

Tutorial to upload large file to Amazon S3 Glacier using AWS CLI

Learning Objectives:

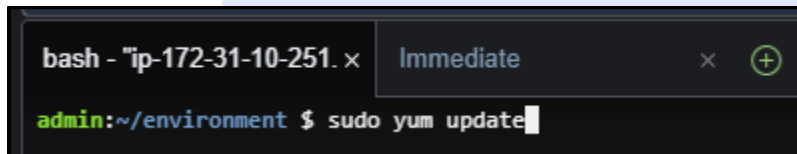
- Learn to create an Amazon S3 Glacier vault & upload a large file using AWS CLI.

Step 1:

We will use AWS Cloud9 EC2 development environment. To know how to create Cloud9 IDE visit our [Cloud9 IDE](#) tutorial.

In a terminal session in the AWS Cloud9 IDE, Run the **yum update** (for Amazon Linux) command to help ensure the latest security updates and bug fixes are installed.

For Amazon Linux:
sudo yum -y update



```
bash - "ip-172-31-10-251. x Immediate x +  
admin:~/environment $ sudo yum update
```

Step 2:

Create an Amazon S3 Glacier vault

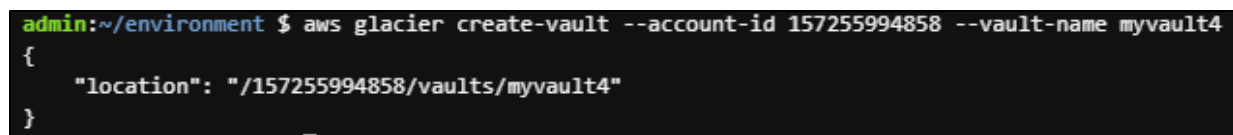
Create a vault with the create-vault command.

All S3 Glacier commands require an account ID parameter.

The hyphen character in the below command is to be replaced with your accountid.

aws glacier create-vault --account-id - --vault-name myvault

Your output should be similar to one shown below except your account-id.



```
admin:~/environment $ aws glacier create-vault --account-id 157255994858 --vault-name myvault4  
{  
  "location": "/157255994858/vaults/myvault4"  
}
```

Step 3:

Prepare a file for uploading

Let's create a file for the test upload. The following commands create a file named *largefile* that contains exactly 3 MiB of random data.

```
dd if=/dev/urandom of=largefile bs=3145728 count=1
```

Below is the output.

```
admin:~/environment $ dd if=/dev/urandom of=largefile bs=3145728 count=1
1+0 records in
1+0 records out
3145728 bytes (3.1 MB) copied, 0.0182619 s, 172 MB/s
```

dd is a utility that copies a number of bytes from an input file to an output file. The previous example uses the system device file /dev/urandom as a source of random data.

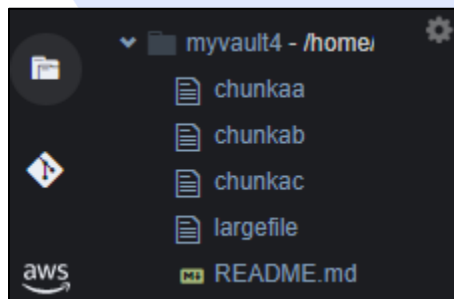
Next, split the file into 1 MiB (1,048,576 byte) chunks.

```
split -b 1048576 --verbose largefile chunk
```

Below is the output.

```
admin:~/environment $ split -b 1048576 --verbose largefile chunk
creating file 'chunkaa'
creating file 'chunkab'
creating file 'chunkac'
```

You can also see this files on your left side of panel



Step 4:

Initiate a multipart upload and upload files

In the below command, replace your account id in place of hyphen & then run the command.

aws glacier initiate-multipart-upload --account-id - --archive-description "multipart upload test" --part-size 1048576 --vault-name myvault Below is the output.

```
admin:~/environment $ aws glacier initiate-multipart-upload --account-id 157255994858 --archive-description "multipart upload test" --part-size 1048576 --vault-name myvault4
{
  "uploadId": "rVh75L6LC31RKUUvy1W3Ts9lhFji1CRmuSeWPtIvE-PXaTztM0iGGQ5ioHb-tCjodWSu0gPWWkrtOpOOIe7mNiJ_tRR",
  "location": "/157255994858/vaults/myvault4/multipart-uploads/rVh75L6LC31RKUUvy1W3Ts9lhFji1CRmuSeWPtIvE-PXaTztM0iGGQ5ioHb-tCjodWSu0gPWWkrtOpOOIe7mNiJ_tRR"
}
```

Before we move forward make sure to copy and save this upload id for future use.

upload id: rVh75L6LC31RKUUvy1W3Ts9lhFji1CRmuSeWPtIvE-PXaTztM0iGGQ5ioHb-tCjodWSu0gPWWkrtOpOOIe7mNiJ_tRR

S3 Glacier requires the size of each part in bytes (1 MiB in this example), your vault name, and an account ID to configure the multipart upload. The AWS CLI outputs an upload ID when the operation is complete. Save the upload ID to a shell variable for later use.

UPLOADID="19gaRezEXAMPLES6Ry5YYdqthHOC_kGRCT03L9yetr220UmPtBYKkOs sZtLqyFu7sY1_IR7vgFuJV6NtcV5zpsJ"

Next, use the upload-multipart-part command to upload each of the three parts.

Replace account-id in place of '-' hyphen. Also replace upload id in place of "\$UPLOAD"

Note:

Use notepad for editing following command make sure that command is in single line

aws glacier upload-multipart-part --upload-id \$UPLOADID --body chunkaa --range 'bytes 0-1048575/*' --account-id - --vault-name myvault

```
admin:~/environment $ aws glacier upload-multipart-part --upload-id rVh75L6LC31RKUUvy1W3Ts9lhFji1CRmuSeWPtIvE-PXaTztM0iGGQ5ioHb-tCjodWSu0gPWWkrtOpOOIe7mNiJ_tRR --body chunkaa --range 'bytes 0-1048575/*' --account-id 157255994858 --vault-name myvault4
{
  "checksum": "a99704c1d812f762d6011cd5589204e6b3542e71dd237d9f6179e1a8e3c44f9e"
}
```

aws glacier upload-multipart-part --upload-id \$UPLOADID --body chunkab --range 'bytes 1048576-2097151/*' --account-id - --vault-name myvault

```
admin:~/environment $ aws glacier upload-multipart-part --upload-id 6V0RL6igI5_i45v_eKJGFEXVcYr1IW7d
--body chunkab --range 'bytes 1048576-2097151/*' --account-id 157255994858 --vault-name myvault4
{
  "checksum": "34952a157fad97cf7bffc9a64f5f5b4266ac7b4ab052753a6c89bc84ea94adeb"
}
```

```
aws glacier upload-multipart-part --upload-id $UPLOADID --body chunkac --range
'bytes 2097152-3145727/*' --account-id - --vault-name myvault
```

```
admin:~/environment $ aws glacier upload-multipart-part --upload-id 6V0RL6igI5_i45v_eKJGFEXVcYr1IW7d
--body chunkac --range 'bytes 2097152-3145727/*' --account-id 157255994858 --vault-name myvault4
{
  "checksum": "0b23d53b8a465f075ebe2679b73e4cdbccf7d72522c7a4e4e50041dbbcc5304c"
}
```

Step 5:

Complete the upload

Amazon S3 Glacier requires a tree hash of the original file to confirm that all of the uploaded pieces reached AWS intact.

To calculate a tree hash, you must split the file into 1 MiB parts and calculate a binary SHA-256 hash of each piece. Then you split the list of hashes into pairs, combine the two binary hashes in each pair, and take hashes of the results. Repeat this process until there is only one hash left. If there is an odd number of hashes at any level, promote it to the next level without modifying it.

The key to calculating a tree hash correctly when using command line utilities is to store each hash in binary format and convert to hexadecimal only at the last step. Combining or hashing the hexadecimal version of any hash in the tree will cause an incorrect result.

To calculate a tree hash

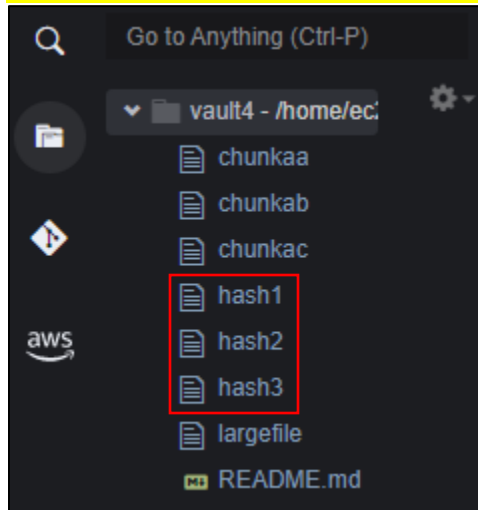
1. Split the original file into 1 MiB parts. This we have already done in step 3 above.
2. Calculate and store the binary SHA-256 hash of each chunk.

```
openssl dgst -sha256 -binary chunkaa > hash1
```

```
openssl dgst -sha256 -binary chunkab > hash2
```

```
openssl dgst -sha256 -binary chunkac > hash3
```

```
admin:~/environment $ openssl dgst -sha256 -binary chunkaa > hash1
admin:~/environment $ openssl dgst -sha256 -binary chunkab > hash2
admin:~/environment $ openssl dgst -sha256 -binary chunkac > hash3
admin:~/environment $
```



3. Combine the first two hashes and take the binary hash of the result.

```
cat hash1 hash2 > hash12
```

```
openssl dgst -sha256 -binary hash12 > hash12hash
```

4. Combine the parent hash of chunks aa and ab with the hash of chunk ac and hash the result, this time outputting hexadecimal. Store the result in a shell variable.

```
cat hash12hash hash3 > hash123
```

```
openssl dgst -sha256 hash123
```

After running 2nd command you will receive following outcome. Make sure to copy it.

```
admin:~/environment $ cat hash1 hash2 > hash12
admin:~/environment $ openssl dgst -sha256 -binary hash12 > hash12hash
admin:~/environment $ cat hash12hash hash3 > hash123
admin:~/environment $ openssl dgst -sha256 hash123
SHA256(hash123)= f49e8a5f5b523cf1d4a8f8b88e9dbd00ca125ef2e770be11461425cecff67fb8
```

SHA256(hash123)=f49e8a5f5b523cf1d4a8f8b88e9dbd00ca125ef2e770be11461425cecff67fb8

Copy the highlighted part from your console for further use.

TREEHASH=*f49e8a5f5b523cf1d4a8f8b88e9dbd00ca125ef2e770be11461425cecff67fb8*

Finally, complete the upload with the complete-multipart-upload command. This command takes the original file's size in bytes, the final tree hash value in hexadecimal, and your account ID and vault name.

```
aws glacier complete-multipart-upload --checksum $TREEHASH --archive-size 3145728 --upload-id $UPLOADID --account-id - --vault-name myvault
```

Below is the output.

```
{
  "archiveId": "m02Re8TD3qJjyEJKec9EeejGsoNFuD3NSMqohuMLopXTyRbyUT1g9rmRZxVWYI",
  "checksum": "f49e8a5f5b523cf1d4a8f8b88e9dbd00ca125ef2e770be11461425cecff67fb",
  "location": "/157255994858/vaults/myvault4/archives/m02Re8TD3qJjyEJKec9EeejGGrQZk2kFqW2RsupfxrtZk_V-2IJ1s0X11w"
}
```

Important NOTE:

Copy the "Archive ID" and save it safely. You will need archive id to empty the vault. Unless we empty the vault you cannot delete the vault.

You can also check the status of the vault using the describe-vault command.

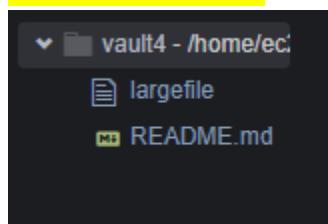
```
aws glacier describe-vault --account-id - --vault-name myvault
```

Below is the output.

```
admin:~/environment $ aws glacier describe-vault --account-id 157255994858 --vault-name myvault4
{
  "SizeInBytes": 0,
  "VaultARN": "arn:aws:glacier:ap-southeast-1:157255994858:vaults/myvault4",
  "NumberOfArchives": 0,
  "CreationDate": "2022-08-24T17:07:46.808Z",
  "VaultName": "myvault4"
}
```

Now it's safe to remove the chunk and hash files that you created.

`rm chunk* hash*` after running this command all files are remove



To delete Archive and empty the vault run the following command
`aws glacier delete-archive --vault-name xyz --account-id - --archive-id="*** archiveid ***"`

Replace your account id and vault name with "xyz" and paste your archive id here `<<archive-id>>` that we copy before.

NOTE:

After running commands it won't show any results it will take 1 day to update vault after that you can delete vault directly from console.

`aws glacier delete-vault --vault-name xyz --account-id -`

Delete the vault to avoid any AWS charges

Delete Cloud9 IDE avoid any AWS charges

Document Created by	Version
Vaibhav Deshpande	22-08-2022
Parth Rewoo	30-08-2022
Aditya k. Jaiswal	30-08-2022