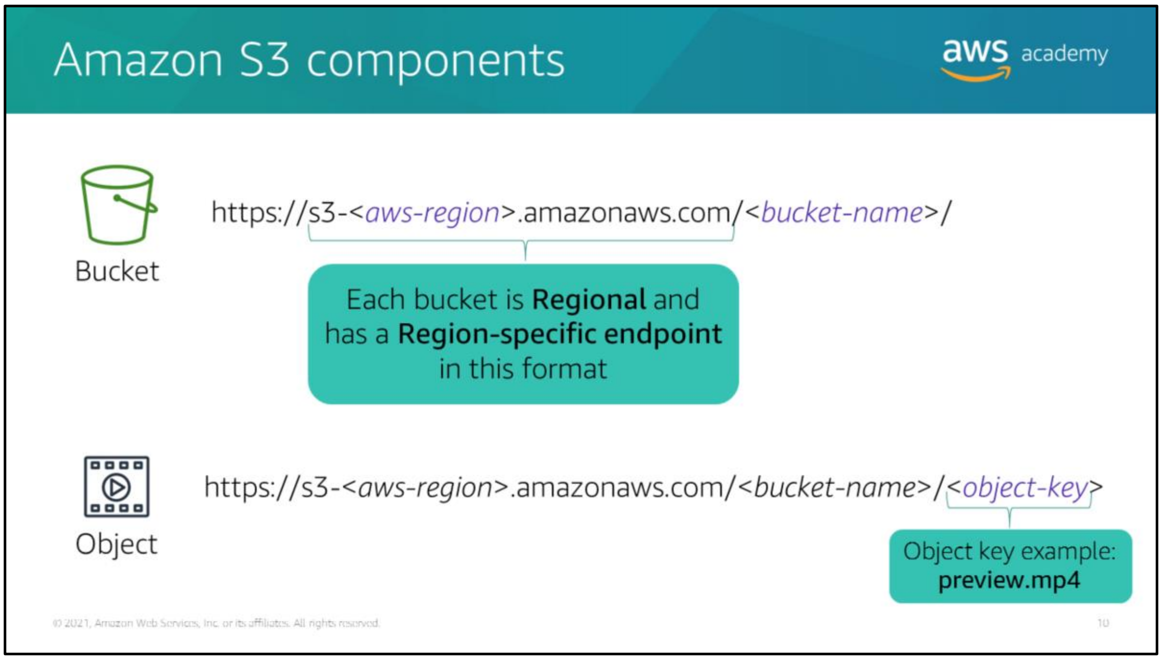
- You can securely store static content—such as images, videos, and music—to Amazon S3 and then distribute the content directly from Amazon S3 or through Amazon CloudFront edge locations.

- You can back up your data and use storage lifecycle policies to archive rarely accessed data to Amazon Simple Storage Service Glacier.

- 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.

- The highly durable, secure, global infrastructure provided by Amazon S3 offers a robust disaster recovery (DR)W solution. You can use Amazon S3 to protect critical data, applications, and IT systems 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.

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 webpage relies on server-side processing, including server-side scripts such as PHP, JSP, or ASP.NET. Amazon S3 doesn’t support server-side scripting. For more information, see Hosting a Static Website Using Amazon S3 at https://docs.aws.amazon.com/AmazonS3/latest/userguide/WebsiteHosting.html.



Amazon S3 components

Bucket

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

Each bucket is **Regional** and has a **Region-specific endpoint** in this format

Object

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

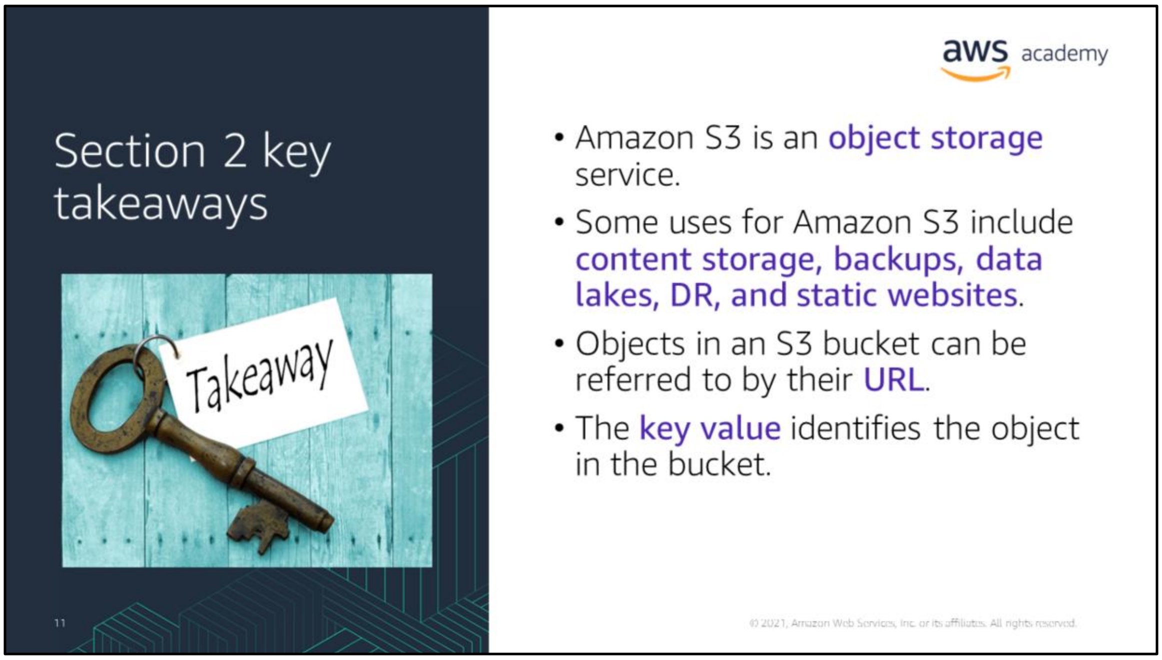
Object key example: **preview.mp4**

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

- A bucket is a container for objects that are stored in Amazon S3. Buckets serve several purposes. They organize the Amazon S3 namespace at the highest level, and they identify the account responsible for storage and data transfer charges. They also play a role in access control, and they serve as the unit of aggregation for usage reporting.
- An object is the fundamental entity that is 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 metadata is a set of name-value pairs that describe the object.
- An object key uniquely identifies the object in a bucket. Each object in a bucket has exactly one key.

Note that each bucket is Regional. You can choose the AWS Region where Amazon S3 will store the buckets that 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 that's created in the US Oregon Region is *us-west-2*.

For additional information, see Amazon S3 Concepts at
https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html.



Section 2 key takeaways

- Amazon S3 is an **object storage** service.
- Some uses for Amazon S3 include **content storage, backups, data lakes, DR, and static websites**.
- Objects in an S3 bucket can be referred to by their **URL**.
- The **key value** identifies the object in the bucket.

11

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

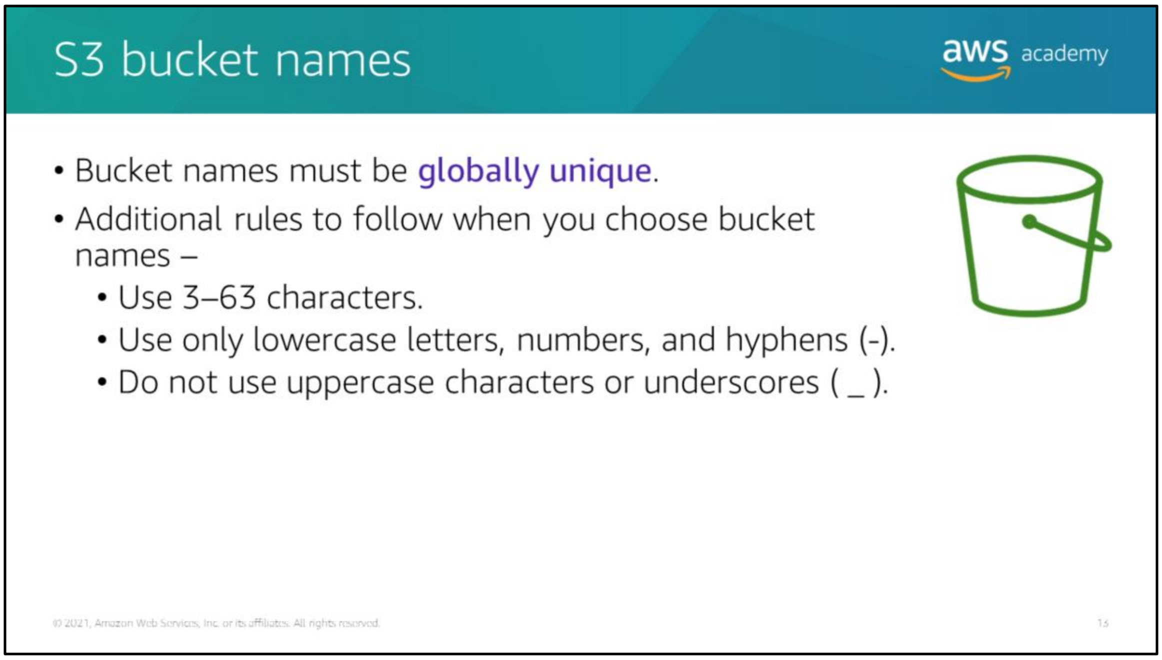
The following are the key takeaways from this section of the module:

- Amazon S3 is an object storage service.
- Some uses for Amazon S3 include content storage, backups, data lakes, DR, and static websites.
- Objects in an S3 bucket can be referred to by their URL.
- The key value identifies the object in the bucket.

Module 3: Developing Storage Solutions

Section 3: Creating S3 buckets

Section 3: Creating S3 buckets.



S3 bucket names

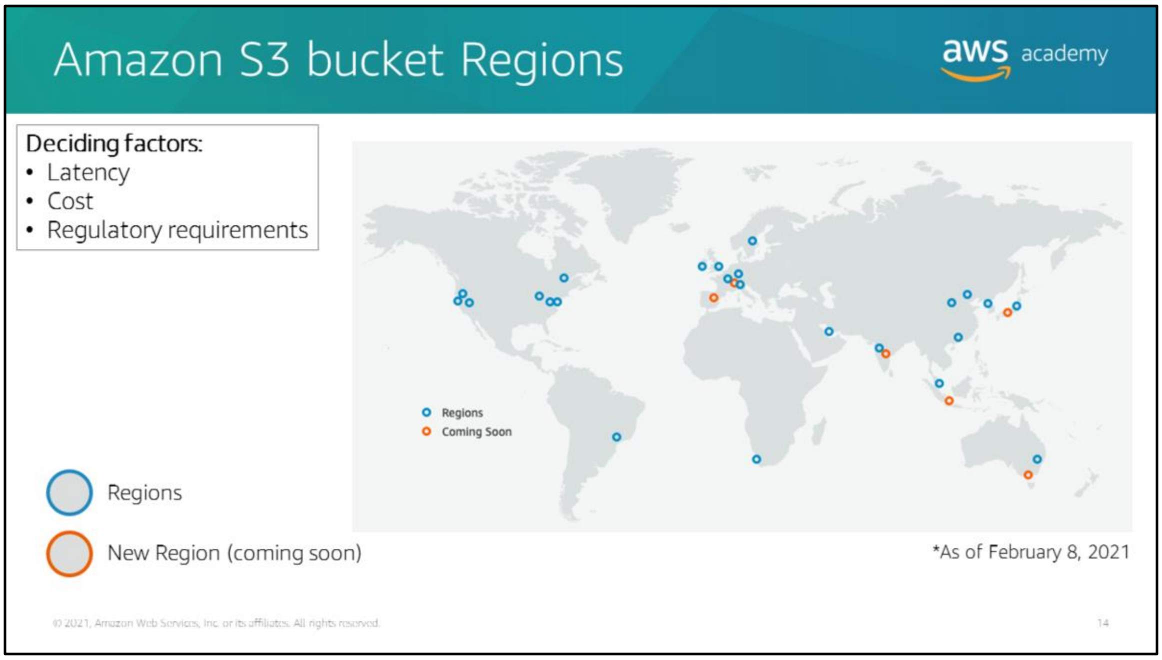
- Bucket names must be **globally unique**.
- Additional rules to follow when you choose bucket names –
  - Use 3–63 characters.
  - Use only lowercase letters, numbers, and hyphens (-).
  - Do not use uppercase characters or underscores (_).

When you create an S3 bucket, you specify a name for it. You must follow specific rules when you choose a bucket name:

- An S3 bucket name must be *globally unique*, and the namespace is shared by all AWS accounts. After bucket creation, the name of that bucket can't be used by another AWS account in any AWS Region until the bucket is deleted. To ensure uniqueness, you could add the current date, the application name, or a randomly generated number to the bucket name.
- Additionally, bucket names –
  - Must be at least 3 characters and no more than 63 characters.
  - Can contain lowercase letters, numbers, and hyphens (-).
  - Must not contain uppercase characters or underscores (_).

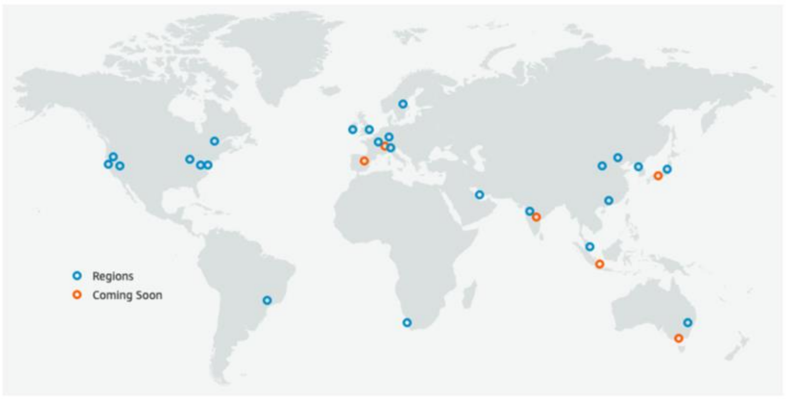
After you create an S3 bucket, you can't change the bucket name, so be careful when choosing the name.

For more information, including rules for naming buckets, see Bucket Restrictions and Limitations at https://docs.aws.amazon.com/AmazonS3/latest/userguide/BucketRestrictions.html.



Amazon S3 bucket Regions

Deciding factors:
- Latency
- Cost
- Regulatory requirements



◯ Regions

◯ New Region (coming soon)

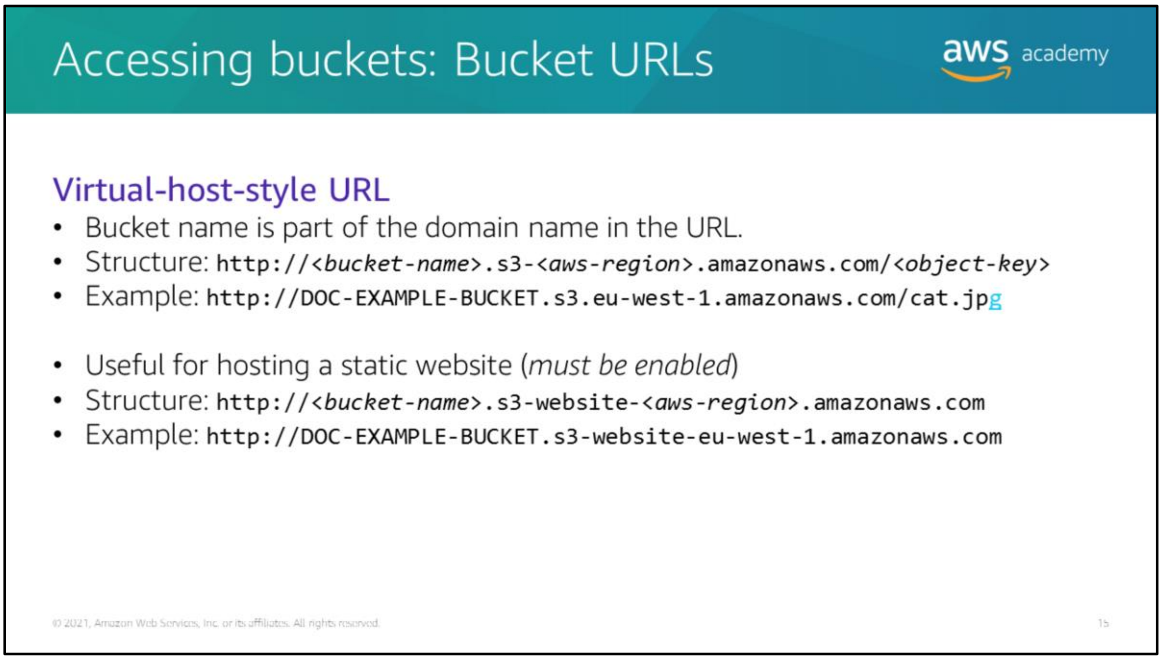
*As of February 8, 2021

When you create S3 creates buckets, you must specify the Region. When you choose a Region, you should consider latency, cost, or regulatory requirements that could affect the data in the bucket.

After you create a bucket, you cannot change its Region.

When you programmatically connect to to an AWS service, you connect to a URL that is the service endpoint. For information about Amazon S3 Regional endpoints, see AWS Service Endpoints at https://docs.aws.amazon.com/general/latest/gr/rande.html.

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



Accessing buckets: Bucket URLs

aws academy

Virtual-host-style URL

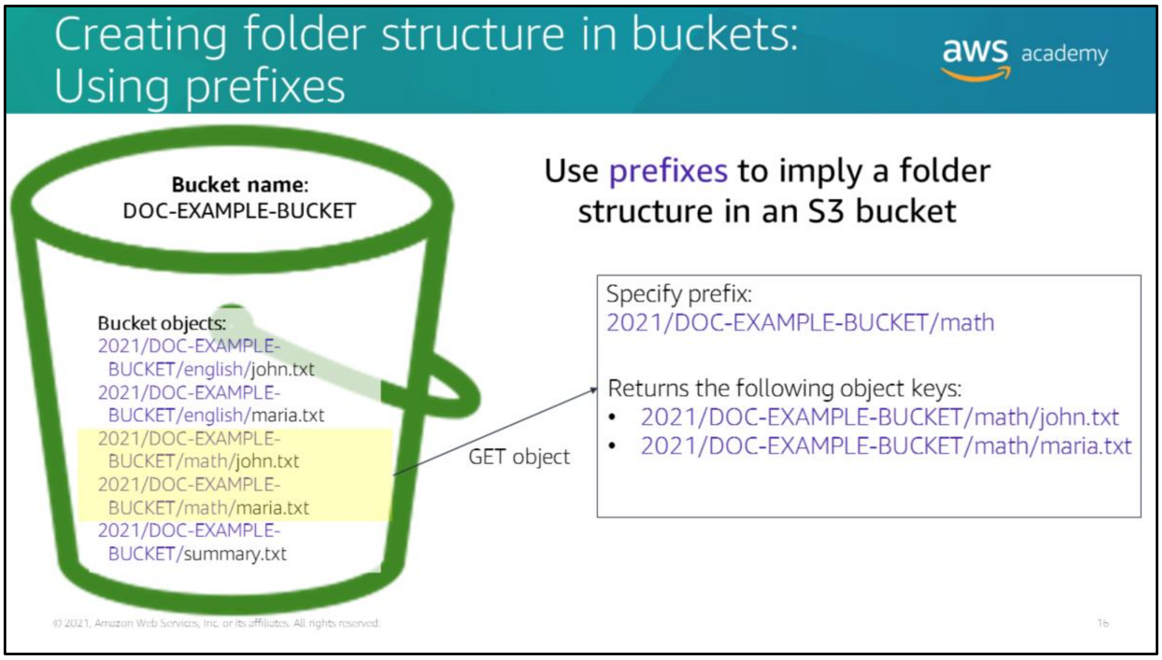
- Bucket name is part of the domain name in the URL.
- Structure: `http://<bucket-name>.s3-<aws-region>.amazonaws.com/<object-key>`
- Example: `http://DOC-EXAMPLE-BUCKET.s3.eu-west-1.amazonaws.com/cat.jpg`

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

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. 15

You can access your bucket by using the Amazon S3 console, or you can access it programmatically. You can access objects with virtual-host-style URLs.
- In a virtual-host-style URL, the bucket name is part of the domain name in the URL.
- The virtual-host-style URL is useful if you use your S3 bucket to host a static website. To host a static website, you configure an S3 bucket for website hosting and then upload your website content to the bucket. This bucket must have public read access.

For more information, see the following resources:
- Working with Amazon S3 Buckets at https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingBucket.html
- Virtual Hosting of Buckets at https://docs.aws.amazon.com/AmazonS3/latest/userguide/VirtualHosting.html
- Hosting a Static Website Using Amazon S3 at https://docs.aws.amazon.com/AmazonS3/latest/userguide/WebsiteHosting.html



Creating folder structure in buckets: Using prefixes

Use prefixes to imply a folder structure in an S3 bucket

Bucket name:
DOC-EXAMPLE-BUCKET
Bucket objects:
2021/DOC-EXAMPLE-BUCKET/english/john.txt
2021/DOC-EXAMPLE-BUCKET/english/maria.txt
2021/DOC-EXAMPLE-BUCKET/math/john.txt
2021/DOC-EXAMPLE-BUCKET/math/maria.txt
2021/DOC-EXAMPLE-BUCKET/summary.txt
GET object

Specify prefix:
2021/DOC-EXAMPLE-BUCKET/math

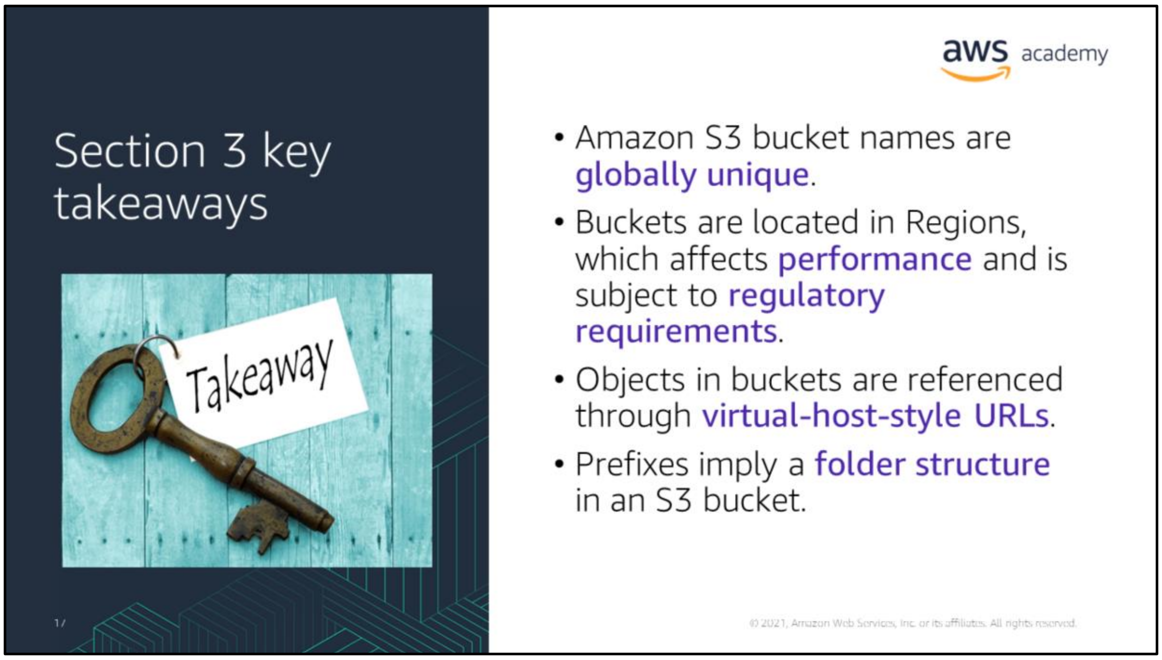
Returns the following object keys:
- 2021/DOC-EXAMPLE-BUCKET/math/john.txt
- 2021/DOC-EXAMPLE-BUCKET/math/maria.txt

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. 16

In Amazon S3, buckets and objects are the primary resources, where objects are stored in buckets. For organizational simplicity, Amazon S3 supports the folder concept to group objects. Amazon S3 uses a shared name prefix for objects (that is, objects that have names that begin with a common string). When you create a folder, the Amazon S3 console creates an object with the name followed by a slash (/). The Amazon S3 console then displays that object as a folder.

When an Amazon S3 bucket is queried, a prefix will limit results to only those objects that begin with that prefix. In the example, the bucket contains objects with student English and math scores for the year 2021. To GET all the math-score object keys for 2021, specify the prefix `2021/DOC-EXAMPLE-BUCKET/math`.

For more information, see the following resources:
- Organizing Objects in the Amazon S3 Console Using Folders at https://docs.aws.amazon.com/AmazonS3/latest/userguide/using-folders.html
- Organizing Objects Using Prefixes at https://docs.aws.amazon.com/AmazonS3/latest/userguide/using-prefixes.html



Section 3 key takeaways

- Amazon S3 bucket names are **globally unique**.
- Buckets are located in Regions, which affects **performance** and is subject to **regulatory requirements**.
- Objects in buckets are referenced through **virtual-host-style URLs**.
- Prefixes imply a **folder structure** in an S3 bucket.

17

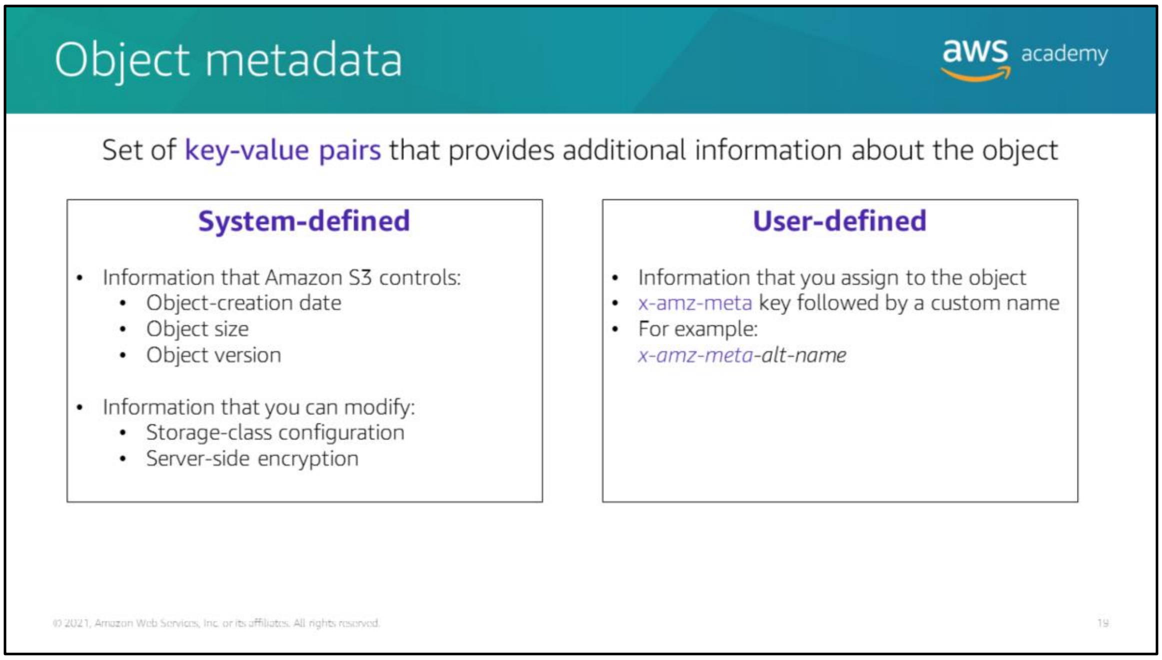
The following are the key takeaways from this section of the module:

- Amazon S3 bucket names are globally unique.
- Buckets are located in Regions, which affects performance and is subject to regulatory requirements.
- Objects in buckets are referenced through virtual-host-style URLs.
- Prefixes imply a folder structure in an S3 bucket.

Module 3: Developing Storage Solutions

Section 4: Working with S3 objects

Section 4: Working with S3 objects.



Object metadata

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

System-defined

- Information that Amazon S3 controls:
  - 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

19

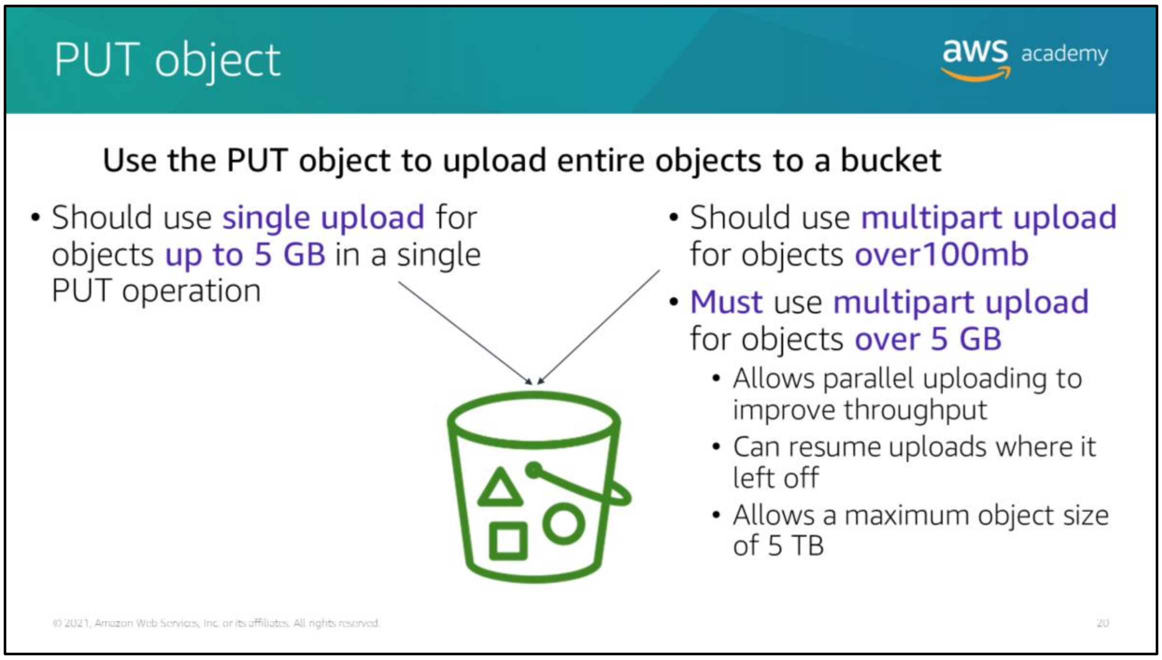
Objects consist of *object data* and *metadata*. Object metadata is a set of key-value pairs that provide additional information about the object. There are two kinds of object metadata:

- *System-defined metadata* – For each object that's 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. Amazon S3 controls certain types of system-defined metadata, such as object-creation date, size, and version. You can update some types of system-defined metadata, such as storage configuration and server-side encryption.

- *User-defined metadata* – You can assign user-defined metadata to an object during an upload or after it has been uploaded. Like system-defined metadata, user-defined metadata is stored with the object. However, Amazon S3 does not process user-defined metadata. User-defined metadata must begin with the *x-amz-meta-* prefix. For example, to create a user-defined metadata value *alt-name*, the metadata key would be *x-amz-meta-alt-name.*

For more information about object metadata, see the following resources:

- Object Metadata at https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingMetadata.html
- Editing Object Metadata at https://docs.aws.amazon.com/AmazonS3/latest/userguide/add-object-metadata.html



PUT object

Use the PUT object to upload entire objects to a bucket

- Should use single upload for objects up to 5 GB in a single PUT operation

- Should use multipart upload for objects over100mb
- Must use multipart upload for objects over 5 GB
  - Allows parallel uploading to improve throughput
  - Can resume uploads where it left off
  - Allows a maximum object size of 5 TB

20

With the PUT object operation, you add an object to an S3 bucket. You must have write permissions to add an object to a bucket. Amazon S3 always writes the entire object. You cannot do partial updates to an object.

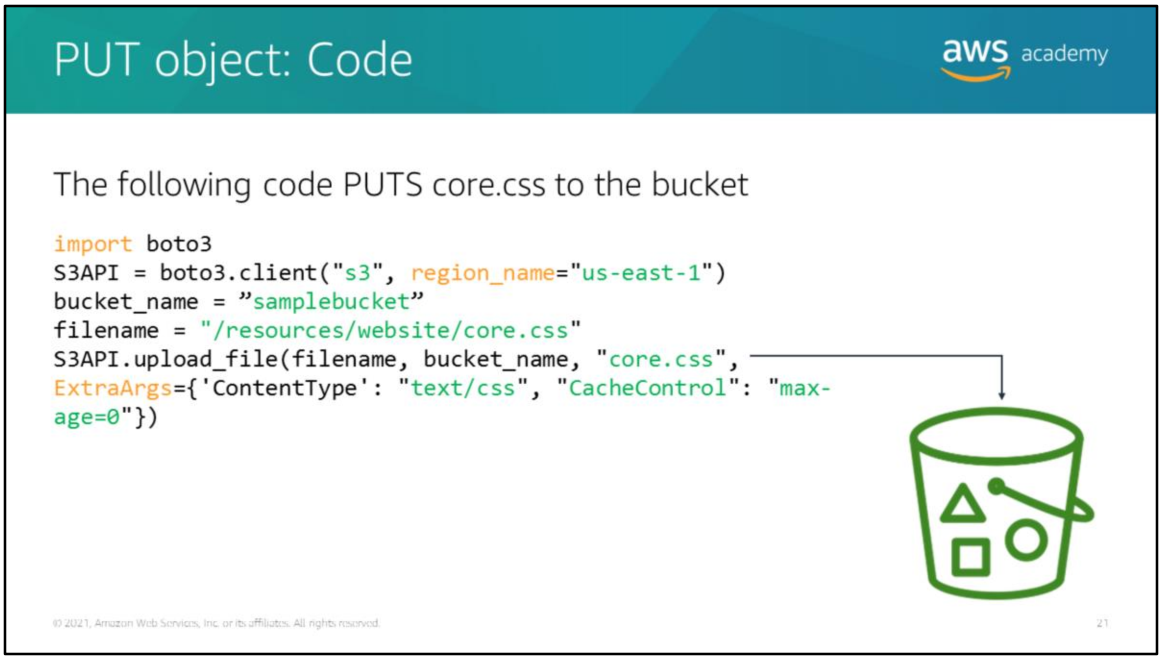
There are two types of uploads available for the PUT operation:

- *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 of up to 5 TB, you should use the multipart upload. You can use multipart upload 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 resume that upload without affecting the other parts. After all parts of your object are uploaded, Amazon S3 assembles these parts and creates the object.

As a general recommendation, when an object’s size reaches 100 MB, you should consider using multipart uploads.

For more information, see the following resources:

- PutObject operation at
  https://docs.aws.amazon.com/AmazonS3/latest/API/API_PutObject.html
- Multipart Upload Overview at
  https://docs.aws.amazon.com/AmazonS3/latest/userguide/mpuoverview.html



PUT object: Code

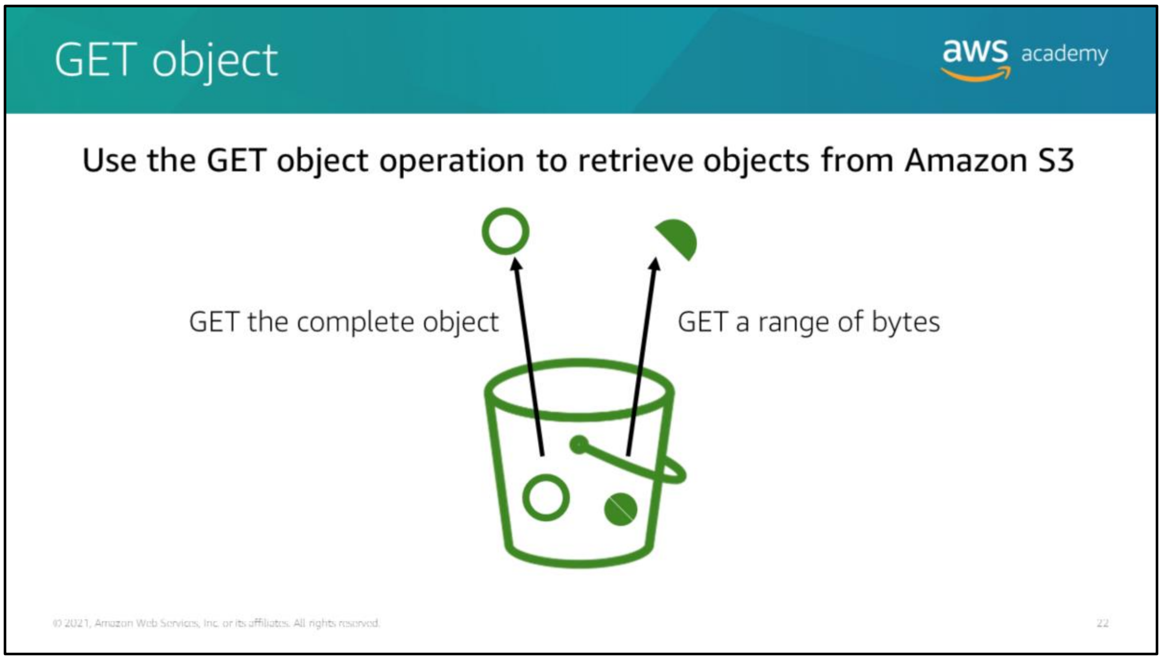
The following code PUTS core.css to the bucket

```
import boto3
S3API = boto3.client("s3", region_name="us-east-1")
bucket_name = ”samplebucket”
filename = "/resources/website/core.css"
S3API.upload_file(filename, bucket_name, "core.css",
ExtraArgs={'ContentType': "text/css", "CacheControl": "max-
age=0"})
```

21

The code demonstrates the PUT operation. In order of each line:

- The code begins by importing the AWS SDK for Python (Boto3) library.
- The variable S3AP is set with the call to boto3.client, which is the client service API. The service (s3) and Region (us-east-1) are specified.
- The variable bucket_name is set for the bucket name.
- The filename value is set to the file that will be uploaded.
- The upload file operation is called, with arguments for the file name, bucket name, the file, and content type. CacheControl sets how long a client web browser will cache the file. Setting the value to zero (such as in the example) means that web browsers will not cache the object. This setting can be very useful for objects that change regularly.



GET object

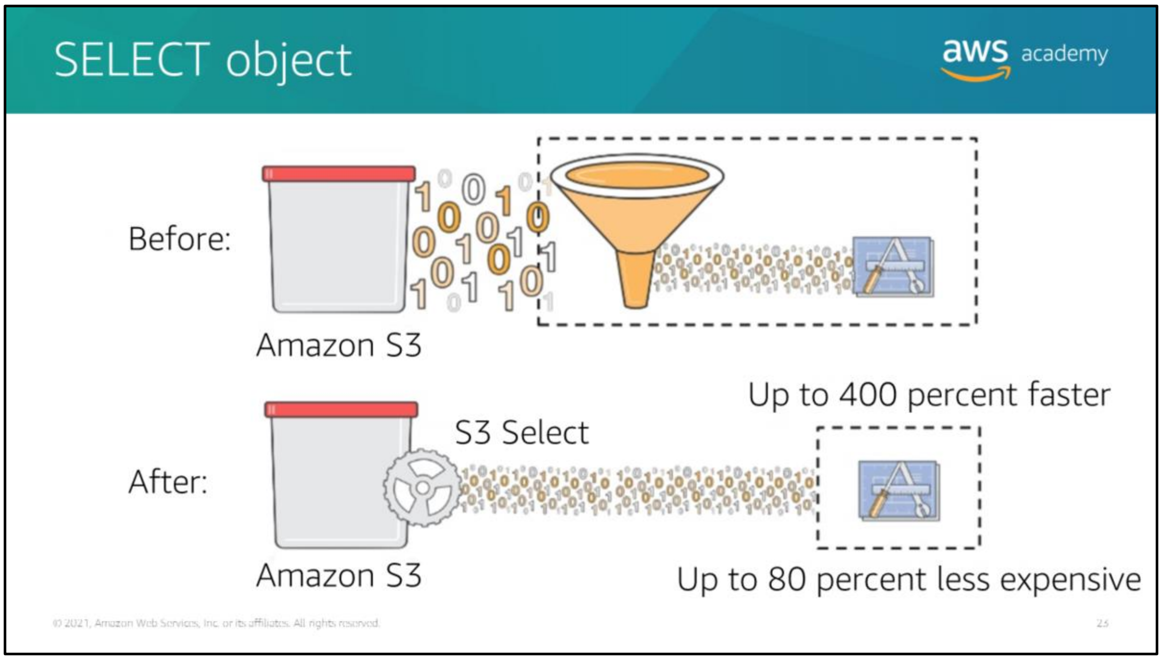
Use the GET object operation to retrieve objects from Amazon S3

GET the complete object
GET a range of bytes

22

You use the GET object operation to retrieve objects from Amazon S3. Using GET requires read access to the bucket. You can retrieve a complete object in a single GET request. You can also retrieve an object in parts by specifying a range of bytes. This feature is useful in scenarios where network connectivity is low or if your application can only process subsets of object data.

For more information about the GET Object operation, see GetObject in the *Amazon S3 API Reference* at https://docs.aws.amazon.com/AmazonS3/latest/API/API_GetObject.html.



SELECT object

aws academy
Before:
Amazon S3
After:
S3 Select
Amazon S3
Up to 400 percent faster
Up to 80 percent less expensive
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.
25

You can use the SELECT operation to query Amazon S3 objects based on structured query language (SQL). You send a SQL expression and 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 records that match the specified SQL expression. You must have `s3:GetObject` permissions for this operation.

For example, you can use the select request to retrieve all records from an S3 object by using:

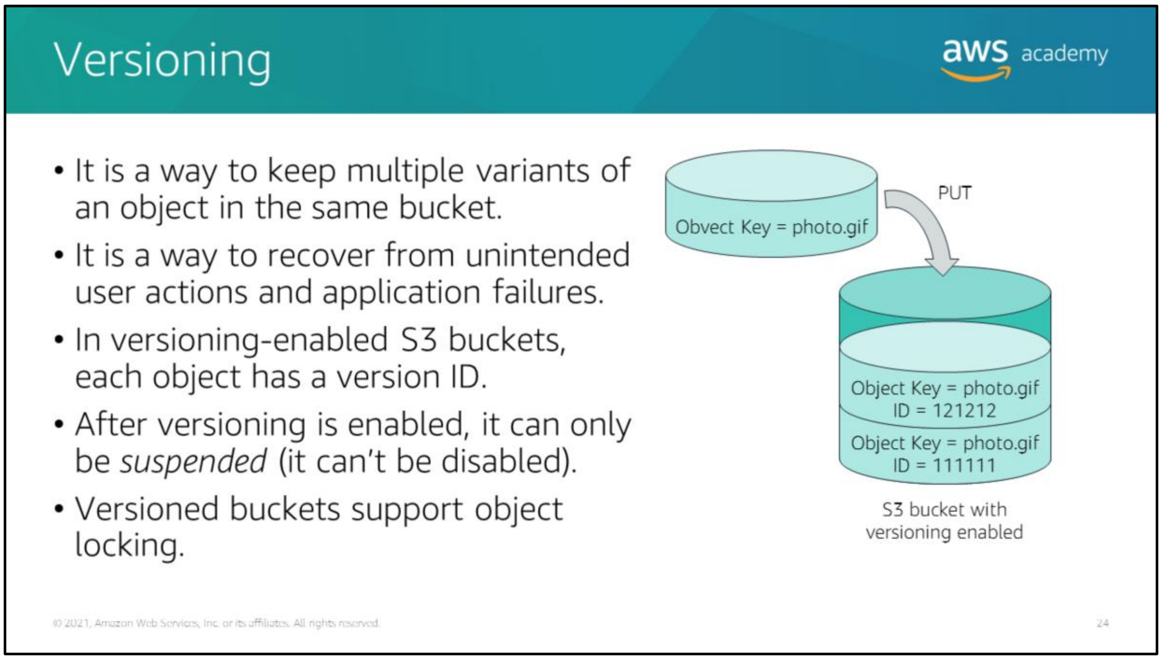
```
Select * from S3Object
```

You could also specify required columns from the query by specifying the column name:

```
SELECT s.Id, s.FirstName, s.SSN FROM S3Object s
```

Because your applications do not need 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, see SelectObjectContent in the *Amazon S3 API Reference* at https://docs.aws.amazon.com/AmazonS3/latest/API/API_SelectObjectContent.html.



Versioning

- It is a way to keep multiple variants of an object in the same bucket.
- It is a way to recover from unintended user actions and application failures.
- In versioning-enabled S3 buckets, each object has a version ID.
- After versioning is enabled, it can only be *suspended* (it can't be disabled).
- Versioned buckets support object locking.

Obvect Key = photo.gif
PUT
Object Key = photo.gif ID = 121212
Object Key = photo.gif ID = 111111
S3 bucket with versioning enabled

Versioning is another feature of Amazon S3 that you can use to protect your data. Versioning is a way to keep multiple variants of an object in the same bucket. Two objects in a bucket can have the same key but different version IDs. For example, an image file named *photo.gif* can have two versions: photo.gif (version 111111) and photo.gif (version 121212).

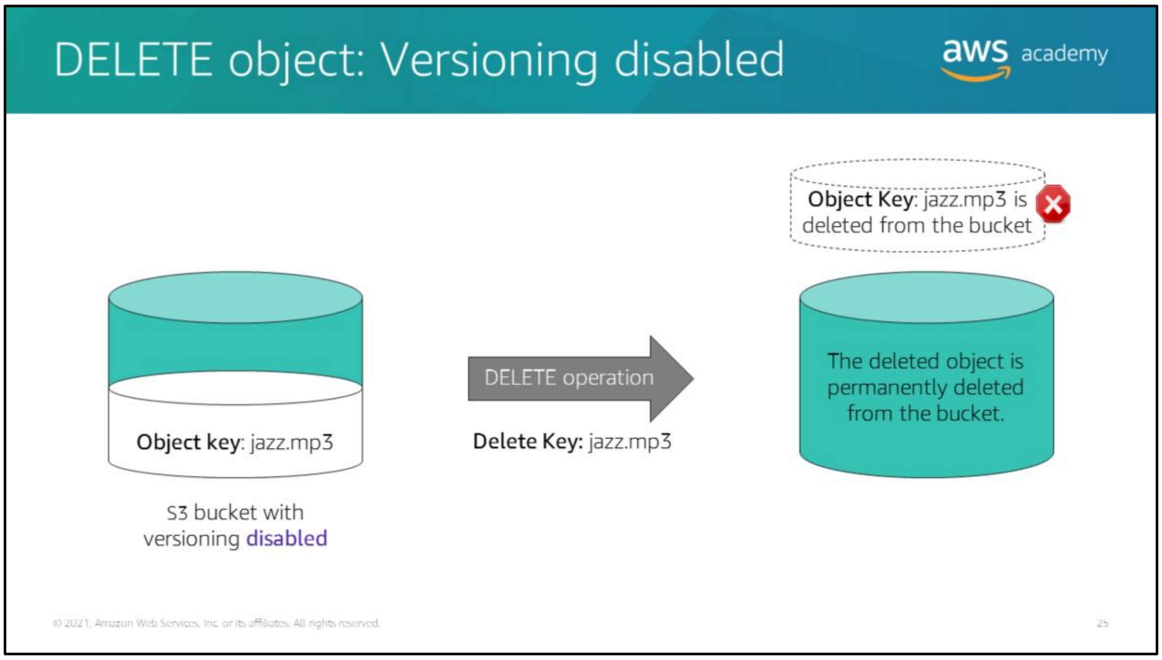
You can use versioning to preserve, retrieve, and restore each version of every object that's stored in your S3 bucket. With versioning, you can recover from unintended user actions (such as overwrites and deletions) and application failures. You can also use versioning to archive objects so that you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

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

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

For more information, see the following resources:

- Using Versioning at https://docs.aws.amazon.com/AmazonS3/latest/userguide/Versioning.html
- Locking Objects Using S3 Object Lock at https://docs.aws.amazon.com/AmazonS3/latest/userguide/object-lock.html

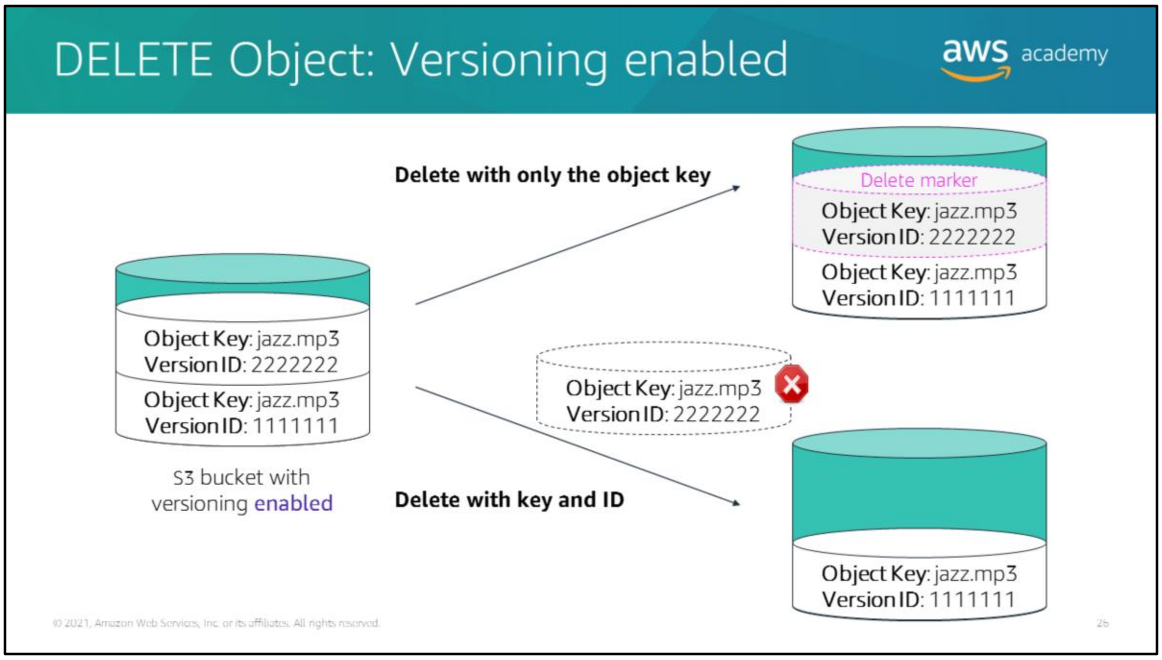


DELETE object: Versioning disabled

aws academy
Object key: jazz.mp3
S3 bucket with versioning disabled
DELETE operation
Delete Key: jazz.mp3
Object Key: jazz.mp3 is deleted from the bucket
The deleted object is permanently deleted from the bucket.
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.
25

With the DELETE object operation, you can delete a single object or multiple objects. You permanently delete an object by specifying the key of the object that you want to delete.

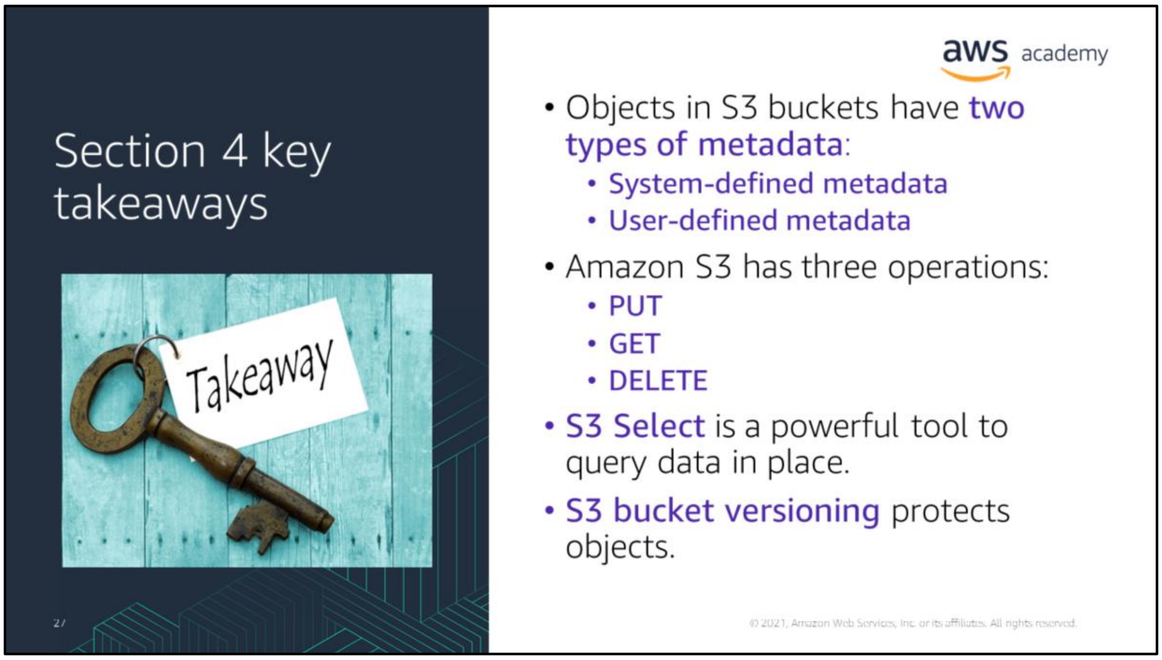
For more information, see Deleting Object Versions at https://docs.aws.amazon.com/AmazonS3/latest/userguide/DeletingObjectVersions.html.



DELETE Object: Versioning enabled

In a bucket that has versioning, you can permanently delete an object by invoking a DELETE request with a key and version ID. You must delete each individual version to remove an object completely. If you specify only a delete key, Amazon S3 adds a delete marker that becomes the current version of the object. Objects that have a delete marker will return a 404: Not Found error message when it receives a request to retrieve the object.

For more information, see Deleting Object Versions at https://docs.aws.amazon.com/AmazonS3/latest/userguide/DeletingObjectVersions.html.



Section 4 key takeaways

- Objects in S3 buckets have **two types of metadata**:
  - **System-defined metadata**
  - **User-defined metadata**
- Amazon S3 has three operations:
  - **PUT**
  - **GET**
  - **DELETE**
- **S3 Select** is a powerful tool to query data in place.
- **S3 bucket versioning** protects objects.

27

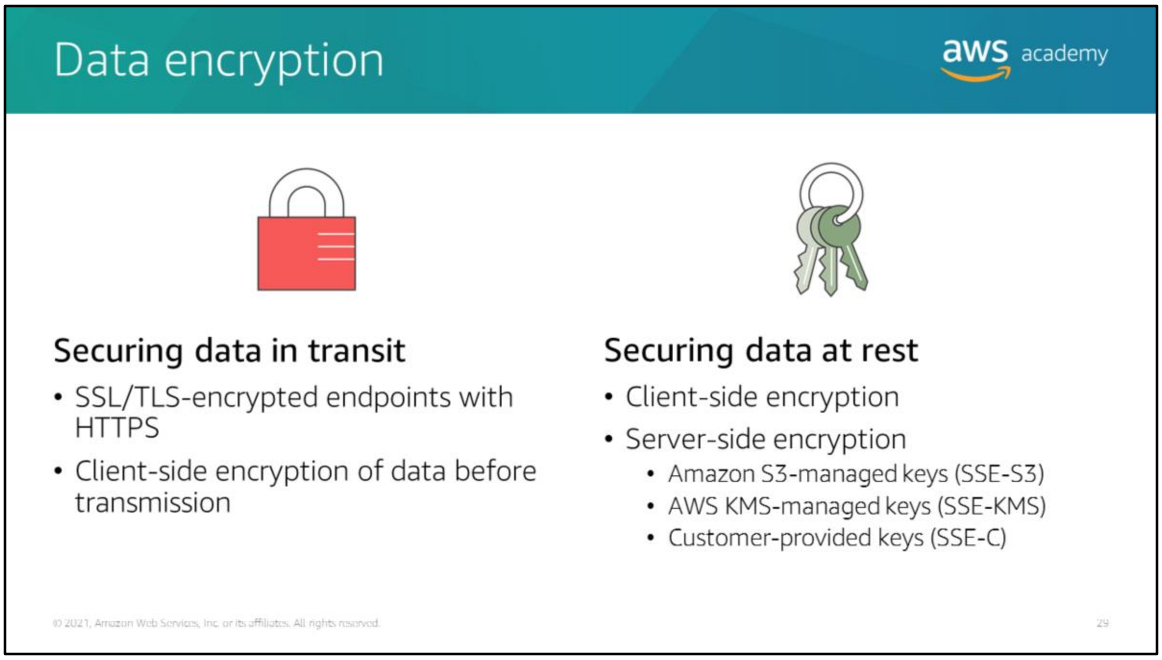
The following are the key takeaways from this section of the module:

- Objects in an Amazon S3 bucket have system defined and can be configured with user-defined metadata.
- The PUT operation writes objects to a bucket, a GET operation reads objects from a bucket, and DELETE removes objects from a bucket.
- S3 Select is a powerful tool to query data in place.
- Versioning enables S3 buckets to protect objects.

Module 3: Developing Storage Solutions

Section 5: Protecting data and managing access to Amazon S3 resources

Section 5: Protecting data and managing access to Amazon S3 resources.



Data encryption

aws academy

Securing data in transit

- SSL/TLS-encrypted endpoints with HTTPS
- Client-side encryption of data before transmission

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)

29

Data protection refers to protecting data while it’s in transit (as it travels to and from Amazon S3) and while it’s at rest (when it’s stored in Amazon S3 data centers).

You can protect data in transit by using Secure Sockets Layer/Transport Layer Security (SSL/TLS) or using client-side encryption. Clients must support TLS version 1.0. AWS recommends TLS version 1.2 or higher. For more information, see Internetwork Traffic Privacy at https://docs.aws.amazon.com/AmazonS3/latest/userguide/inter-network-traffic-privacy.html.

To protect data at rest in Amazon S3, you can use 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. For information about which SDKs support client-side encryption, see Protecting Data Using Client-Side Encryption at https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingClientSideEncryption.html.

With server-side encryption, Amazon S3 encrypts your data at the object level. It then decrypts the data for you when you access it. You have three options, depending on how you choose to manage the encryption keys.

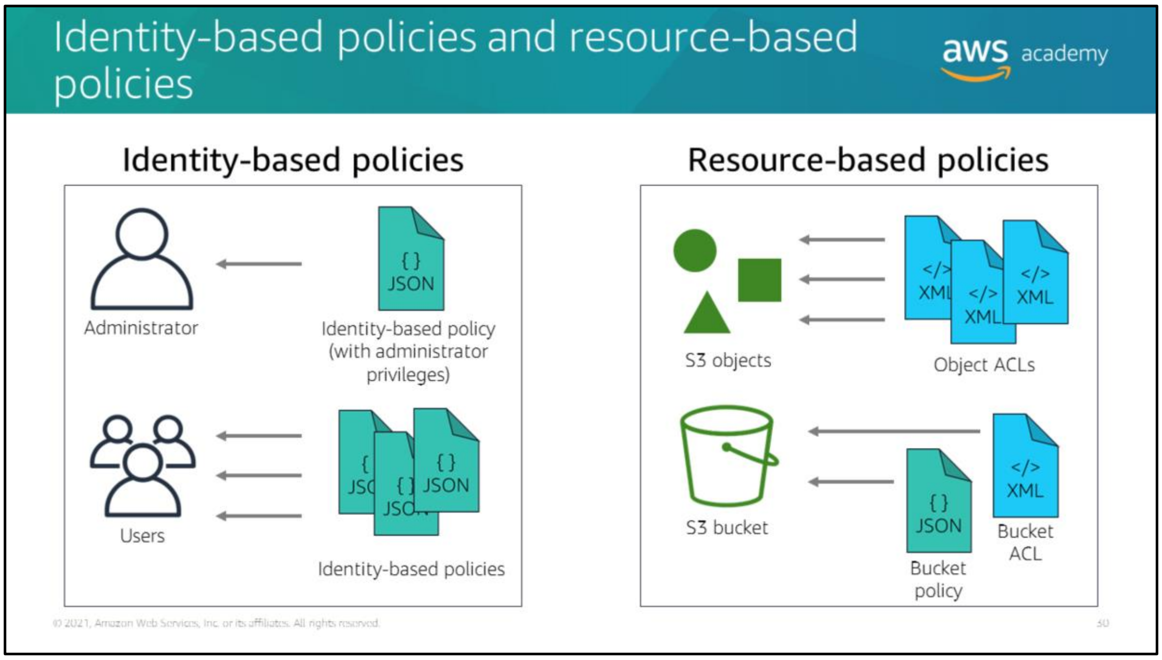
Server-Side Encryption with Amazon S3-managed Keys (SSE-S3) – Each object is encrypted with a unique key that employs strong multi-factor encryption. It also encrypts the key itself with a root key that it regularly rotates. Amazon S3 server-side encryption uses 256-bit Advanced Encryption Standard (AES-256) to encrypt your data.

Server-Side Encryption with AWS KMS-managed Keys (SSE-KMS) – This option is similar to SSE-S3. Using an envelope key requires additional permissions. (An envelope key protects your data's encryption key.) These additional permissions provide added protection against unauthorized access of your S3 objects. SSE-KMS also provides you with an audit trail of when your key was used and by whom. Additionally, you can create and manage encryption keys yourself, or you can use a default key that is unique to you, the service you that you use, and the Region.

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 handles the decryption when you access your objects.

Note that S3 will not encrypt objects that are already in the bucket when encryption is enabled. S3 only encrypts objects that are added to the bucket after encryption is enabled.

For more information about server-side encryption with Amazon S3, see Protecting Data Using Server-Side Encryption at https://docs.aws.amazon.com/AmazonS3/latest/userguide/serv-side-encryption.html.



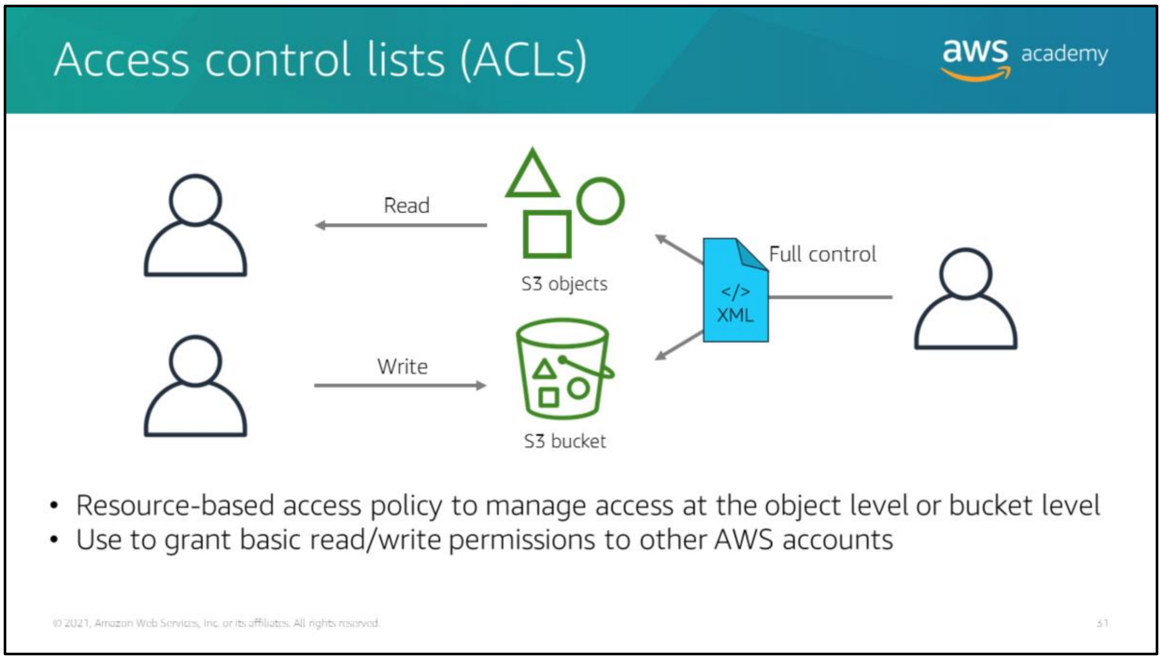
Identity-based policies and resource-based policies

You can manage access to your Amazon S3 resources by writing *access policies* that grant permissions to other users. For example, you can grant PUT object permissions to a user who will then be able to 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.

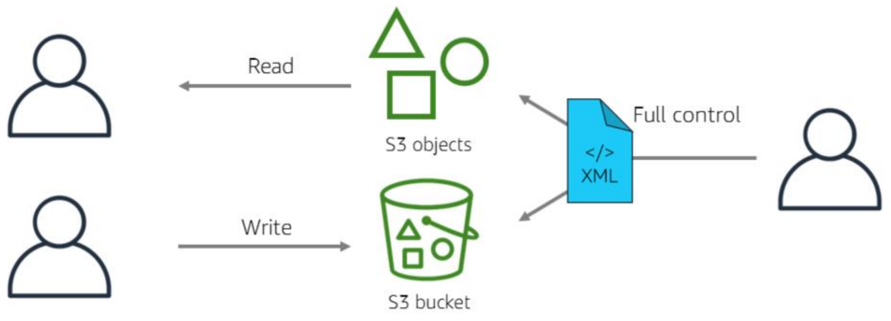
Amazon S3 offers two types of access policies:

- *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.

For more information, see Overview of Managing Access at https://docs.aws.amazon.com/AmazonS3/latest/userguide/access-control-overview.html.



Access control lists (ACLs)



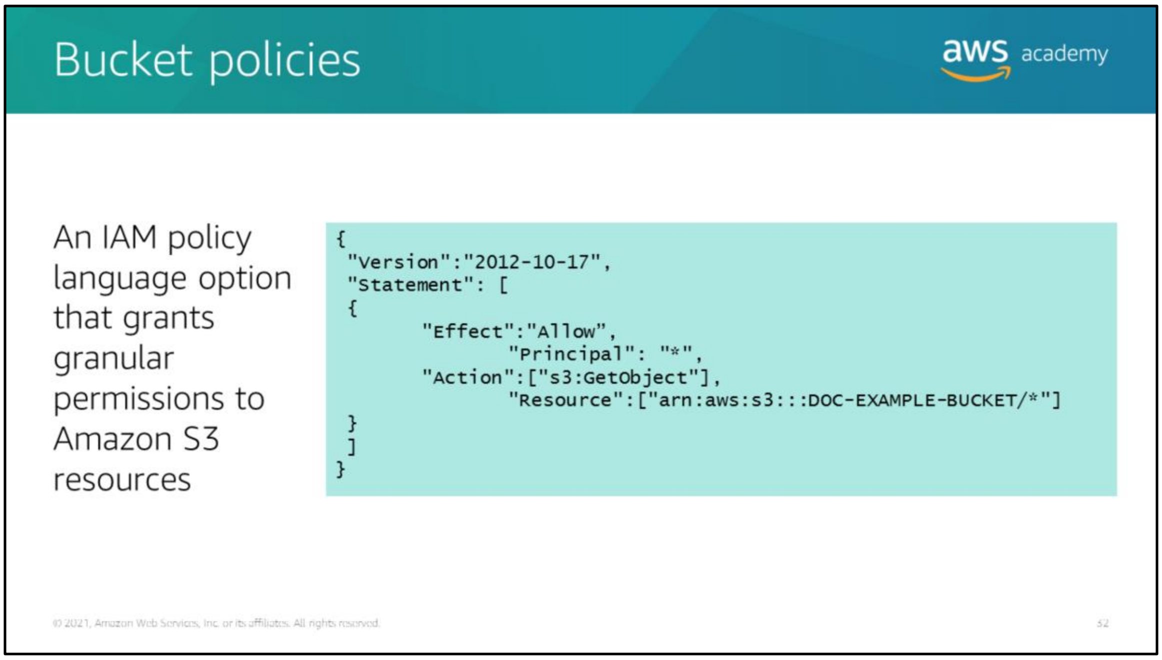
- Resource-based access policy to manage access at the object level or bucket level
- Use to grant basic read/write permissions to other AWS accounts

Access control lists (ACLs) are resource-based access policies that manage access to buckets and objects. You can use ACLs to grant necessary read/write permissions to other AWS accounts.

You use an object ACL for the following scenarios:

- An object ACL is the only way to manage access to objects that the bucket owner does not own. The AWS account that owns a bucket can grant permissions to another AWS account to upload objects to that bucket. The bucket owner does not own these objects. The AWS account that created the object must grant permissions to that object by using object ACLs.
- Permissions vary by object. You must manage permissions at the object level. For large numbers of objects, you can write a single policy statement that grants object-level permissions to an AWS account.
- Object ACLs control only object-level permissions. The entire bucket has a single bucket policy, but object ACLs are specified per object.

For more information about managing access with ACLs, see Access Control List (ACL) Overview at https://docs.aws.amazon.com/AmazonS3/latest/userguide/acl_overview.html.



Bucket policies

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

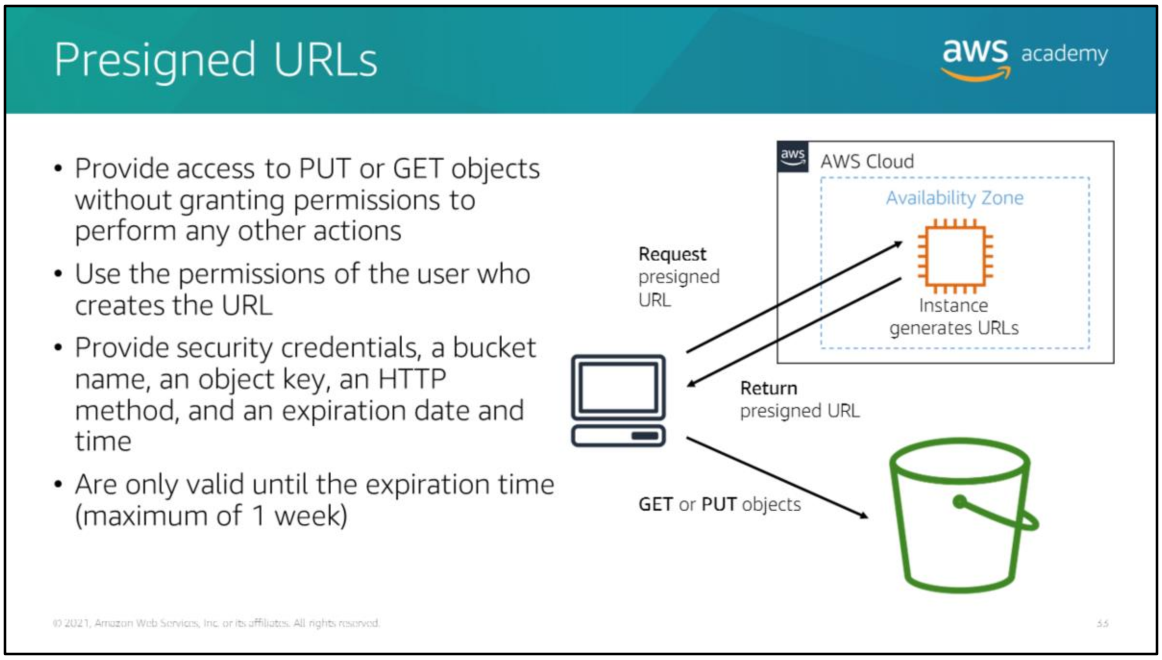
```
{
 "Version":"2012-10-17",
 "Statement": [
 {
       "Effect":"Allow",
              "Principal": "*",
       "Action":["s3:GetObject"],
              "Resource":["arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"]
 }
 ]
}
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. 52

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 permissions on all objects in the bucket. The bucket policy has one statement, which allows the `s3:GetObject` action (read permissions) on objects in a bucket named DOC-EXAMPLE-BUCKET. The policy grants anonymous access by specifying the principal with a wild card (*).

Use a bucket policy to manage cross-account permissions for all Amazon S3 permissions. For more information, see Access Policy Guidelines at https://docs.aws.amazon.com/AmazonS3/latest/userguide/access-policy-alternatives-guidelines.html.



Presigned URLs

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

aws academy
AWS Cloud
Availability Zone
Instance generates URLs
Request presigned URL
Return presigned URL
GET or PUT objects

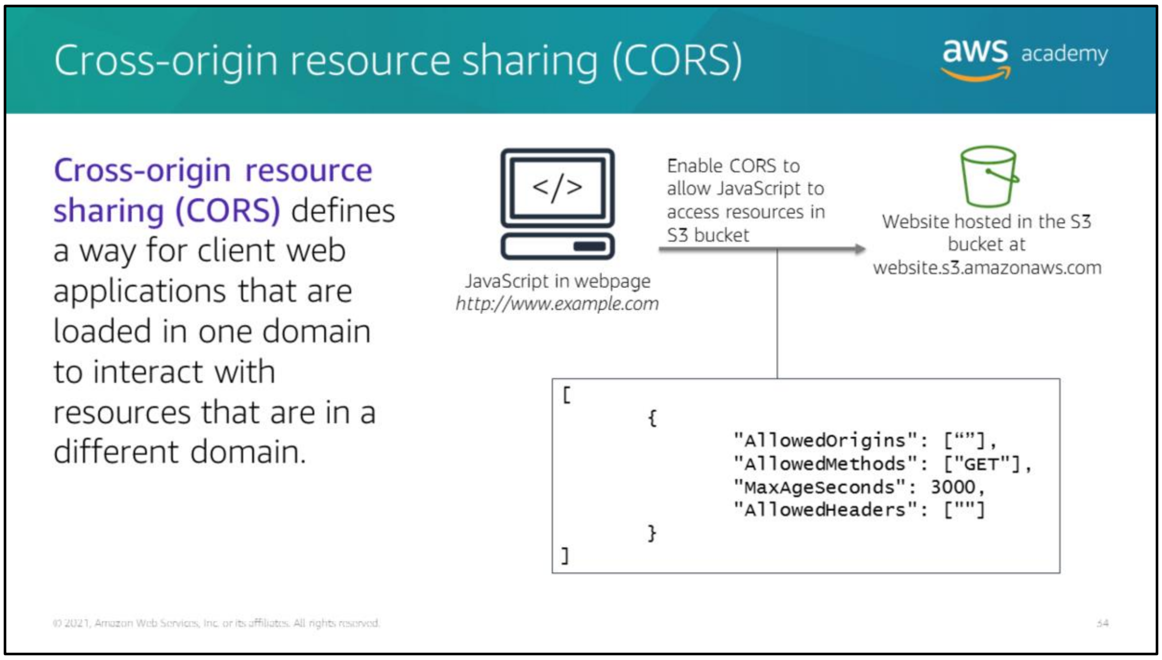
55

Presigned URLs are useful if you want a user to upload or download a specific object from a bucket without changing AWS security credentials or permissions. When you create a presigned URL, you must provide:

- Your security credentials
- A bucket name
- An object key
- An HTTP method (PUT for uploading objects, GET for retrieving objects)
- The expiration date and time

You can use the AWS CLI to generate presigned URLs for downloading objects. Only the address of the object is required (for example: s3://<bucket name>/<object key>).

For more information about presigned URLs, see Uploading Objects Using Presigned URLs at https://docs.aws.amazon.com/AmazonS3/latest/userguide/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 that are in a different domain.

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

```
[
        {
                "AllowedOrigins": [“”],
                "AllowedMethods": ["GET"],
                "MaxAgeSeconds": 3000,
                "AllowedHeaders": [""]
        }
]
```

54

Websites sometimes need to load resources from other locations that are outside the website’s domain. Cross-origin resource sharing (CORS) defines a way for client web applications that are in one domain to interact with resources in a different domain.

Consider the following examples of when to enable CORS:

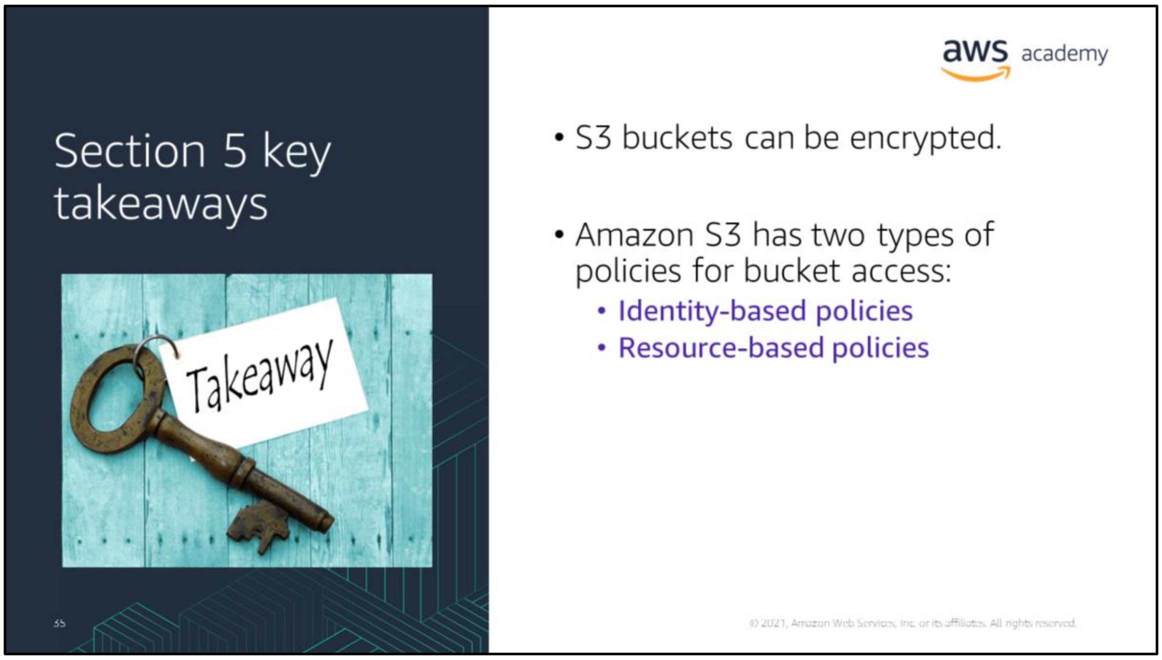
- JavaScript in one domain’s webpage (*http://www.example.com*) wants to use your S3 bucket resources by using the endpoint *website.s3.amazonaws.com*. The browser will allow such cross-domain access only if CORS is enabled on your bucket.
- You host a web font in your S3 bucket. A webpage that’s in a different domain uses this web font. The browser that loads the webpage will perform a CORS check to ensure that its domain can 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 JSON file. It must specify rules that identify the origins that you will allow to access your bucket, the allowed operations (HTTP methods), and other operation-specific information. You can add up to 100 rules to the configuration. Applying the policy can be done from the console, the AWS SDKS, or by calling the S3 REST API.

For more information, see the following resources:

- Cross-Origin Resource Sharing at https://docs.aws.amazon.com/AmazonS3/latest/userguide/cors.html
- Enabling Cross-Origin Resource Sharing at https://docs.aws.amazon.com/AmazonS3/latest/userguide/ManageCorsUsing.html



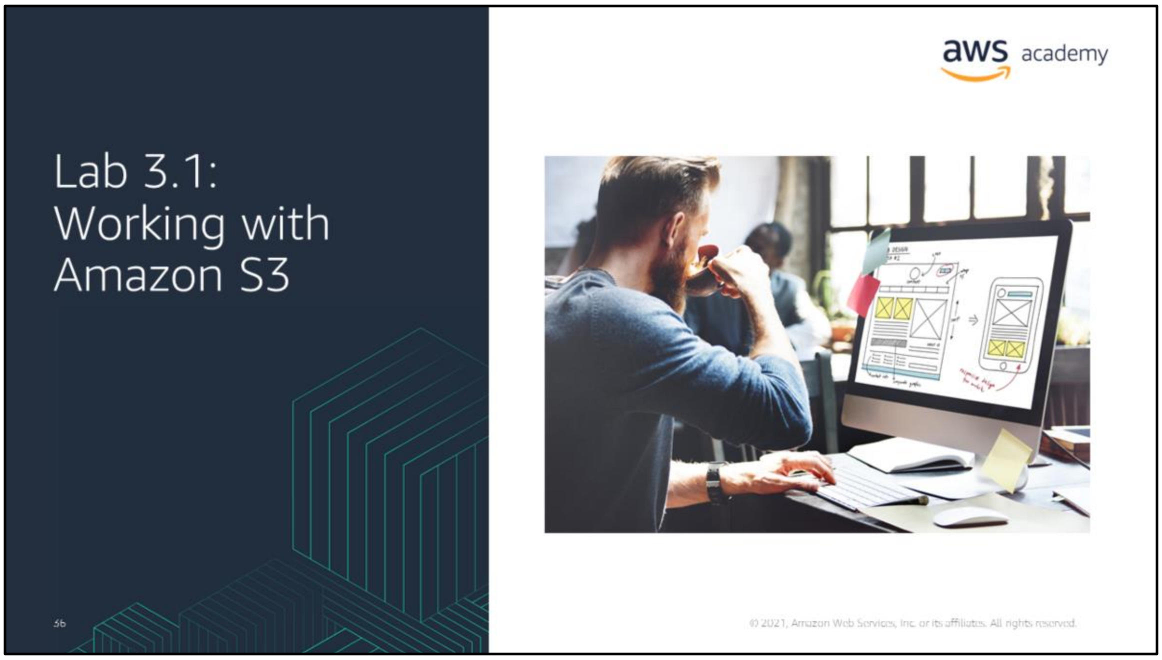
Section 5 key takeaways

- S3 buckets can be encrypted.
- Amazon S3 has two types of policies for bucket access:
  - Identity-based policies
  - Resource-based policies

55

The following are the key takeaways from this section of the module:

- S3 buckets can be encrypted.
- Amazon S3 has two types of policies for bucket access—Identity-based policies and resource-based policies.

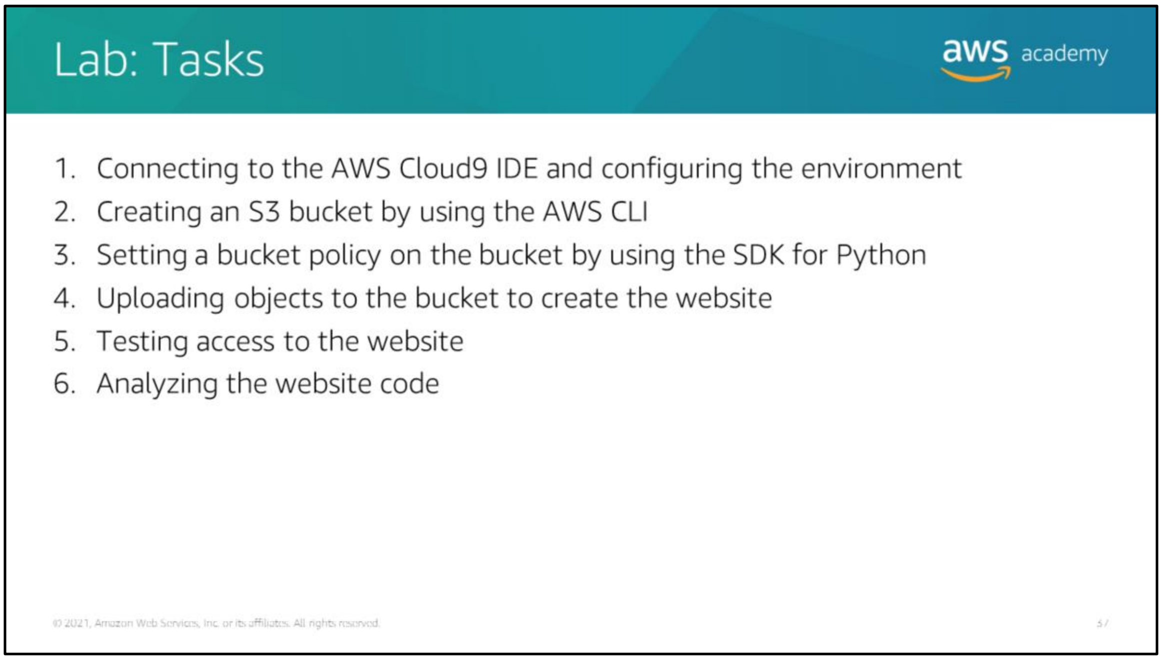


Lab 3.1: Working with Amazon S3

aws academy

56

You will now complete Lab 3.1: Working with Amazon S3.

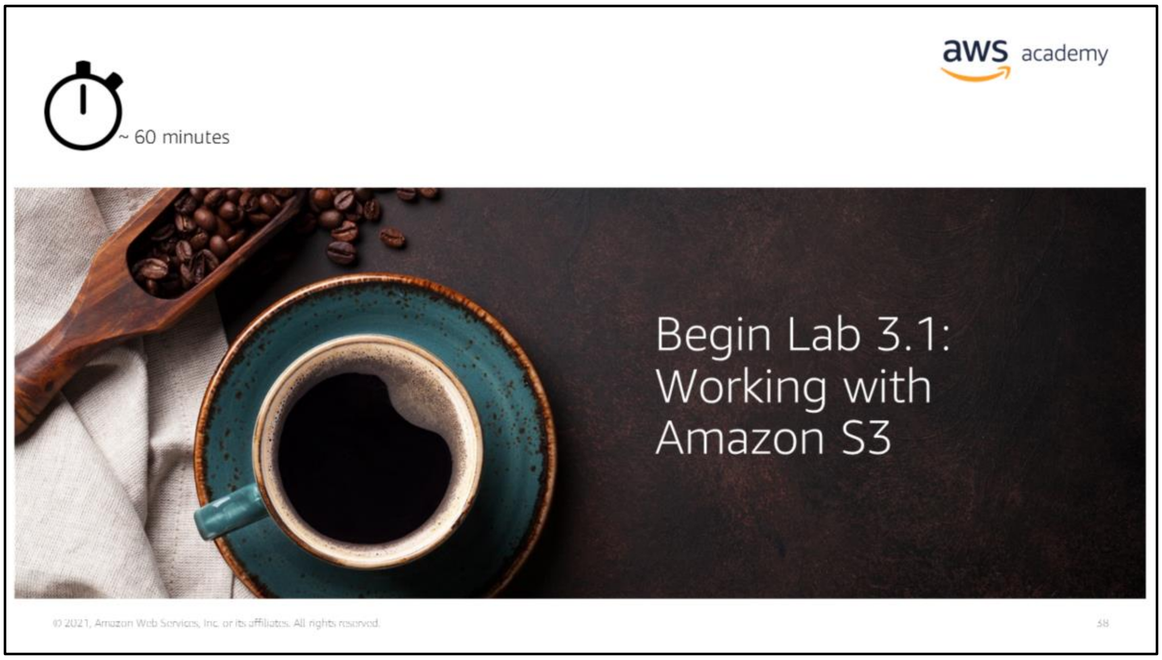


Lab: Tasks

1. Connecting to the AWS Cloud9 IDE and configuring the environment
2. Creating an S3 bucket by using the AWS CLI
3. Setting a bucket policy on the bucket by using the SDK for Python
4. Uploading objects to the bucket to create the website
5. Testing access to the website
6. Analyzing the website code

In this lab, you will complete the following tasks:

1. Connecting to the AWS Cloud9 IDE and configuring the environment
2. Creating an S3 bucket by using the AWS CLI
3. Setting a bucket policy on the bucket by using the SDK for Python
4. Uploading objects to the bucket to create the website
5. Testing access to the website
6. Analyzing the website code

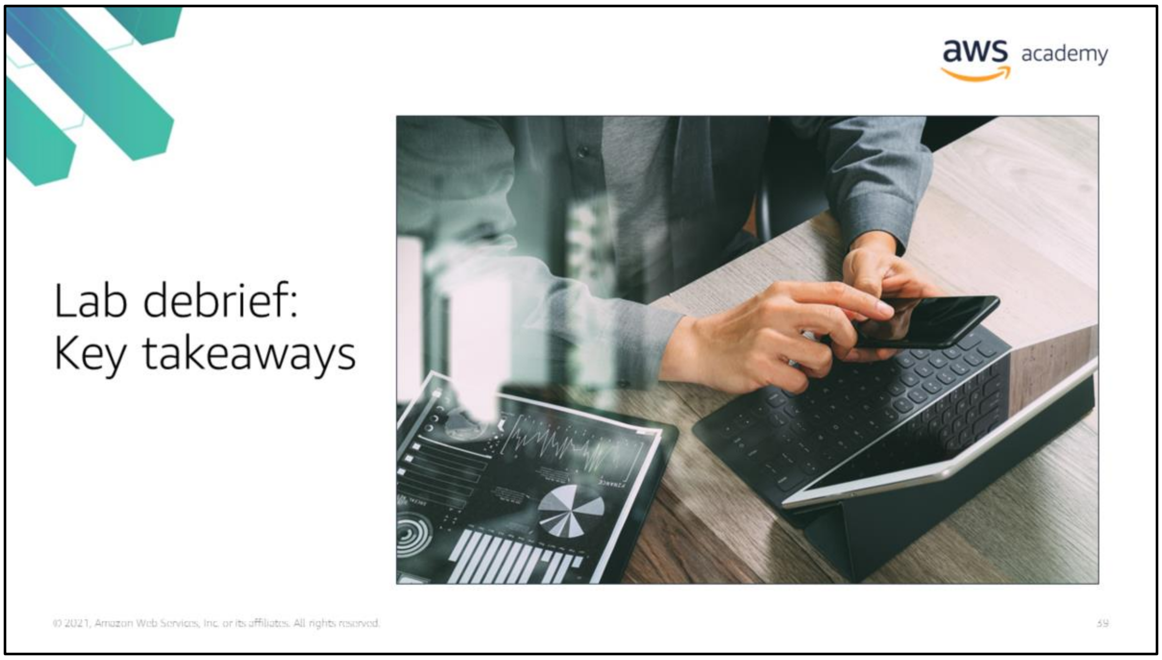


~ 60 minutes

Begin Lab 3.1: Working with Amazon S3

58

It is now time to start the lab.



Lab debrief: Key takeaways

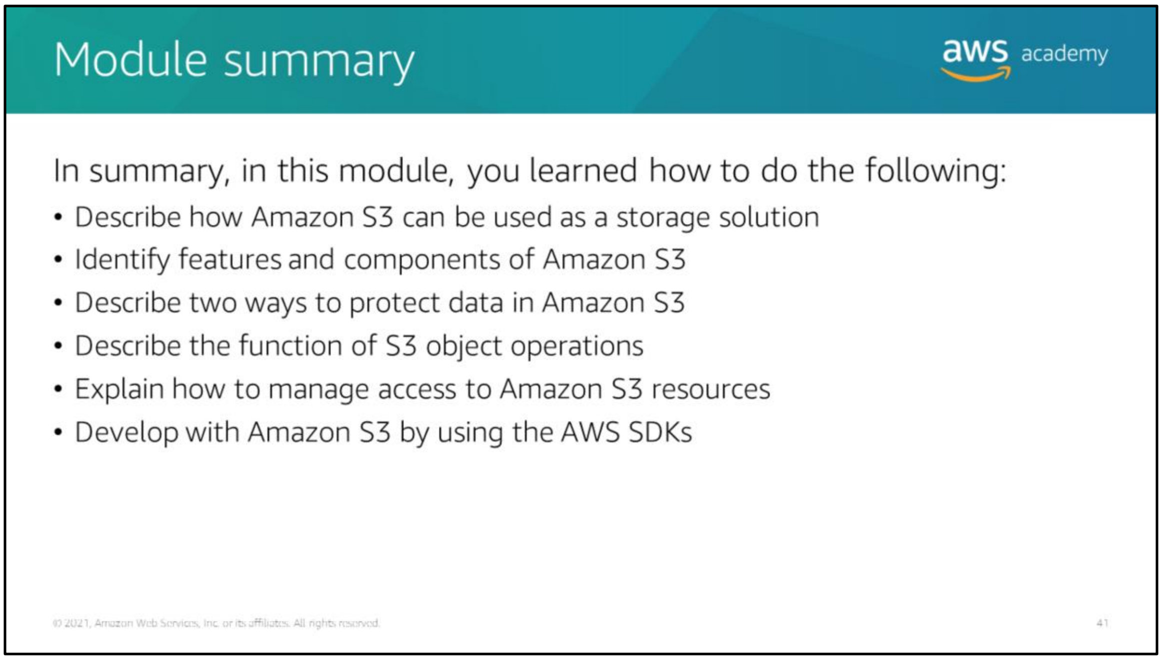
Your educator might choose to lead a conversation about the key takeaways from this lab after you have completed it.

Module 3: Developing Storage Solutions

Module wrap-up

aws academy

It's now time to review the module and wrap up with a knowledge check and discussion of a practice certification exam question.

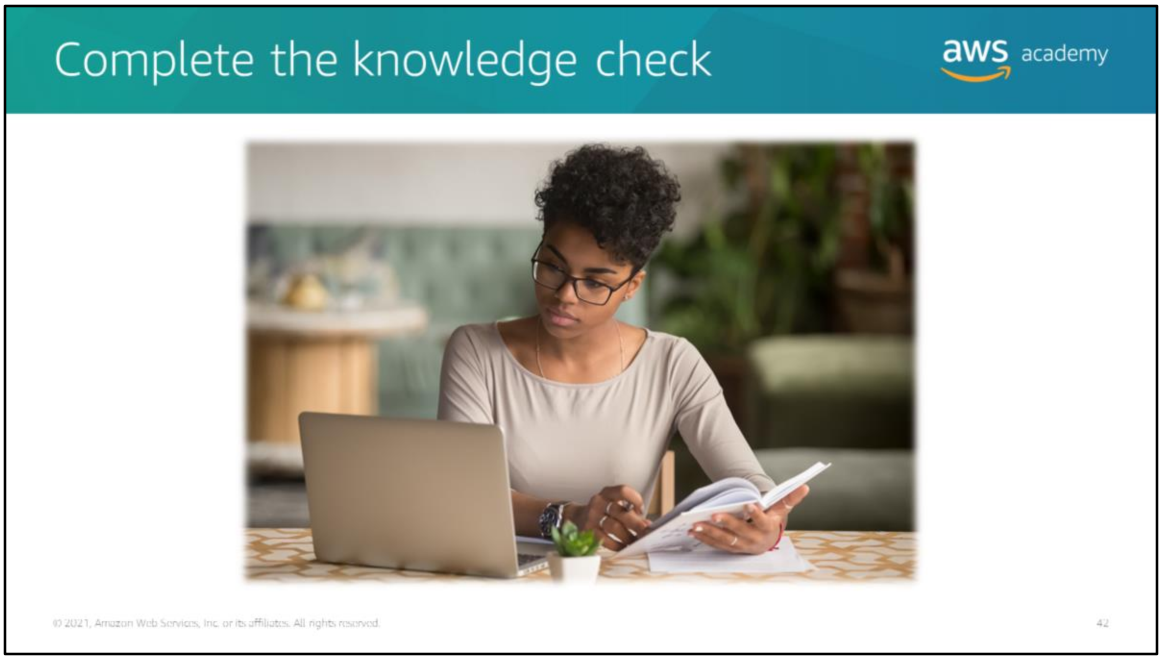


Module summary

In summary, in this module, you learned how to do the following:
- Describe how Amazon S3 can be used as a storage solution
- Identify features and components of Amazon S3
- Describe two ways to protect data in Amazon S3
- Describe the function of S3 object operations
- Explain how to manage access to Amazon S3 resources
- Develop with Amazon S3 by using the AWS SDKs

In summary, in this module, you learned how to do the following:

- Describe how Amazon S3 can be used as a storage solution
- Identify features and components of Amazon S3
- Describe two ways to protect data in Amazon S3
- Describe the function of S3 object operations
- Explain how to manage access to Amazon S3 resources
- Develop with Amazon S3 by using the AWS SDKs

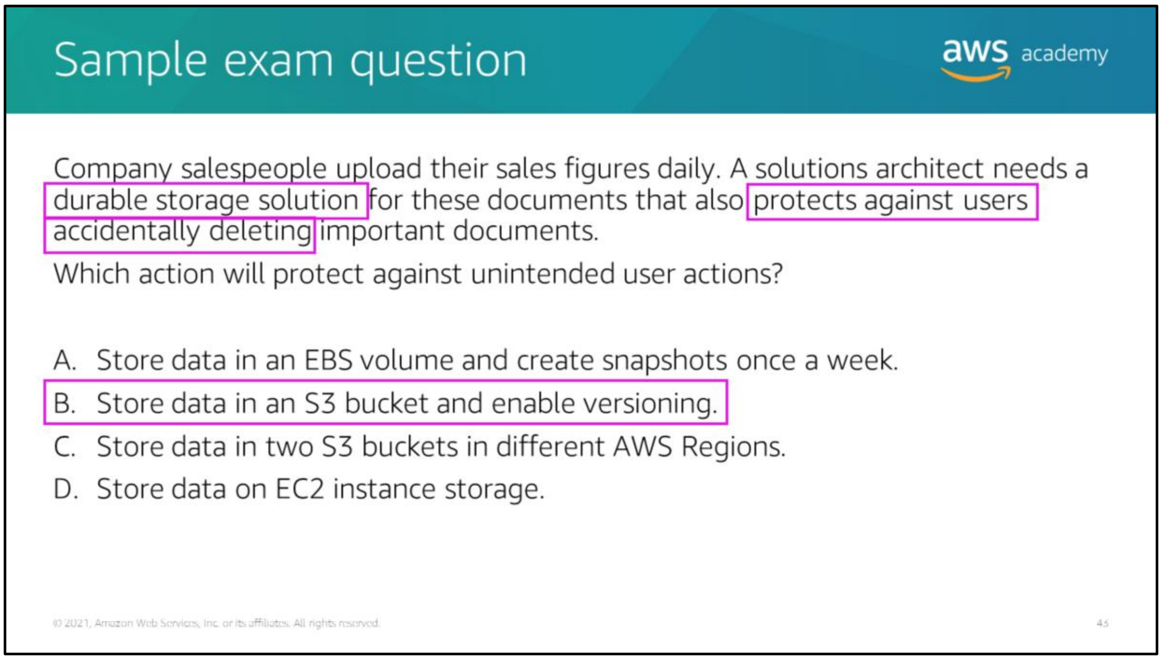


Complete the knowledge check

aws academy

42

It is now time to complete the knowledge check for this module.



Sample exam question

Company salespeople upload their sales figures daily. A solutions architect needs a durable storage solution for these documents that also protects against users accidentally deleting important documents.

Which action will protect against unintended user actions?

A. Store data in an EBS volume and create snapshots once a week.

B. Store data in an S3 bucket and enable versioning.

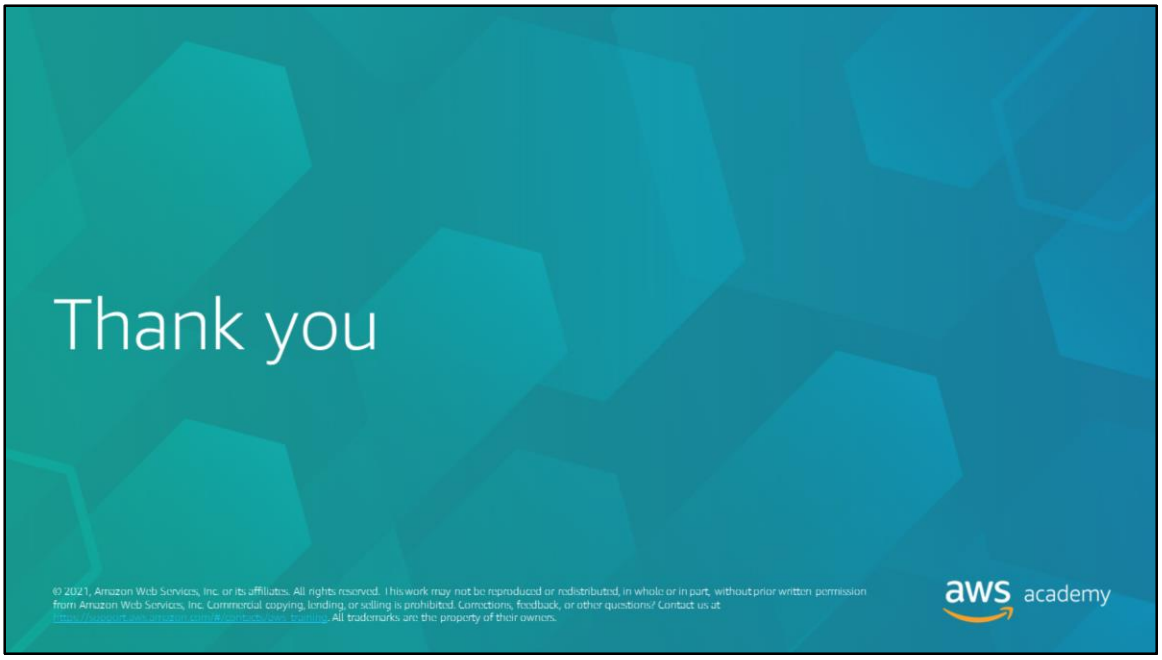
C. Store data in two S3 buckets in different AWS Regions.

D. Store data on EC2 instance storage.

Look at the answer choices and rule them out based on the keywords that were previously highlighted.

The correct answer is B. *Store data in an S3 bucket and enable versioning.*

Amazon S3 is a durable storage solution that the solutions architect can use to store files. To help ensure that users don’t delete important documents, the solutions architect can enable versioning on the bucket.



Thank you

Thank you for completing this module.