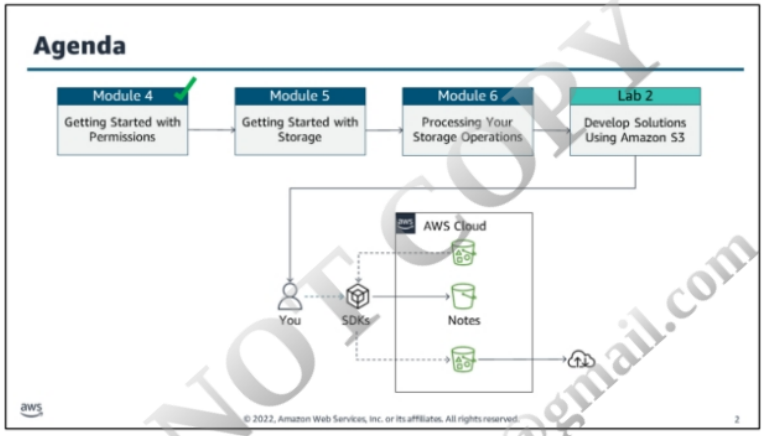
Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification

Module 5: Getting Started with Storage

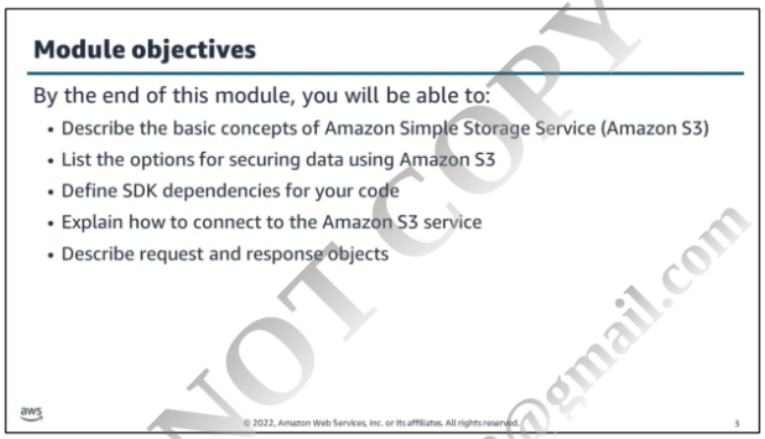


Module 5 – Compare feature sets and use cases for available AWS storage solutions. Learn basic concepts about Amazon Simple Storage Service (Amazon S3).

Module 6 – Learn how to work programmatically with Amazon S3 and deploy a static website to Amazon S3.

Lab 2 – Identify the appropriate AWS solutions for application workloads for big data.

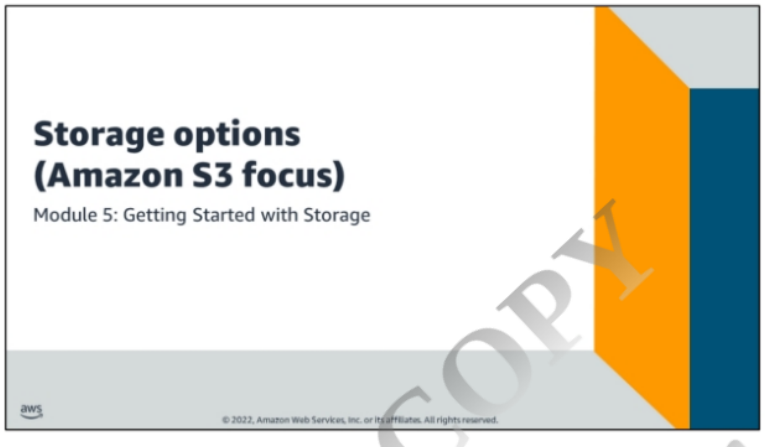
© 2022 Amazon Web Services, Inc. or its affiliates. All rights reserved.

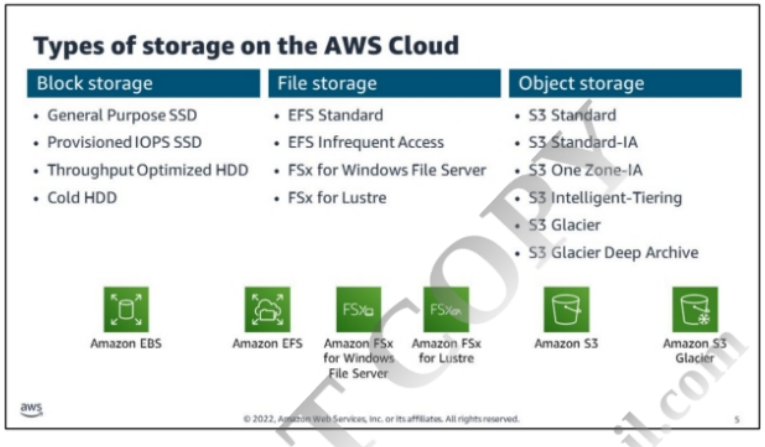


This module covers Amazon S3 and its basic components. You will review Amazon S3 use cases and how they support the application that you are developing.

Soon you will be coding. In this module, you will learn how to configure your development environment for working with Amazon S3 by using the AWS SDKs and the AWS CLI.

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.





AWS storage solutions consist of a portfolio of services so that you can store, access, govern, and analyze your data.

AWS supports the following key storage types.

- **Block storage**
  - Raw storage
  - Data organized as an array of unrelated blocks
  - Host file system places data on disk

  Example: Hard disks, storage area network (SAN), storage arrays

  Storage service: Amazon Elastic Block Store (Amazon EBS)

- **File storage**
  - File (serving) system manages unrelated data blocks
  - Native file system places data on disk

  Example: Network attached storage (NAS) appliances, Windows file servers

  Storage services: Amazon Elastic File Store (Amazon EFS), Amazon FSx for Windows File Server, and Amazon FSx for Lustre

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

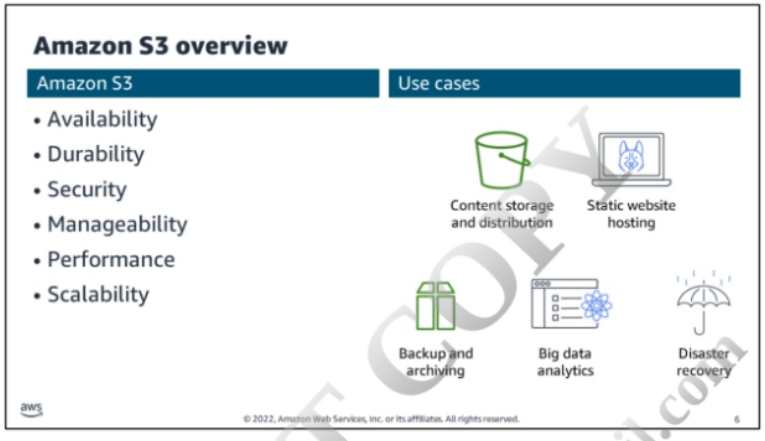
AWS Training and Certification Module 5: Getting Started with Storage

- **Object storage**
  - Stores virtual containers that encapsulate the data, data attributes, metadata, and object IDs
  - API access to data
  - Metadata-driven, policy-based, and so on.

  Example: Ceph, OpenStack Swift

  Storage service: Amazon Simple Storage Service (Amazon S3)

© 2022 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Amazon S3 at a glance

- **Durability**
  - Designed for 11 9's (99.999999999%) of durability
- **Availability**
  - Designed for 99.99% availability
- **Scalability**
  - Unlimited storage
  - Batch operations
- **Performance**
  - Parallel requests
  - Transfer acceleration
  - Multipart uploads
  - Read-after-create consistency

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

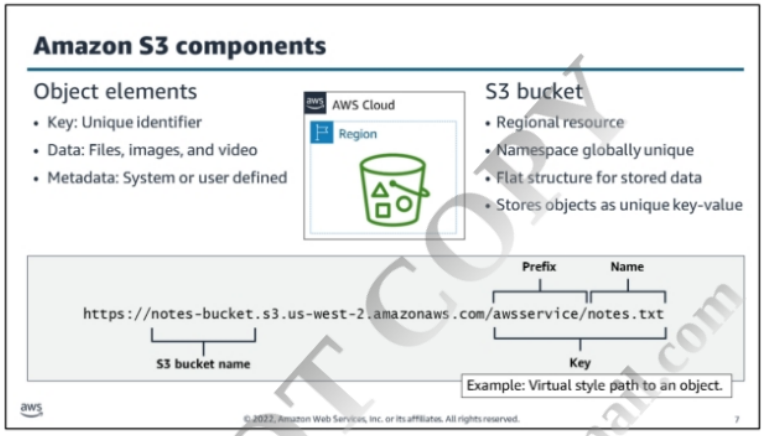
AWS Training and Certification Module 5: Getting Started with Storage

- **Manageability**
  - REST API
  - AWS SDKs
  - Prefixes/Tags
  - Event notifications
  - Storage classes
  - Lifecycle management
- **Security**
  - S3 Public Access block
  - Default encryption
  - Granular controls
  - Regulatory compliant
  - Secure at rest and at transfer

Amazon S3 provides developers and IT teams with highly secure, durable, and scalable object storage. You can use Amazon S3 as a storage solution for the following use cases:

- Content storage and distribution
- Static website hosting
- Backup and archiving
- Big data analytics
- Disaster recovery

In the application, you will use Amazon S3 for hosting the website and storing content.



After you create an S3 bucket, you can store any number of data files inside the bucket. Data files are stored in S3 buckets as objects. An object can be any kind of file such as text, video, image, or another binary format.

Although a bucket is created in an AWS Region, the bucket is considered a global resource. This means that the bucket name must be unique across Amazon S3. After you name a bucket, you cannot change its name. If you are hosting a website on the bucket, consider naming your bucket the same as your domain name. This way, you can use DNS resolution through your domain provider (example: Amazon Route 53) to map an alias to the true bucket name.

Here are some rules for naming your bucket:

- Use 3–63 characters.
- Use only lowercase letters, numbers, and hyphens (-).
- Do not use a period (.) while using virtual hosted-style buckets with SSL. Buckets that have a period in the bucket name can cause certificate exceptions when accessed with HTTPS-based URLs.
- Do not use the underscore (_) character.

For more information, see "Bucket restrictions and limitations" in the *Amazon Simple Storage Service (S3) User Guide* (https://docs.aws.amazon.com/AmazonS3/latest/dev/BucketRestrictions.html).

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

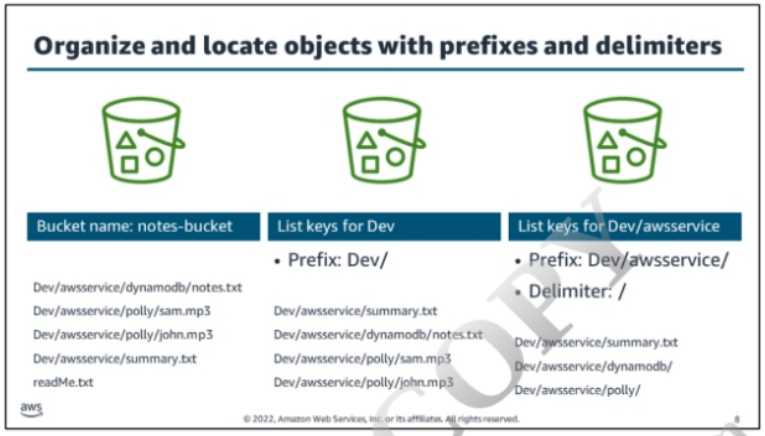
AWS Training and Certification Module 5: Getting Started with Storage

An object includes the following components:

- **Key** – Unique identifier such us a combination of path and file name. It is a way to reference the object.
- **Data** – Any type of file stored in an S3 bucket.
- **Metadata** – System metadata or user-defined metadata, such as creation date, version ID, and size.

S3 buckets are object storage and therefore do not have directory structures. To create a logical structure in your S3 bucket, use prefixes and delimiters. Prefixes are similar to directory names. You can group objects based on their prefix. Delimiters provide additional control when browsing the object hierarchy.

© 2022 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Objects in S3 buckets have no hierarchy. You can use prefixes (such as Dev) in key names to group similar items. To organize your keys and create a logical hierarchy, you can use delimiters (any string or character such as / or _) in key names.

If you use prefixes and delimiters to organize keys in a bucket, you can retrieve subsets of keys that match certain criteria. You can list keys by prefix. You can also retrieve a set of common key prefixes by specifying a delimiter. This is one mechanism for your application to classify and index the data. Prefixes and delimiters are also used to fine-tune security settings.

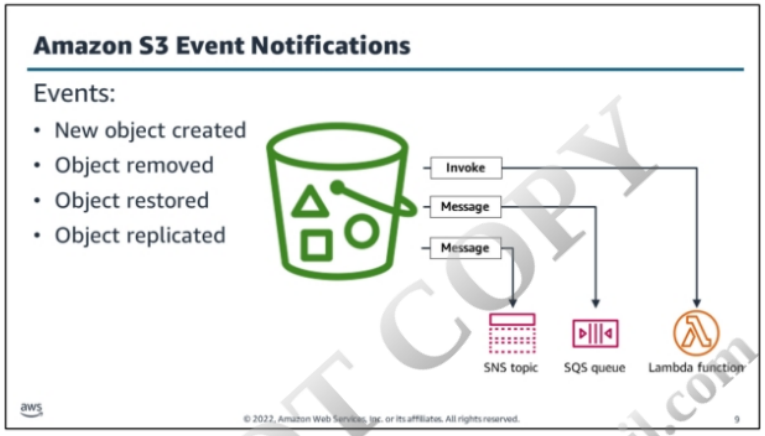
Example: To demonstrate how they work, consider this example. The bucket named notes-bucket contains objects relevant to services used in an awsservices project for the development team (denoted as Dev).

- To list keys related to the Dev team, specify a prefix of Dev/.
- To list keys only at the level of Dev/awsservices/, specify a prefix of Dev/awsservice/ and then specify the delimiter as /. The key Dev/awsservice/summary.txt is returned because it contains the prefix Dev/awsservice/ and does not contain the delimiter / after the prefix.
- To find a service for which files are available, list the keys by specifying a prefix of Dev/awsservice/ and a delimiter of /. Then, get common prefixes.

For more information, see "Organizing objects using prefixes" in the *Amazon Simple Storage Service (S3) User Guide* (https://docs.aws.amazon.com/AmazonS3/latest/userguide/ListingKeysUsingAPIs.html).

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 5: Getting Started with Storage



When certain events happen in your S3 bucket, you can use the Amazon S3 Event Notifications feature to receive notifications about those events. You can configure Amazon S3 Event Notifications to publish events to specific destinations, such as the following:

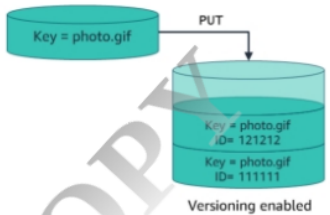
- Amazon Simple Notification Service (Amazon SNS) topics
- Amazon Simple Queue Service (Amazon SQS) queues
- AWS Lambda functions

In your project, an event will invoke Lambda functions.

© 2022 Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon S3 objects: Versioning

Version-enabled buckets
- Each object has a version ID.
- Object locking is supported.



Versioning
An object's version ID is part of the system-defined metadata.
By default, versioning is disabled in S3 buckets.
- In versioning-disabled buckets, an object has a version ID of null.
- In versioning-enabled buckets, each version of an object has a unique version ID.

An example of a version ID is
3/L4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY+MTRCxf3vjVBH40Nr8X8gdRQBpUMLUo

To prevent data from being changed, overwritten, or deleted, configure the Amazon S3 Object Lock feature.

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

Deleting object versions

DELETE – Inserts a delete marker in the bucket, and sets it as the current version of the object.

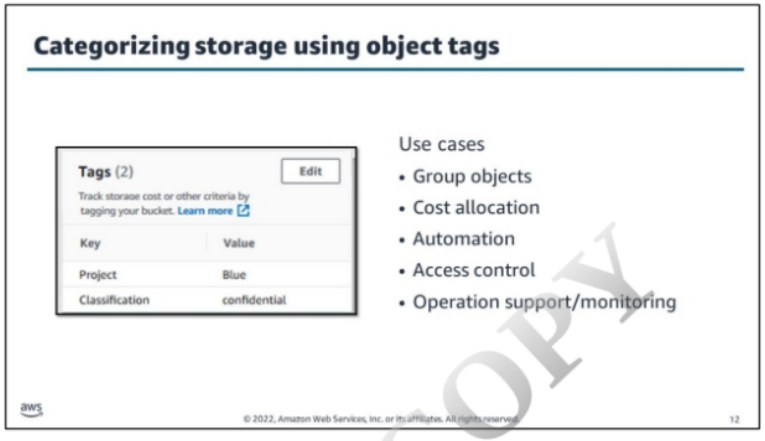
DELETE Object versionId – Deletes a specific object version permanently.

Photo.gif is marked as deleted.

Photo.gif version 121212 was permanently delete.

Sample Request: Deleting a specified version of an object

```
DELETE /photo.gif?versionId=121212 HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: wed, 26 Oct 2021 17:50:00 GMT
Authorization: AWS
AKIAIOSFODNN7EXAMPLE:xQE0diMbLRepdf3YB+FIEXAMPLE=
Content-Type: text/plain
Content-Length: 0
```



Tags are another way to manage your Amazon S3 objects and buckets. Tags are unique key-value pairs that attach to AWS resources, such as S3 buckets and objects, as metadata. You can create, update, delete, copy, or replace tags at any time during the life of an object.

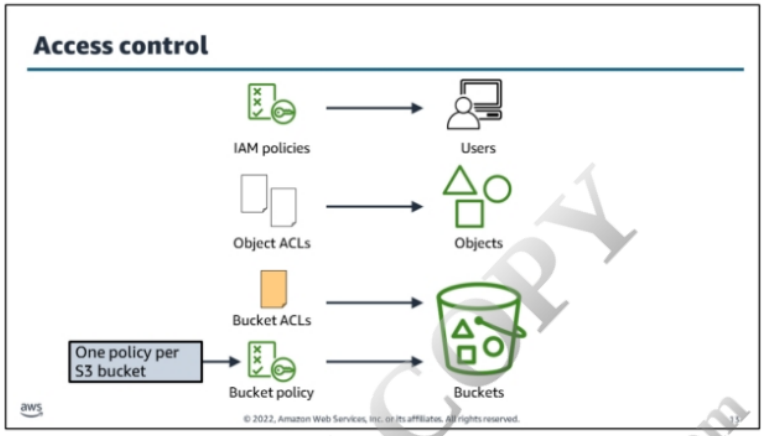
Use cases

The following are common use cases for tags:

- **Group objects** – Tag resources with unique business, compliance, or project identifiers.
- **Cost allocation** – Use tags in billing that are specific to your cost centers. Then, generate AWS tag-based reports to find the actual costs for each cost center.
- **Automation** – Tag resources for specific automation procedures, such as backup or replication.
- **Access control** – Create access control lists and restrict access.
- **Operation support and monitoring** – Use tags to identify key systems.

For more information, refer to the "Tagging Best Practices" technical paper (https://d1.awsstatic.com/whitepapers/aws-tagging-best-practices.pdf).

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.



By default, Amazon S3 bucket settings block public access to the bucket and its objects. This means that only the resource owner will be able to access the bucket and resources. You can then adjust access to your Amazon S3 buckets and objects. Block Public Access settings override bucket policies and object permissions.

Amazon S3 supports both user-based and resource-based access control.

- *IAM policies* are identity (user-based) policies. They attach to AWS Identity and Access Management (IAM) users, groups, or roles. IAM policies define what specific access an identity (role, user, or group) has to buckets and objects.
- *Access control lists (ACLs)* are legacy, resource-based policies. They attach to S3 buckets or objects and define the type of access given to an AWS account or group.
- *Bucket policies* attach to S3 buckets. A bucket policy can grant public or anonymous access to the bucket and its content. It can specify what actions a person or an application is allowed or denied from performing on the bucket and its objects. Alternatively, it can define access to specific S3 buckets or objects, grant access across AWS accounts, and allow or block access based on conditions.

IAM policies and S3 bucket policies are the recommended way of managing access control.

Example: S3 bucket policies

The S3 bucket policy grants read-only permission to an anonymous user

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


A bucket policy is a resource-based IAM policy.

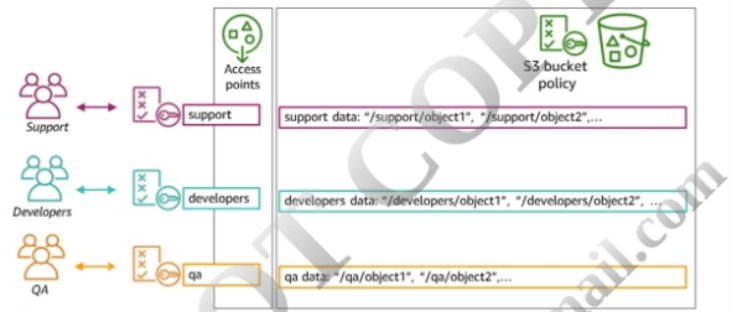
The policy has five key elements:
- **Resource** – Amazon Resource Name (ARN) defines the resource to which you allow or deny permissions.
- **Action** – Operations allowed or denied for each resource.
- **Effect** – Allow or deny indicates whether the policy allows or denies access.
- **Condition** – Specifies the conditions when the policy is in effect.
- **Principal** – Defines who the policy applies to.

Add a policy to a bucket to grant AWS accounts or IAM users access permissions for the bucket and the objects in it.

Object permissions apply only to the objects that the bucket owner creates.

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

Amazon S3 access points

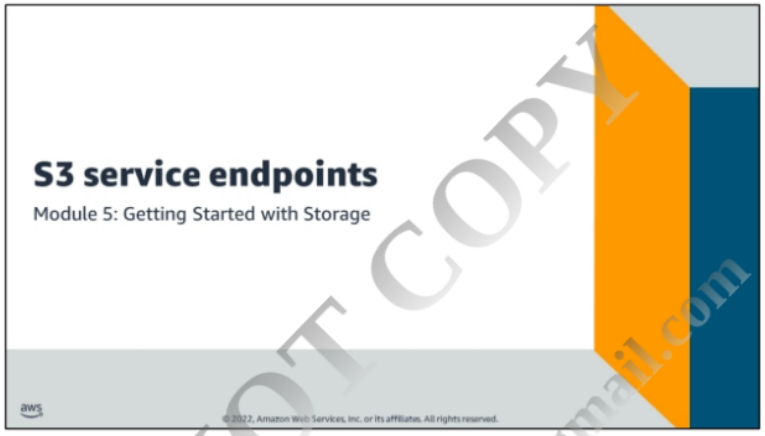


Amazon S3 access points are unique hostnames (named network endpoints). They are attached to buckets. Using access points, you can share access to S3 buckets with others.

Use cases include:
- **Decentralized teams** – Each group gets their own access point with tailored permissions and access.
- **Data lakes** – Provide granular control of teams accessing the data lake.
- **Cross-account data exchange** – Grant external users or users from other accounts access to Amazon S3 objects.
- **Centralized control** – Many access points, but one set of policies is used for storage management.

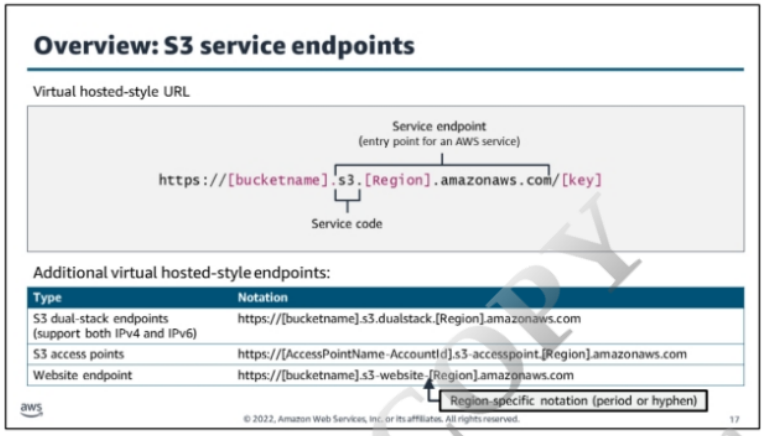
Each Amazon S3 access point is configured with an access policy specific to the use case. Every access point is associated with a bucket. The access point contains a network origin control and a Block Public Access control.

For example:
- Create an access point with a network origin control that permits storage access only from your virtual private cloud.
- Create an access point with the access point policy configured to allow access only to objects with a defined prefix, such as support.



Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

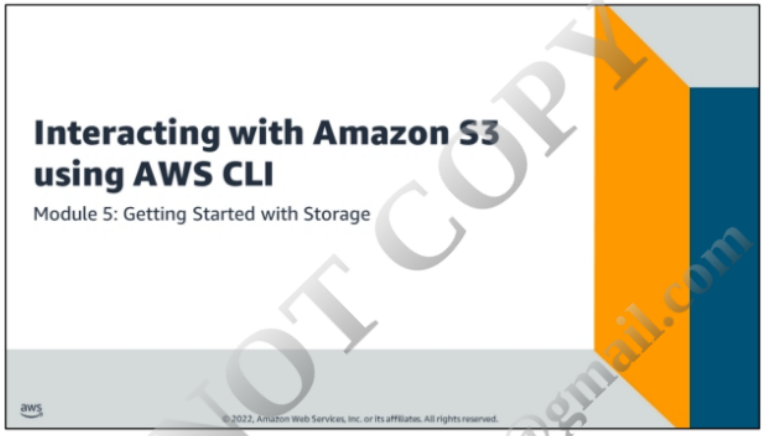
AWS Training and Certification Module 5: Getting Started with Storage



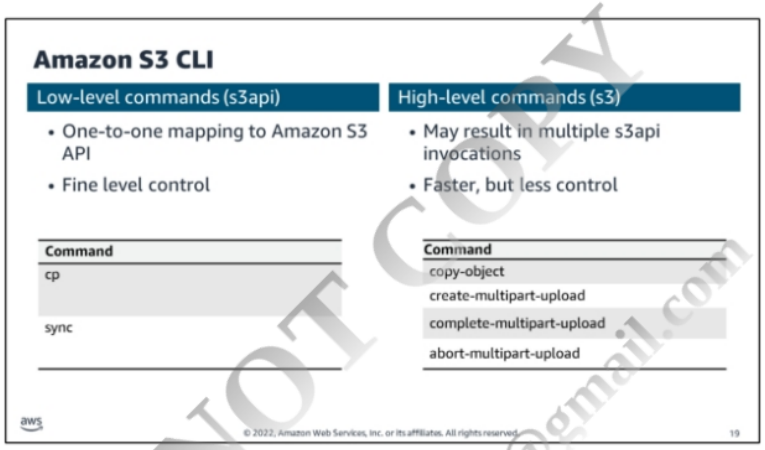
Amazon S3 supports programmatic access to S3 buckets and objects. When connecting to an AWS service programmatically, use an AWS service endpoint. This endpoint is the URL of the entry point for an AWS service. The example illustrates a virtual style path to an object. Accessing the root-level bucket is similar to the following URL: https://[*bucketname*].s3.[*region*].amazonaws.com

URL endpoints can vary. For example, the Amazon S3 dual-stack endpoints must specify the AWS Region. AWS SDKs and AWS CLI automatically determine and use the service's default endpoint for the AWS Region based on the selected service and the specified region. However, you can specify alternative endpoints for the request, such as an endpoint within an Amazon Virtual Private Cloud (Amazon VPC).

Note: Even though virtual hosted-style URLs are used to access Amazon S3 resources, some services require an alternative syntax to access a bucket. For example, S3://[*bucket-name*]/[*key*].



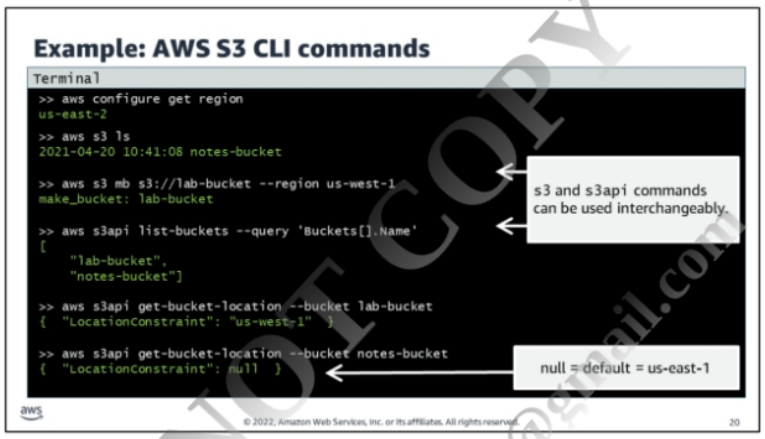
Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.



Amazon S3 defines two sets of Amazon S3 commands for working with AWS CLI: s3 and s3api.

The s3 set of commands are low-level commands written at the API level. The s3api commands are more object oriented and simplify managing Amazon S3 objects.

For example, if you want to upload a set of files from your local location to your S3 bucket, you would want to use the s3 commands cp or sync. These commands can copy large files and synchronize source and destination directories.

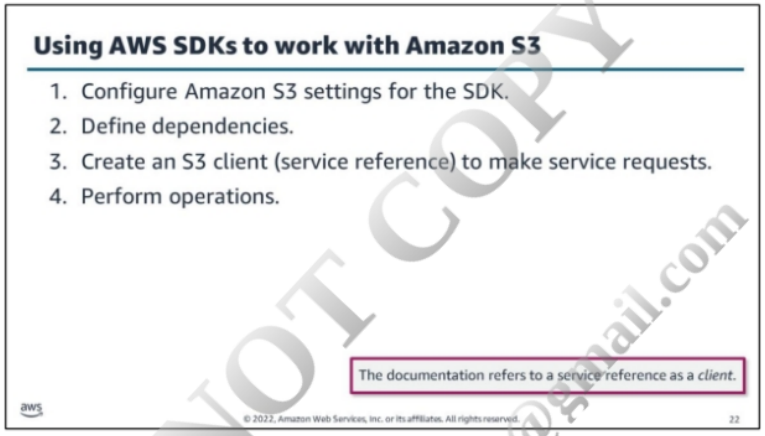


You can use the high-level and low-level commands interchangeably.

This example shows the following settings and operations:

- Check the region setting. It is `us-east-2`.
- List the buckets (all of them). Note that one bucket is listed.
- Create a bucket in `us-west-1`.
- Uses `s3api` to list the buckets by name. The response is a list of buckets in the format specified in your configuration (such as `yaml`, `json`, `text`).
- Get the bucket location for each bucket individually.
  Note that if you don't specify an AWS Region when creating a bucket, the bucket will be created in the default region. The `LocationConstraint` property value `null` indicates that the bucket was created in the default region, which is `us-east-1`.

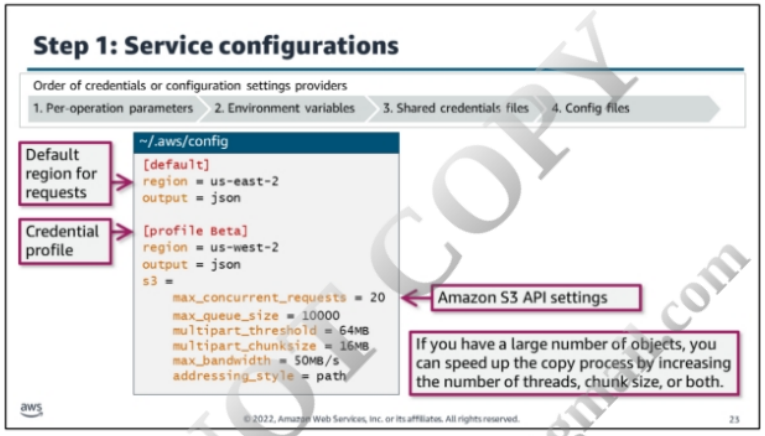
Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.



Steps for working with the AWS SDKs and Amazon S3:

1. Configure development environment for the SDK settings so that your code can interact with the AWS service.
   The AWS SDK must know the credentials so that it can sign and encrypt requests appropriately. It also requires configuration details, such as the AWS Region of the services so that it can determine the endpoints, version of the SDK used, and more.
2. Define dependencies for the AWS SDK, and bring in the packages and libraries that you need.
3. Create and configure a reference to a service object so that you can make requests.
   Note: In the AWS documentation, the Amazon S3 service reference is generally referred to as the *service client*. The service client refers to both low-level (client) and high-level S3 (resource) client interface reference.
4. Perform operations.

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.



To decipher the endpoints and successfully interact with AWS service APIs, the AWS SDKs and AWS CLI need information to perform the requested operation. For example, the SDK or tool needs information about the endpoints, logging options, Signature Version 4 support, and more.

This information is categorized as follows:

- **Configuration details**, which include settings required to process the request, such as the following:
  - API version used
  - Default region for request
  - Response format
  - Service-specific settings
- **Credentials information**, which includes sensitive information, such as password and access keys that are used to sign requests. These credentials verify who you are and whether you have permission to access the AWS resources.

Credential providers

You can use various methods to supply credentials and configuration information. These methods are called credential providers.

Some AWS SDKs and tools share the following common credential providers for finding the required information. The providers are listed in the order in which the AWS SDK or tool looks for credentials or configuration settings.

1. **Per-operation parameters** – You can specify certain settings on a per-operation basis. You can therefore change them as needed for each operation that you invoke.
   - For an SDK, these are set when instantiating an AWS client or service.
   - For AWS CLI, these are specified as parameter values.

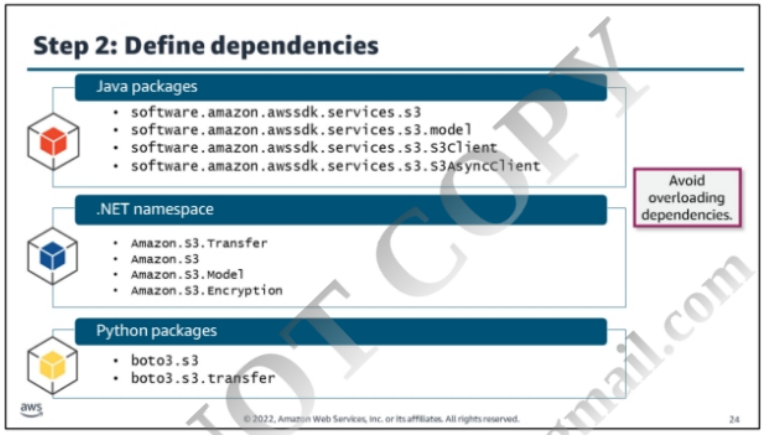
2. **Environment variables** – Operating system environment variables can store settings such as:
   - **Credentials** – AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY (recommended because it is recognized by all the AWS SDKs and AWS CLI except for .NET). Alternatively, AWS_ACCESS_KEY and AWS_SECRET_KEY (only recognized by Java SDK)
   - **Region** – AWS_REGION

3. **Shared config and credentials files** – AWS uses the config and credentials files to store settings for specific profiles. The files are stored at the default location shared by all AWS SDKs and the AWS CLI.
   - The **credentials** file keeps sensitive values such as secret access keys. It is located at ~/.aws/credentials on Linux, macOS, or Unix, or C:\Users\USERNAME\.aws\credentials on Windows. In the credentials file, you can have Amazon S3 settings that specialize Amazon S3 to your specific needs.
   - The **config** file keeps all the profile settings such as the AWS Region. The file is located at ~/.aws/config on Linux, macOS, or Unix, or C:\Users\USERNAME\.aws\config on Windows.

After you have configured the AWS SDK and AWS CLI with the information necessary to perform the requested operations, you are ready to work with the AWS services programmatically.

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 5: Getting Started with Storage



The SDKs offer simplified ways to build with services. However, to work with AWS services, you must configure the dependencies of the services that you will be using. Each AWS service has a service interface with methods for each action in the service API.

For Python, you define these dependencies by importing Boto3, which describes all services.

For the AWS SDKs for .NET or Java, import only the specific service packages and methods you need for your application.

Java packages

- **software.amazon.awssdk.services.s3** – APIs for creating Amazon S3 clients and working with buckets and objects
- **software.amazon.awssdk.services.s3.model** – Low-level API classes to create requests and process responses
- **software.amazon.awssdk.services.s3.S3Client** – For the S3 client

- **software.amazon.awssdk.services.s3.S3AsyncClient** – For the S3 asynchronous client

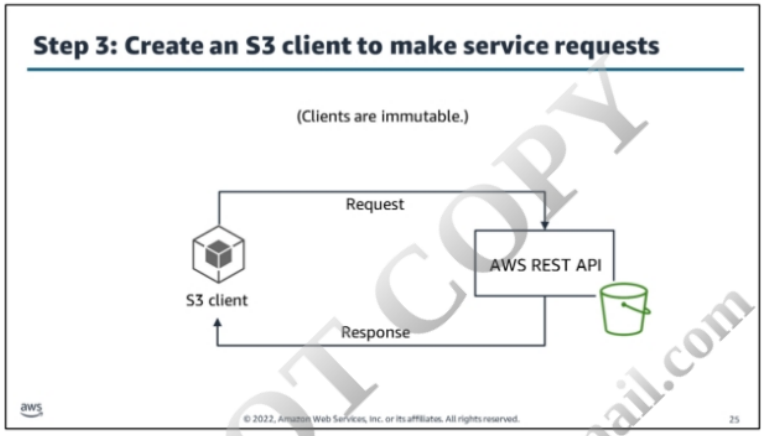
.NET namespace

- **Amazon.S3.Transfer** – High-level API for transferring data
- **Amazon.S3** – Implementation of low-level APIs for Amazon S3 REST multipart upload API
- **Amason.S3.Model** – Low-level API classes to create and process requests
- **Amazon.S3.Encryption** – Encryption client

Python packages

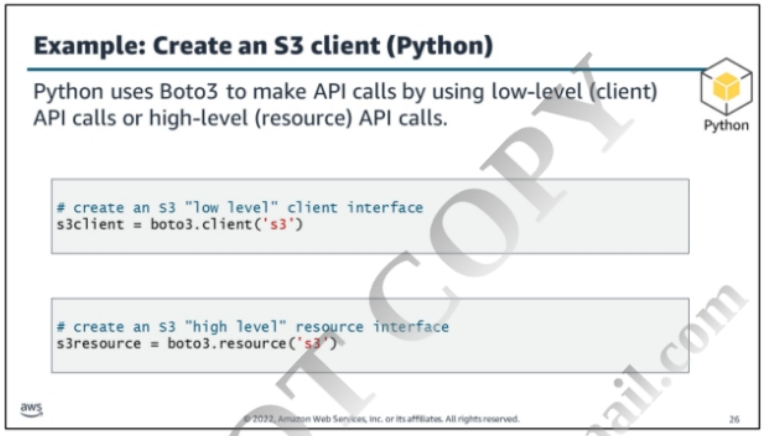
- **boto3.s3** – Low-level client representation of Amazon S3. It defines low-level API (client) and high-level API (resource).
- **boto3.s3.transfer** – For file transfers

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.



You already followed the best practice to create a shared configuration that is used as common settings across all developers. You also configured access control to the bucket and objects by using IAM policies, bucket policies, and ACLs.

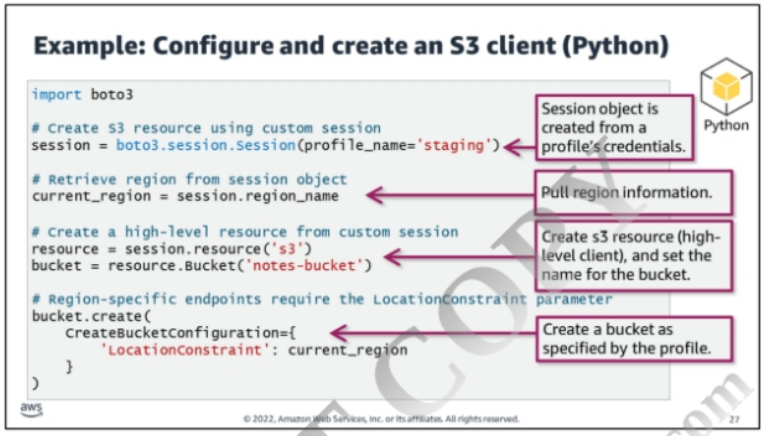
Before you can make calls to the service using the API, you must create an S3 client that references the S3 API. You can configure the S3 client before creating it. However, after the reference is created, it is immutable.



Remember that Python supports a low-level API (client) and a high-level API (resource).

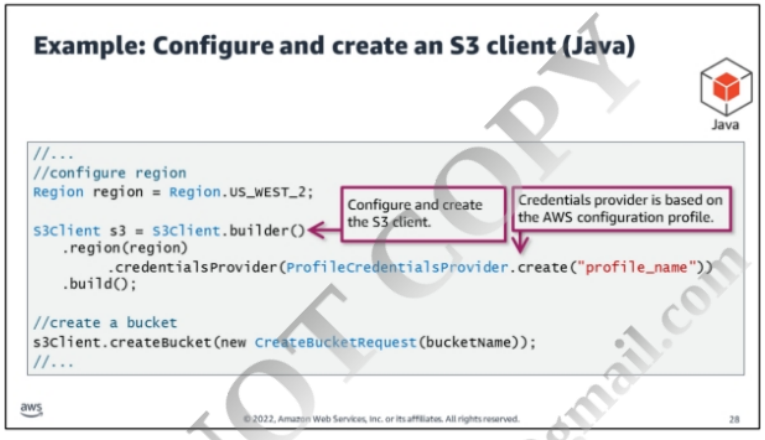
Creating a bucket by using a client returns a response as a dictionary. If you create a bucket using the resource API, you will receive a bucket instance.

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.



In this Python example, the configuration information of a specific profile is passed in as parameters when creating the client. However, to make the code more flexible and not dependent on a hardcoded AWS Region, class `boto3.session` is used. A *session* is an object that stores the configuration state, and you can pull this information to create the client.

After the client is created, you can use it to perform operations on the service, such as creating a bucket.



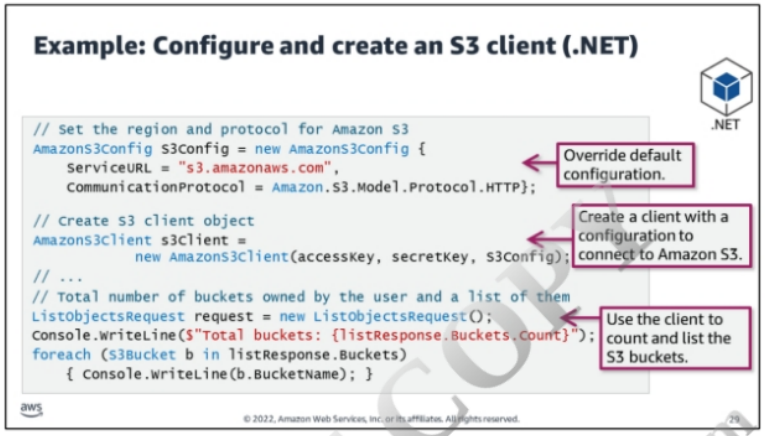
The AWS SDK for Java also provides high-level and low-level APIs.
When creating a client, use the factory-method **builder** to configure and build your client.

The code snippet illustrates the following:

- The AWS Region, which is defined in the instance profile, is overwritten with the region provided in the code.
- The SDK retrieves AWS credentials from the Instance Profile service.

In the above example, you notice that ***profile_name*** is passed to resolve credentials.

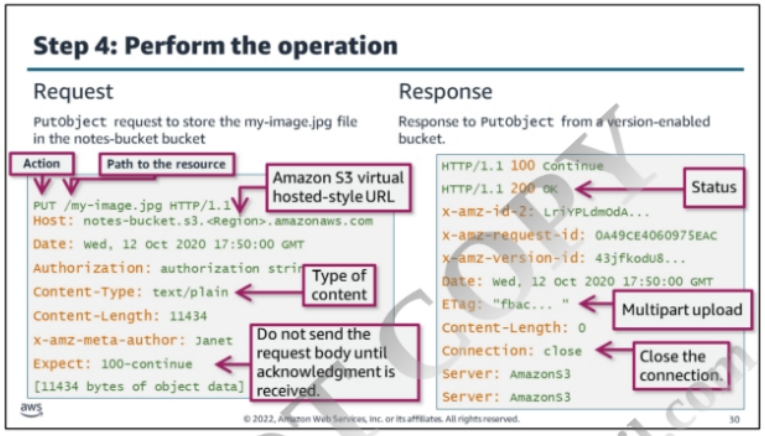
Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.



In this .NET example, you programmatically override the default configuration. You create a client based on the configuration.

Important: When the client is created, the access key and the secret key are passed in as parameters. Passing credentials through the code is not recommended because doing so creates a security risk.

When you have the client, you can interact with the service. Here, the client is used to fetch a count of the total number of buckets and list them by name.



Here is an S3 HTTP example of the PutObject request and the response from a version-enabled bucket.

Although Amazon SDKs do not expose many of the details of the API call, it is important to understand the type of information that the request and response provide. For example, headers associate metadata with an API request and response. Headers in a request are used to sign the requests, specify the content type, calculate the length, and more. A response uses headers and parameters to specify whether the connection is closed or the version ID of an object was uploaded into a version-enabled bucket.

ETag header

Sometimes it's necessary to review the metadata specified by the headers. For example, you can use the ETag header of a response for file and object comparison. When an object is uploaded to an S3 bucket, it receives an ETag. This ETag is a checksum that is calculated using the MD5 algorithm. If the upload is multipart, the ETag contains part of the information.

Another example is the x-amz-request-id and x-amz-id-2 headers. Amazon S3 generates these headers for each request that it processes and are useful when troubleshooting.

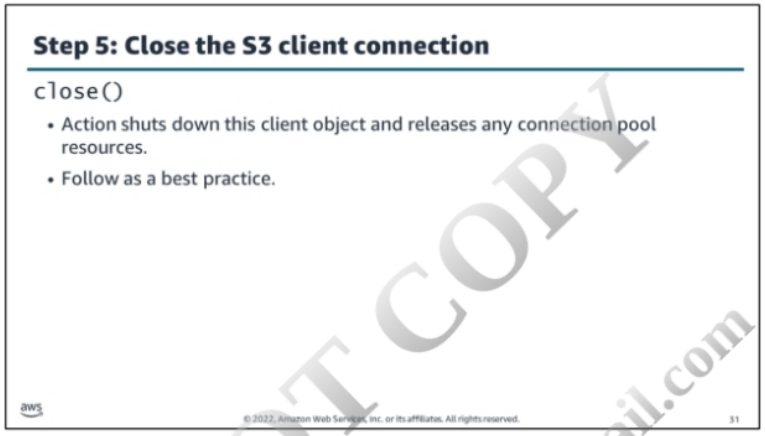
Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 5: Getting Started with Storage

For more information, see the following in the *Amazon Simple Storage Service (S3) API Reference*:

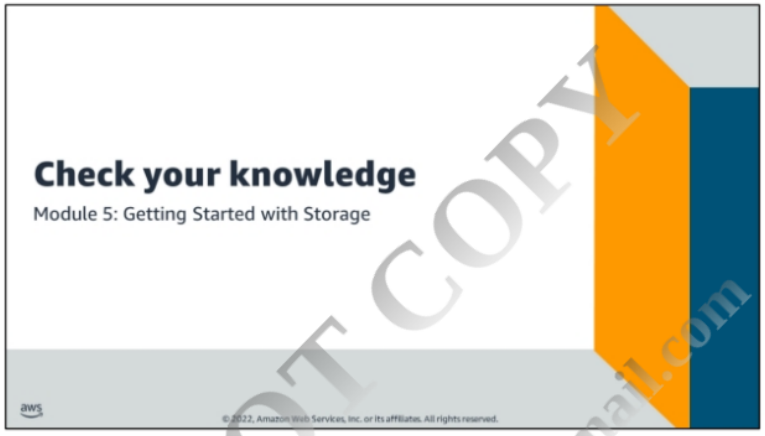
- "Common Request Headers" (https://docs.aws.amazon.com/AmazonS3/latest/API/RESTCommonRequestHeaders.html)
- "Common Response Headers" (https://docs.aws.amazon.com/AmazonS3/latest/API/RESTCommonResponseHeaders.html)

© 2022 Amazon Web Services, Inc. or its affiliates. All rights reserved.

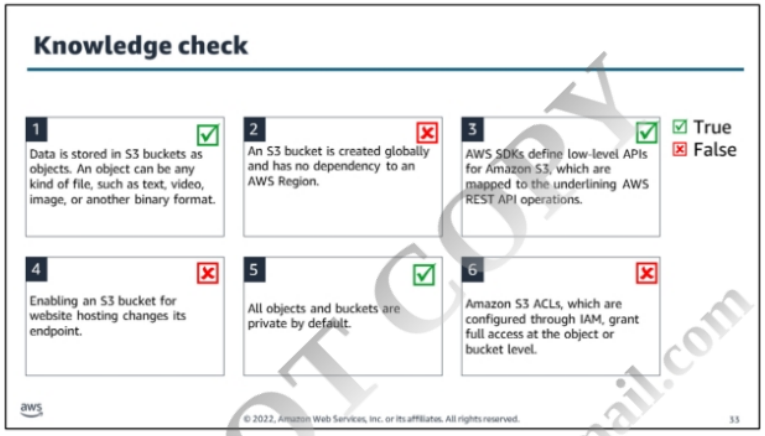


For best performance, close the client after you have completed the Amazon S3 operations. However, in most cases, a close request is not explicitly required because the SDKs automatically clean up, close, and remove the client from the connection pool.

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.



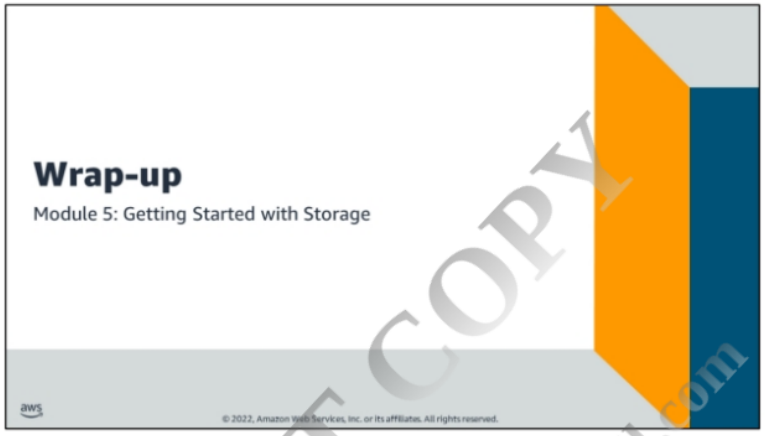
DO NOT COPY
sameersheik68@gmail.com



1. (True)
2. (False) Although an S3 bucket is a global resource, it is created in a specific AWS Region.
3. (True)
4. (False) Enabling web hosting just enables or adds the additional web endpoint, it doesn't change any existing ones.
5. (True)
6. (False) Amazon S3 access control lists are created and maintained from the **Permissions** tab of an S3 bucket.

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 5: Getting Started with Storage



© 2022 Amazon Web Services, Inc. or its affiliates. All rights reserved.

Module summary

You are now able to:

- Describe the basic concepts of Amazon Simple Storage Service (Amazon S3)
- List the options for securing data using Amazon S3
- Define SDK dependencies for your code
- Explain how to connect to the Amazon S3 service
- Describe request and response objects

 35

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

Thank you

Corrections, feedback, or other questions?
Contact us at https://support.aws.amazon.com/#/contacts/aws-training.
All trademarks are the property of their owners.