**Using Amazon S3 with the AWS CLI**

You can access the features of Amazon Simple Storage Service (Amazon S3) using the AWS Command Line Interface (AWS CLI).  
The AWS CLI provides commands for accessing Amazon S3:

**Note**

The AWS CLI supports copying, moving, and syncing from Amazon S3 to Amazon S3 using the *server-side* COPY operation provided by Amazon S3. This means that your files are kept in the cloud, and are *not* downloaded to the client machine, then back up to Amazon S3.

When operations such as these can be performed completely in the cloud, only the bandwidth necessary for the HTTP request and response is used.

### **Listing Your Buckets**

Use the [s3 ls](https://docs.aws.amazon.com/cli/latest/reference/s3/ls.html) command to list your buckets. Here are some examples of common usage.

The following command lists all buckets.

$ **aws s3 ls**

2018-12-11 17:08:50 my-bucket

2018-12-14 14:55:44 my-bucket2

The following command lists all objects and folders ([referred to in S3 as 'prefixes'](https://docs.aws.amazon.com/AmazonS3/latest/dev/ListingKeysHierarchy.html)) in a bucket.

$ **aws s3 ls *s3://bucket-name***

PRE path/

2018-12-04 19:05:48 3 MyFile1.txt

The previous output shows that under the prefix path/ there exists one file named MyFile1.txt.

You can filter the output to a specific prefix by including it in the command. The following command lists the objects in *bucket-name*/path (that is, objects in *bucket-name* filtered by the prefix path/).

$ **aws s3 ls *s3://bucket-name/path/***

2018-12-06 18:59:32 3 MyFile2.txt

### **Deleting a Bucket**

To remove a bucket, use the [s3 rb](https://docs.aws.amazon.com/cli/latest/reference/s3/rb.html) command.

$ **aws s3 rb *s3://bucket-name***

By default, the bucket must be empty for the operation to succeed. To remove a non-empty bucket, you need to include the --force option.

The following example deletes all objects and subfolders in the bucket and then removes the bucket.

$ **aws s3 rb *s3://bucket-name* --force**

**Note**

If you're using a versioned bucket that contains previously deleted—but retained—objects, this command does not allow you to remove the bucket. You must first remove all of the content.

**Managing Objects**

The high-level aws s3 commands make it convenient to manage Amazon S3 objects. The object commands include [s3 cp](https://docs.aws.amazon.com/cli/latest/reference/s3/cp.html), [s3 ls](https://docs.aws.amazon.com/cli/latest/reference/s3/ls.html), [s3 mv](https://docs.aws.amazon.com/cli/latest/reference/s3/mv.html), [s3 rm](https://docs.aws.amazon.com/cli/latest/reference/s3/rm.html), and [s3 sync](https://docs.aws.amazon.com/cli/latest/reference/s3/sync.html).

The cp, ls, mv, and rm commands work similarly to their Unix counterparts and enable you to work seamlessly across your local directories and Amazon S3 buckets. The sync command synchronizes the contents of a bucket and a directory, or two buckets.

**Note**

All high-level commands that involve uploading objects into an Amazon S3 bucket (s3 cp, s3 mv, and s3 sync) automatically perform a multipart upload when the object is large.

Failed uploads can't be resumed when using these commands. If the multipart upload fails due to a timeout or is manually canceled by pressing **Ctrl+C**, the AWS CLI cleans up any files created and aborts the upload. This process can take several minutes.

The cp, mv, and sync commands include a --grants option that you can use to grant permissions on the object to specified users or groups. Set the --grants option to a list of permissions using following syntax.

--grants *Permission*=*Grantee\_Type*=*Grantee\_ID*

[*Permission*=*Grantee\_Type*=*Grantee\_ID* ...]

Each value contains the following elements:

* *Permission* – Specifies the granted permissions, and can be set to read, readacl, writeacl, or full.
* *Grantee\_Type* – Specifies how to identify the grantee, and can be set to uri, emailaddress, or id.
* *Grantee\_ID* – Specifies the grantee based on *Grantee\_Type*.
  + uri – The group's URI. For more information.
  + emailaddress – The account's email address.
  + id – The account's canonical ID.

For more information on Amazon S3 access control, see [Access Control](https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingAuthAccess.html).

The following example copies an object into a bucket. It grants read permissions on the object to everyone and full permissions (read, readacl, and writeacl) to the account associated withuser@example.com.

$ **aws s3 cp file.txt s3://my-bucket/ --grants *read=uri=http://acs.amazonaws.com/groups/global/AllUsers full=emailaddress=user@example.com***

You can also specify a nondefault storage class (REDUCED\_REDUNDANCY or STANDARD\_IA) for objects that you upload to Amazon S3. To do this, use the --storage-class option.

$ **aws s3 cp file.txt s3://my-bucket/ *--storage-class REDUCED\_REDUNDANCY***

The s3 sync command uses the following syntax. Possible source-target combinations are:

* Local file system to Amazon S3
* Amazon S3 to local file system
* Amazon S3 to Amazon S3

$ **aws s3 sync <source> <target> [--options]**

The following example synchronizes the contents of an Amazon S3 folder named path in my-bucket with the current working directory. s3 sync updates any files that have a different size or modified time than files with the same name at the destination. The output displays specific operations performed during the sync. Notice that the operation recursively synchronizes the subdirectory MySubdirectory and its contents with s3://my-bucket/path/MySubdirectory.

$ **aws s3 sync . s3://my-bucket/path**

upload: MySubdirectory\MyFile3.txt to s3://my-bucket/path/MySubdirectory/MyFile3.txt

upload: MyFile2.txt to s3://my-bucket/path/MyFile2.txt

upload: MyFile1.txt to s3://my-bucket/path/MyFile1.txt

Typically, s3 sync only copies missing or outdated files or objects between the source and target. However, you can also supply the --delete option to remove files or objects from the target that are not present in the source.

The following example, which extends the previous one, shows how this works.

// Delete local file

$ **rm ./MyFile1.txt**

// Attempt sync without --delete option - nothing happens

$ **aws s3 sync . s3://my-bucket/path**

// Sync with deletion - object is deleted from bucket

$ **aws s3 sync . s3://my-bucket/path --delete**

delete: s3://my-bucket/path/MyFile1.txt

// Delete object from bucket

$ **aws s3 rm s3://my-bucket/path/MySubdirectory/MyFile3.txt**

delete: s3://my-bucket/path/MySubdirectory/MyFile3.txt

// Sync with deletion - local file is deleted

$ **aws s3 sync s3://my-bucket/path . --delete**

delete: MySubdirectory\MyFile3.txt

// Sync with Infrequent Access storage class

$ **aws s3 sync . s3://my-bucket/path --storage-class STANDARD\_IA**

You can use the --exclude and --include options to specify rules that filter the files or objects to copy during the sync operation. By default, all items in a specified folder are included in the sync. Therefore, --include is needed only when you have to specify exceptions to the --excludeoption (that is, --include effectively means "don't exclude"). The options apply in the order that's specified, as shown in the following example.

Local directory contains 3 files:

MyFile1.txt

MyFile2.rtf

MyFile88.txt

'''

$ **aws s3 sync . s3://my-bucket/path --exclude "\*.txt"**

upload: MyFile2.rtf to s3://my-bucket/path/MyFile2.rtf

'''

$ **aws s3 sync . s3://my-bucket/path --exclude "\*.txt" --include "MyFile\*.txt"**

upload: MyFile1.txt to s3://my-bucket/path/MyFile1.txt

upload: MyFile88.txt to s3://my-bucket/path/MyFile88.txt

upload: MyFile2.rtf to s3://my-bucket/path/MyFile2.rtf

'''

$ **aws s3 sync . s3://my-bucket/path --exclude "\*.txt" --include "MyFile\*.txt" --exclude "MyFile?.txt"**

upload: MyFile2.rtf to s3://my-bucket/path/MyFile2.rtf

upload: MyFile88.txt to s3://my-bucket/path/MyFile88.txt

The --exclude and --include options also filter files or objects to be deleted during an s3 syncoperation that includes the --delete option. In this case, the parameter string must specify files to exclude from, or include for, deletion in the context of the target directory or bucket. The following shows an example.

Assume local directory and s3://my-bucket/path currently in sync and each contains 3 files:

MyFile1.txt

MyFile2.rtf

MyFile88.txt

'''

// Delete local .txt files

$ **rm \*.txt**

// Sync with delete, excluding files that match a pattern. MyFile88.txt is deleted, while remote MyFile1.txt is not.

$ **aws s3 sync . s3://my-bucket/path --delete --exclude "my-bucket/path/MyFile?.txt"**

delete: s3://my-bucket/path/MyFile88.txt

'''

// Delete MyFile2.rtf

$ **aws s3 rm s3://my-bucket/path/MyFile2.rtf**

// Sync with delete, excluding MyFile2.rtf - local file is NOT deleted

$ **aws s3 sync s3://my-bucket/path . --delete --exclude "./MyFile2.rtf"**

download: s3://my-bucket/path/MyFile1.txt to MyFile1.txt

'''

// Sync with delete, local copy of MyFile2.rtf is deleted

$ **aws s3 sync s3://my-bucket/path . --delete**

delete: MyFile2.rtf

The s3 sync command also accepts an --acl option, by which you may set the access permissions for files copied to Amazon S3. The --acl option accepts private, public-read, and public-read-write values.

$ **aws s3 sync . s3://my-bucket/path --acl public-read**

As previously mentioned, the s3 command set includes cp, mv, ls, and rm, and they work in similar ways to their Unix counterparts. The following are some examples.

// Copy MyFile.txt in current directory to s3://my-bucket/path

$ **aws s3 cp MyFile.txt s3://my-bucket/path/**

// Move all .jpg files in s3://my-bucket/path to ./MyDirectory

$ **aws s3 mv s3://my-bucket/path ./MyDirectory --exclude "\*" --include "\*.jpg" --recursive**

// List the contents of my-bucket

$ **aws s3 ls s3://my-bucket**

// List the contents of path in my-bucket

$ **aws s3 ls s3://my-bucket/path/**

// Delete s3://my-bucket/path/MyFile.txt

$ **aws s3 rm s3://my-bucket/path/MyFile.txt**

// Delete s3://my-bucket/path and all of its contents

$ **aws s3 rm s3://my-bucket/path --recursive**