PAUL NEWSON | 07/23/15

Migrating to Google Cloud Storage Nearline From Amazon Glacier

Migrating to Google Cloud Storage Nearline From Amazon Glacier

Table of contents:

Summary

Google Cloud Storage Nearline is a low cost storage class available in Google Cloud Storage which is attractive for archival workloads, such as cold storage and disaster recovery. Amazon Glacier also provides a storage service suitable for archival workloads, but retrieval latency in Amazon Glacier is measured in hours whereas Google Cloud Storage Nearline provides retrieval latency measured in seconds. This document contains best practices for migrating data currently stored in Amazon Glacier to Google Cloud Storage Nearline.

Overview

At a high level, migrating data from Amazon Glacier to Google Cloud Storage Nearline is simple:

1. Make the data available for download from Amazon Glacier
2. Copy the data to a bucket using the Nearline Storage class

Unfortunately, the details can be tricky.

First, there are two different ways data can be stored in Amazon Glacier: directly via the Amazon Glacier API, or via an Amazon S3 Object Lifecycle Management rule. How you make your data available for download is different depending on how it was stored. The sections “Migrating Data Stored Directly in Amazon Glacier” and “Migrating Data Stored in Amazon Glacier via Amazon S3” provide details on each of the these scenarios.

Second, you must be careful how quickly you make your data available for download, because it is easy to generate a very large bill from Amazon Web Services if you restore too quickly. The key to managing restore costs is to make your data available at a steady, consistent rate, because Amazon Web Services charges you according to *peak retrieval rate* reached during each month, even if you only use that peak rate for a few hours of the whole month.

Third, once your data has been made available for download, you have a limited amount of time to download it. Thus you must not make your data available at a rate faster than you are able to copy it to Cloud Storage Nearline, or you will end up having to make it available all over again.

Once your data is available for download, you must stage it in a location where it can be copied to Cloud Storage Nearline. For data stored directly in Amazon Glacier, we recommend using a Google Compute Engine instance for this purpose. For data stored in Amazon Glacier via Amazon S3, no additional staging is required.

Finally, the command line utility, **`gsutil`**, can be used to copy the data from its staging location (either a Compute Engine instance, or Amazon S3) to Cloud Storage Nearline. If you have a large amount of data to transfer, and if that data is staged in Amazon S3, then you may want to use the Storage Transfer Service instead of **`gsutil`**.

The sections that follow will provide details on each of the decision points and steps above, with the goal of enabling you to migrate your data from Amazon Glacier to Cloud Storage Nearline as efficiently and cost effectively as possible.

Prerequisites

Before embarking on the task of migrating data from Amazon Glacier to Google Cloud Storage Nearline, there are a few prerequisites that should be mentioned.

Create a Workspace and Staging Area

To migrate your data from Amazon Glacier to Cloud Storage Nearline, you need a place to run **`aws`** and **`gsutil`** commands, as well as a staging area to store the archives that you have downloaded from Amazon Glacier, but have not yet uploaded to Cloud Storage Nearline. Any computer with good network connectivity to both Amazon Glacier and Google Cloud Storage, adequate storage space, and properly installed and configured **`aws`** and **`gsutil`** utilities can be used for this purpose.

If you need some guidance on how to set up an appropriate workspace, please see Appendix A, which provides step by step instructions for creating and configuring a Compute Engine instance that will work well for following the rest of the instructions in this document.

Any time you see a command line that begins with the `cloud$` prompt in this document, it is assumed you are running this command from a Compute Engine instance configured as described in Appendix A, or an equivalent workspace. By contrast, when you see the `local$` prompt, these commands can be run from your local workstation.

Also note that in order to create a Compute Engine instance as described in Appendix A, you must have a Google Cloud Platform project with billing enabled, which is also a prerequisite for all the other steps in this document. The rest of this document assumes that you have done this, either on your own, or by following the instructions in Appendix A.

Create a Bucket Using the Nearline Storage Class

If you plan to migrate your data to Cloud Storage Nearline, you need a place to put it. One of the advantages of Cloud Storage Nearline is that it works exactly the same as all

the other storage classes in Google Cloud Storage, and therefore works with the exact same tools and libraries.

The one change you need to make to work with Cloud Storage Nearline is to specify the Nearline Storage class when you create the bucket. The following **`gsutil`** command will do that:

```
cloud$ gsutil mb -l US -c NL gs://nearline-migration
```

For illustration purposes, we have included the `-l US` flag here, even though that is the default if omitted, to show how to specify a location for the bucket at creation time. If you wish your data to reside in Europe or Asia you may specify -l EU or -l ASIA instead, or you may specify a region if you prefer a regional bucket. For a discussion of regional vs. continental buckets, and what scenarios are preferable for each, please see the discussion in the main Cloud Storage Nearline whitepaper, available here: https://cloud.google.com/files/GoogleCloudStorageNearline.pdf

Choose a Naming Convention

Amazon Glacier organizes data into vaults and archives. Vault names are chosen by humans, and therefore are probably relatively short and human readable. `ArchiveIds` are chosen by the Amazon Glacier service and are long random strings. While they are not very human readable, `ArchiveIds` do have the advantage that they are guaranteed to be unique within a vault, and that they are made up of characters that are generally filename-friendly.

Amazon Glacier also provides the option of associating an `ArchiveDescription` with each archive. This can be used to store a more human readable string with each archive. However, there's no guarantee that these archive descriptions are unique or that they can be used as filenames.

> Note: This section applies only to data stored directly in Amazon Glacier, not to data stored in Amazon Glacier via an Amazon S3 Lifecycle rule. If all your data has been stored via Amazon S3, then you may skip this section.

Cloud Storage Nearline does not impose long random strings for the names of each object that you upload - the object names are provided by the user. This is a good thing, because it means you can name your objects in a way that makes sense for your application. But in order to migrate data from Amazon Glacier to Cloud Storage Nearline,

you need to decide how to map the names from Amazon Glacier to Cloud Storage Nearline.

One possible naming convention is to combine the vault name with the `ArchiveId` from Amazon Glacier to generate a string that is guaranteed to be unique. If an archive also has an `ArchiveDescription`, that can be saved as a custom metadata header, e.g. x-goog-meta-archive-description. This way no information is lost in the migration from Amazon Glacier to Cloud Storage Nearline. This is the convention that will be followed in the examples in this document, since it is guaranteed to be reliable.

While this convention is useful for migrating data from Amazon Glacier to Cloud Storage Nearline, there is still the open question of how to name new objects that you store in Cloud Storage Nearline that have never been stored in Amazon Glacier. You could choose to mimic Amazon Glacier's behavior by generating your own long random strings for object names, but that seems like a missed opportunity to choose a friendlier naming convention for your objects moving forward.

If you believe that your `ArchiveDescriptions` are in fact always present, always unique within a vault, and will be accepted as both filenames and object names, then you could choose to use those as your object names instead. This may have the advantage that your naming convention in Cloud Storage Nearline moving forward will match the `ArchiveDescriptions` you've used in Amazon Glacier, and thus your new and old data will share the same naming convention. However if you do choose this approach, be prepared to deal with some archives that did not follow your convention perfectly, since it was not enforced by Amazon Glacier, and thus duplications, missing data, and other problems may have slipped through. It is probably a good idea to retrieve a complete inventory of your vault (as described later in this document), and analyze the `ArchiveDescriptions` contained there to check for violations of your expectations, before embarking on this path.

Finally, if you have maintained an external (to Amazon Glacier) database of archives and vaults which contains additional metadata, you may be able to choose filenames and object names that are not based solely on the `ArchiveIds` or `ArchiveDescriptions` contained in Amazon Glacier. If you have such a database, you may have additional flexibility on your naming convention, and it is likely worth analyzing that metadata to see if you can do better than the generic suggestions provided here.

Begin Storing New Data in Cloud Storage Nearline

Since the purpose of this document is to help you migrate existing archival storage from Amazon Glacier to Cloud Storage Nearline, we assume that you have already made the decision to use Cloud Storage Nearline, or are seriously considering doing so, and that you have existing archival data stored in Amazon Glacier.

Any significant migration of data from one storage service to another will take a non-trivial amount of time. During this migration period, you will be in a situation where you will have some data in Cloud Storage Nearline, and some data in Amazon Glacier. There are probably changes you need to make to your existing systems and processes to handle this period of time when you have data in both services. You must make these changes and be prepared for this state *before* you begin migrating data.

Since you need to be prepared for having data in both systems at the same time, and since you must do that before starting any migration, and since storing data in Amazon Glacier for a short period of time is not cost effective (there is a 90 day minimum charge, plus the potentially even larger cost of retrieving the data), it makes sense to begin storing all new archival data in Cloud Storage Nearline. By doing this, you ensure that your systems and processes have been updated to deal with having data in both places, and you are not making your migration effort more difficult with each passing day.

Migrating Data Stored Directly in Amazon Glacier

The instructions in this section apply to data that has been stored directly in Amazon Glacier. It does not apply to data that has been stored Amazon Glacier via an Amazon S3 Lifecycle rule. If all of your data has been stored via Amazon S3, the instructions in this section do not apply to your scenario, and you should skip ahead to the “Migrating Data Stored in Amazon Glacier via Amazon S3” section below.

Get an Inventory of Archives

The first challenge with retrieving data stored directly in Amazon Glacier is to figure out what data you have in Amazon Glacier. In Amazon Glacier parlance, “Archives” are stored in “Vaults”. For the purposes of this document, we are going to assume that you only have a single Vault named “test-vault”. To see what a Vault contains, you can request that Vault’s inventory. Unfortunately, inventories can be up to 48 hours out of date. For this reason Amazon suggests you keep a record of what you uploaded to Amazon Glacier in a database outside of Amazon Glacier [1]. However, if you have followed the suggestion in the previous section to begin writing new data to Cloud Storage Nearline instead of

Amazon Glacier, you need only wait 48 hours before you know you have a complete inventory, since no new archives are being written to the vault.

If you need a vault inventory you can use the **aws** command line to get it. First, create a file (in this example, `inventory-job.json`) containing the following JSON:

```
{
    "accountId": "-",
    "vaultName": "test-vault",
    "jobParameters": {
        "Type": "inventory-retrieval"
    }
}
```

You should of course replace "`test-vault`" with the name of your vault. Then, you can use that file to create an inventory retrieval job:

```
cloud$ aws glacier initiate-job --cli-input-json "`cat inventory-job.json`"
{
    "location": "/26[...]/vaults/test-vault/jobs/hLMG[...]",
    "jobId": "hLMG[...]"
}
```

> Note: `JobIds` and `ArchiveIds` are long strings of random characters. The actual values used in these examples are not relevant, since you will have different `JobIds` and `ArchiveIds`. Thus, to make this document easier to read, in these examples, we show only the first few characters, followed by `[...]` to represent the rest of the random characters. The same convention has been applied to some other fields, such as checksums where doing so makes this document more readable.

This creates the inventory retrieval job, but unfortunately, the inventory is not immediately available. You must wait until the job completes, which like all Amazon Glacier jobs, may take 3-5 hours. You can check the status of the inventory retrieval job like this:

```
cloud$ aws glacier describe-job \
    --account-id - \
    --vault-name 'test-vault' \
```

```
    --job-id hLMG[...]
{
    "InventoryRetrievalParameters": {
        "Format": "JSON"
    },
    "VaultARN": "arn:aws:glacier:us-west-2:26[...]:vaults/test-vault",
    "Completed": false,
    "JobId": "hLMG[...]",
    "Action": "InventoryRetrieval",
    "CreationDate": "2015-05-08T20:44:42.706Z",
    "StatusCode": "InProgress"
}
```

Note that you need the long random string that is the job-id in order to check status. If you happen to misplace this string, you can see what jobs you have in progress like so:

```
cloud$ aws glacier list-jobs \
    --account-id - \
    --vault-name test-vault
{
    "JobList": [
        {
            "InventoryRetrievalParameters": {
                "Format": "JSON"
            },
            "VaultARN":
"arn:aws:glacier:us-west-2:[...]:vaults/test-vault",
            "Completed": false,
            "JobId": "hLMG[...]",
            "Action": "InventoryRetrieval",
            "CreationDate": "2015-05-08T20:44:42.706Z",
            "StatusCode": "InProgress"
        }
    ]
}
```

Once your job completes, you must use the `getjoboutput` command to fetch the output:

```
cloud$ aws glacier get-job-output \
    --account-id - \
    --vault-name test-vault \
    --job-id hLMG[...] \
    inventory.out
{
    "status": 200,
    "acceptRanges": "bytes",
    "contentType": "application/json"
}
```

In this example, the vault contains only two archives. The JSON inventory is a little difficult to read, having no newlines, but if you look carefully, you'll find the two archive names in there. In this document, we've colored them **red** here so they are easier to see. Also contained in the inventory are the `ArchiveDescriptions` that were (optionally) associated with these Archives when they were created. These are shown in **green**.

```
cloud$ cat inventory.out
{"VaultARN":"arn:aws:glacier:us-west-2:26[...]:vaults/test-vault","Inventor
yDate":"2015-05-08T02:11:44Z","ArchiveList":[{"ArchiveId":"mvVq4T23zHClIUlM
gbAW7L_Y0mJ-qS2RTIf5CfnUmwd79PVB5OBCr_AxzFI6rI-x72ohcn54V24PWARg7SnQTVpo3H1
lBMFGK-zV0DdC-G30wM7kuS6FmCGsfhJAxjPpWZ57nLCxHw","ArchiveDescription":"1K",
"CreationDate":"2015-05-07T23:19:08Z","Size":1024,"SHA256TreeHash":"0e3d94a
694239297bdfb549c936162a1097dcab0f6faf768f6ae5775e12f7131"},{"ArchiveId":"f
PTlYn8ysZx3krtTQ6lDsjrecvjHlv3NjEkFnTG5I2PAglq9Je-UvXtwkdFpio_To5q8xJIIgs8B
kphTyJgWej0izy3sG82fXDE-mds601PFMKYaw7ZN1SA5czs614hI-KaKm3_WRw","ArchiveDes
cription":"1M","CreationDate":"2015-05-07T23:19:42Z","Size":1048576,"SHA256
TreeHash":"ca704d89500db25d41c6e0e471ee5189b10ec138cda3d729f774d2cf809ef715
"}]}
```

You now have a list of `ArchiveIds` in your Amazon Glacier vault that you want to migrate to Cloud Storage Nearline.

Make an Archive Available for Download

Now that we have `ArchiveIds` in hand, either via the inventory job, or via some external database of `ArchiveIds`, we are going to show how to create a retrieval job to make an archive available for download. First, create a file containing the following JSON, but using your vaultName and `ArchiveId`:

```
{
```

```
    "accountId": "-",
    "vaultName": "test-vault",
    "jobParameters": {
        "Type": "archive-retrieval",
        "ArchiveId": "mvVq[...]"
    }
}
```

Then you can initiate the actual retrieval with an initiate-job command:

```
cloud$ aws glacier initiate-job --cli-input-json "`cat retrieve-job.json`"
{
    "location": "/26[...]/vaults/test-vault/jobs/ZGAT[...]",
    "jobId": "ZGAT[...]"
}
```

Like all Amazon Glacier jobs, you should expect this job to take between 3 and 5 hours to complete.

Download the Archive to a Staging Area

When the retrieval job completes, you can get the actual bytes of the archive using a `getjoboutput` command:

```
cloud$ aws glacier get-job-output \
    --account-id - \
    --vault-name test-vault \
    --job-id ZGAT[...] \
    /data/test-vault/mvVq[...]
{
    "status": 200,
    "acceptRanges": "bytes",
    "contentType": "application/octet-stream",
    "checksum": "0e3d[...]"
}
```

As discussed in the “Choose a Naming Convention” section above, we are using the combination of the vault name and `ArchiveId` to name the local file to store the data from the archive.

If you are running these commands manually, as shown here, you effectively need to poll using `describe-job` or `list-jobs` to see when the job is complete. If you attempt to `get-job-output` before the job is complete, you will get an error.

Once your retrieval job completes, you have a limited amount of time to call `get-job-output` before all record of the job and its output disappears. Amazon Glacier's documentation states "A job ID will not expire for at least 24 hours after Amazon Glacier completes the job". [2] This means if you kick off a set of retrieval jobs on Friday, you shouldn't expect the output to still be available when you get back into the office on Monday.

Finally, if you perform these steps manually as shown here, after the `get-job-output` command completes, your data is now stored on the physical computer or virtual machine on which you ran that command. Choose that machine wisely. Since the purpose of this whitepaper is to show you how to migrate to Cloud Storage Nearline, we suggest you run the commands shown here on a Google Compute Engine instance, such as one configured as described in Appendix A. This will make the next step, of using **`gsutil`** to upload that data into Cloud Storage Nearline simpler and faster.

Upload Data to Cloud Storage Nearline

Now that you have the bytes of your archive available in a staging area, you can use the **`gsutil cp`** command to copy those bytes to Cloud Storage Nearline:

```
cloud$ gsutil \
    -h x-goog-meta-archive-description:1K
    cp \
    /data/test-vault/mvVq[...] \
    gs://nearline-migration/test-vault/mvVq[...]
```

Note that we have again followed the convention that the object name in Cloud Storage Nearline is the combination of the vault name and `ArchiveId` from Amazon Glacier. Additionally, we have added a custom metadata header to the object to capture the `ArchiveDescription` from Amazon Glacier.

Migrating in Batches

In the previous sections, we showed how to migrate a single archive from Amazon Glacier to Cloud Storage Nearline. If you are reading this document, it is probably a safe assumption that you have more than one archive you want to migrate. Unfortunately,

there is currently no tool or service capable of analyzing your Amazon Glacier storage, and automating all the steps shown above. Perhaps such a tool will exist in the future, but until then there are a few simple things you could do to process more than one archive at a time.

With some fairly straightforward scripting in a language such as python, you could parse your Amazon Glacier inventory JSON and break it up in to batches of roughly equal size, where each batch could be reasonably downloaded in a single day. With some equally straightforward scripting, you could write a program to call the **`aws`** command line to kick off the retrieval jobs for a batch of archives, and a similar program to kick off the actual retrieval of the archive data once it becomes available. With these relatively simple scripts, it would be possible to kick off the Amazon Glacier retrieval jobs early in the workday, and then kick off the downloads before the end of the workday, letting them run overnight to copy the archive data to your staging area.

Once you have the data in your staging area, no further scripting will be needed, because gsutil has some great features for performing uploads in parallel, and even performing simple "sync" operations. For example, assuming the directory /data/test-vault has a large number of files in it, some of which have already been uploaded to the gs://nearline-migration bucket, the following gsutil command will upload only new or changed files, and will not upload files that are already present in the destination bucket:

```
cloud$ gsutil -m rsync -r -c batch-log.txt \
    /data/test-vault gs://nearline-migration/test-vault
```

The **`-m`** option instructs gsutil to use multiple threads to copy, which will significantly increase performance if there are many files to copy. The **`-r`** option tells gsutil to recurse into any subdirectories below `/data/test-vault`. The **`-c`** option tells gsutil to continue to copy even if there is an error copying one of the files. Note that we did not specify the **`-d`** option, which means no objects will be deleted from the destination bucket.

Migrating Data Stored in Amazon Glacier via Amazon S3

It is also possible to store data in Amazon Glacier via an Amazon S3 Lifecycle rule. Since this document is about migrating from Amazon Glacier to Cloud Storage Nearline, it is assumed that the reader is already familiar with Amazon S3, has already configured such a lifecycle rule, and thus has objects stored in Amazon S3 with a Storage Class of

"Glacier", and is now concerned with how to migrate that data to Cloud Storage Nearline. This section will discuss how to migrate that data to Cloud Storage Nearline.

Get an Inventory of Archives

Amazon S3 objects that have been archived to Amazon Glacier are not visible via the Amazon Glacier API. Thus you cannot get a list of Archives as described in the previous section. Instead, you must use Amazon S3 tools to determine which objects are stored in Amazon Glacier.

Unfortunately, the **`aws s3`** command provides no way to check the storage class of an object, so the only way Amazon Web Services provides to see that an object has a storage class of "Glacier" is in the Amazon Web Services Console, or by making a direct call to the Amazon S3 API [5].

The **`aws s3`** command also does not provide a way to restore an object with a storage class of "Glacier" means that you will either have to write code to restore objects, or use the Amazon Web Services Console anyway.

For the purposes of this document, we will assume you are using the Amazon Web Services console to check the storage class of the objects you wish to migrate to Cloud Storage Nearline.

Make an Archive Available for Download

A "GET" request to an Amazon S3 Object with a Storage Class of "Glacier" will result in a response of `AccessDeniedException: 403 InvalidObjectState`. Before you can use either the Storage Transfer Service or `gsutil` to copy objects from Amazon S3 to Cloud Storage Nearline, you must "Restore" the objects.

The `aws s3` command line interface provides no way to check the Storage Class of an object, or to initiate a restore operation. Your options are to use the AWS Management Console, or write code that issues requests to the Amazon S3 API. This document describes how to use the AWS Management Console to restore objects.

To Restore an object with a Storage Class of "Glacier" so it can be copied using the Storage Transfer Service or `gsutil`, right click on that object in the Amazon S3 Console, and select "Initiate Restore". You will be prompted to specify "the number of days for which your archived data will be temporarily accessible". It is not possible to specify "forever". Make sure you choose a long enough period of time that you can be confident

that you will be able to transfer the data before it is again unavailable. Unlike with direct Amazon Glacier restore jobs, you are not limited to a fixed 24 hour window of availability.

This will kick off an Amazon Glacier Restore job, and in three to five hours the Storage Class of the object will still read "Glacier" but there will be an additional note that says "Restored until..." with the date that it will again become unavailable.

The Amazon S3 documentation also states "you should restore objects only for the duration you need because of the storage costs associated with the object copy. For pricing information, go to the Pricing section of the Amazon S3 product detail page."[4] It is unclear from the pricing page what price is charged for the restored copy. It is probably safe to assume that you will be charged the cost of Standard Storage for the restored copy and the cost of Glacier Storage for the copy still held in Amazon Glacier.

If you need more time to download the object, or if you no longer need it to be in a restored state, you can update the "Restored until... " date in the console to move the expiry date forward or backward in time.

Note: The Amazon Glacier Restore jobs that are initiated by Amazon S3 are not visible via `aws glacier list-jobs`.

Upload Data to Cloud Storage Nearline

Once you have Restored an object in Amazon S3, you can copy it using **`gsutil`** directly, with no intermediate staging area, as follows:

```
cloud$ gsutil cp s3://nearline-migration/1K gs://nearline-migration/1K
```

The bytes of the object will flow from Amazon S3, to the computer running **`gsutil`**, and then to Cloud Storage Nearline. Using a Compute Engine virtual machine that is colocated with the location of the destination bucket, such as one configured as described in Appendix A, will be the most efficient way to run this command.

Migrating in Batches

Unfortunately, without resorting to writing code that accesses the Amazon S3 API directly, there is no easy way to script batches of restore operations. However, once Amazon S3 objects have been restored, you can use gsutil to copy entire buckets in a single command:

```
cloud$ gsutil -m rsync -r -c -L batch-log.txt \
    s3://nearline-migration gs://nearline-migration
```

The **-m** option instructs gsutil to use multiple threads to copy, which will significantly increase performance if there are many files to copy. The **-r** option tells gsutil to recurse into subdirectories. The **-c** option tells gsutil to continue to copy even if there is an error copying one of the files. And the **-L** option tells gsutil to write a log of everything that happened in the batch copy operation to the file `batch-log.txt`. Note that we did not specify the **-d** option, which means no objects will be deleted from the destination bucket.

For large direct copies from Amazon S3 to Cloud Storage Nearline, there is a second option, which is capable of handling very large amounts of data in a massively parallel fashion - the Storage Transfer Service. The Storage Transfer Service is designed to move large quantities of data between Cloud Storage buckets in different regions, or in different Storage Classes, or to similarly move large quantities of data from Amazon S3 to Google Cloud Storage. You can find the the Storage Transfer Service in the Storage section of the left side navigation menu under “Transfer Service”.

On the Transfer Service page, you can click “New Transfer”, which takes you to a form where you can select “Amazon S3 Bucket” specify the bucket name and credentials to use, then specify a Google Cloud Storage bucket as the destination. Finally, you can choose to run this transfer once, right away, or run it on a recurring basis. You can also specify additional options, such as only transferring files with a particular prefix. Since this is a relatively new feature, you should check the UI for the complete set of options.

Understanding the Cost of Retrieving Data from Amazon Glacier

In the previous sections, we walked through how to retrieve your data from Amazon Glacier. But before you begin doing so, it is critically important to understand how Amazon Web Services charges for this service, or you could end up with an extremely unpleasant surprise on your Amazon Web Services bill.

The Amazon Glacier Pricing page states “Data Retrievals: Free †” [3]

Note the dagger. It's important. While there are some retrieval jobs that will not result in a charge on your bill, there are many important scenarios that could result in a surprisingly large charge on your bill.

If you follow the dagger to the footnote you find "You can retrieve up to 5% of your average monthly storage (pro-rated daily) for free each month. If you choose to retrieve more than this amount of data in a month, you are charged a retrieval fee starting at $0.01 per gigabyte." While this sounds straightforward, it is unfortunately oversimplified. There is a small "Learn More" link next to this statement which takes you to a Frequently Asked Questions page. [4] There you will find a question titled "How will I be charged when retrieving large amounts of data from Amazon Glacier?", the answer to which covers 11 paragraphs. Clearly there is something slightly more complex than "$0.01 per gigabyte" going on here.

> Note: Cloud Storage Nearline also charges $0.01 per gigabyte to read data. In the Nearline case, it really is that simple. If you read a gigabyte, you are charged $0.01 for reading that data.

The most important thing to understand about Amazon Glacier's billing model for retrievals is that it's based on retrieval *rate* not *volume*. Since a retrieval job takes between 3 and 5 hours, for the purposes of billing Amazon Web Services assumes the job will take 4 hours, and divides the size of the retrieval (in GB) by 4 to the retrieval *rate* in GB/hour. So if you submit a retrieval job for a 6 GB archive, the retrieval rate is 1.5 GB / hour.

However, Amazon Web Services groups together all the retrieval jobs submitted within an hour. So if you submitted 10 retrieval jobs, each for a 6 GB archive, the retrieval rate is now (10 * 6 GB) / 4 hours = 15 GB / hour.

> Note: It's not clear from the FAQ how Amazon Web Services calculates overlapping retrievals. For example, if you submit a retrieval job for a 6 GB archive in hour 1, then another in hour 2, and third in hour 3, you will have three retrieval jobs running at the same time, which could be interpreted to mean a retrieval rate of (3 * 6 GB) / 4 hours = 4.5 GB / hour. Or, Amazon Web Services might only consider the hour in which the jobs are submitted, and thus consider this a rate of (1 * 6 GB) / 4 hours = 1.5 GB / hour.

Amazon Web Services looks at the peak retrieval rate for the entire month, and then charges you as if you used that peak rate *for the entire month*. For example, if you submitted the 10 retrieval jobs for a 6 GB archive within an hour as described above (for a retrieval rate of 15 GB / hour) once in a month, and did *nothing else* for the rest of the month, you will be charged as if you were using 15 GB / hour for the *whole month*. A 30 day month has 720 hours, so this would be 15 * $0.01 * 720 = $108. Note that the storage cost for 60 GB / month in Amazon Glacier is $0.60, so a $108 retrieval cost is 180 times the amount being paid for storage.

But what about "You can retrieve up to 5% of your average monthly storage (pro-rated daily) for free each month"? Amazon Web Services' FAQ states "To calculate your free data we look at your daily allowance and divide it by the number of hours in the day that you retrieved data." For the purpose of this example, let's say that the 10 archives (6 GB each) represents everything you've stored in Amazon Glacier, so you have 60 GB stored. 5% of 60 GB is 3 GB. But note the "pro rated daily" wording - this means that while you can retrieve 3 GB per month for free, you must spread it evenly over all the days in the month. Assuming a 30 day month, you can only retrieve 3 GB / 30 = 0.1 GB / day without being charged.

In the example above, where we submitted 10 retrieval jobs for 6 GB archives within an hour, resulting in a peak retrieval rate of 15 GB / hour, we would be retrieving data for 4 hours. Thus our 0.1 GB of free quota for the day is further divided by 4 for a free rate of 0.025 GB / hour. We get to deduct this from our peak rate, giving us a billable rate of 14.75 GB / hour instead of 15 GB / hour, lowering the cost from $108 to $106.20 - not a particularly significant savings.

Note that this example is a worst case scenario for Amazon Glacier retrievals - where you need to retrieve everything you have in Amazon Glacier as quickly as possible. If the data you store in Amazon Glacier is for disaster recovery, this may be something you need to be ready for, because you might need all of it quickly.

Note that this section has focused exclusively on the GB / hour charge on the retrieval rate, since that is the most difficult to understand, and potentially the most costly part of the cost of migration to Cloud Storage Nearline. There are other costs associated with migrating, such as egress and per-request charges, but these are relatively easy to understand and will likely not dominate the overall cost, and are not discussed further here.

Recommendations

Since this document is intended to provide best practices for migrating from Amazon Glacier to Cloud Storage Nearline, we can safely assume that you will want to retrieve all or most of your data from Amazon Glacier in a relatively short period of time. With that in mind, here are some rules of thumb to follow when planning your migration in order to minimize your Amazon Glacier retrieval costs:

1. Retrieve your data from Amazon Glacier *evenly* over a minimum period of one *complete* month.

This is the single most important piece of advice. Since you're going to be billed as if you were using your peak retrieval rate within a calendar month for the whole month, you should strive to actually use your peak retrieval rate for the whole month. Also note that you must align your migration with the calendar month. If you start your migration on the 15th of March, and end it on the 15th of April, your migration will cost you roughly *double* what it would've cost had you started it on the 1st of April and ended on the 30th of April, even though both migrations take approximately 30 days.

Note that this is not at all trivial to do! Running commands manually as shown in the previous sections will not get it done for you, since you probably need to sleep and take days off. You will need some form of automation, which Amazon does not provide, thus you will likely need to *write* some automation.

2. Make sure you only retrieve your data once.

Remember that your job output is only available for a limited time after the job completes! If you don't get it copied somewhere durable before the limited time has elapsed, you'll have to retrieve it (and pay for that retrieval) a second time. Again some form of automation is indicated for this.

3. If you don't need some of your data *don't migrate it.*

Not retrieving the data is free. The cost of Amazon Glacier (and Cloud Storage Nearline) storage is so low that it is possible that the cost of storing a piece of data is less than the cost of investigating if it still needs to be stored. However, since the cost of retrieving it is *not* small, now might be a good time to decide what data is no longer needed.

4. Don't delete any data from Amazon Glacier (including any data you don't intend to migrate!) until *after* you have completed your migration.

This will stretch the 5% of free retrievals further.

5. Consider extending your migration over additional *complete* months.

If you migrate over two months, you'll be able to retrieve 10% of your data for free. Three months gives you 15%. Twenty months gives you 100% free, if you can wait that long. Once again, automation will be key.

Appendix A: Creating a Workspace and Staging Area

The instructions in this appendix assume you are signed in using a Google account, are using a Linux or Mac environment, and are comfortable running commands on a command-line interface.

Create or Select a Cloud Platform Project

If you have not already created the Google Cloud Platform project that you intend to migrate your data to, you should create one now by going to the Google Developers Console (https://console.developers.google.com) and clicking on "Create Project".

> Note: The Google Developer Console UI is constantly being improved, so depending on when you are reading this, you may find the UI to be slightly different than described.

If you have never used the Google Cloud Platform before, you may qualify for a Free Trial project. If this is the case, you will see a "Sign up for a free trial" link you can click on. This will give you $300 in free trial credit to test the platform over a period of 60 days.

> Note: The details of the Free Trial offer a subject to change, and may not be available at the time you are reading this document. Visit https://cloud.google.com for the most up to date offers and pricing.

Install and Configure the Cloud SDK

You should now install the Cloud SDK on your local workstation by following the instructions at https://cloud.google.com/sdk.

After running the **`gcloud auth login`** step, you should run:

```
local$ gcloud config set project PROJECT_ID
```

Where `PROJECT_ID` is the id of the project you plan to migrate your data to. This command configures **`gcloud`** and **`gsutil`** to use this project by default, so you don't need to specify it explicitly via the `--project_id` flag on each command.

> Note: Example commands following a `local$` prompt should be run on your local workstation. Example commands following a `cloud$` prompt should be run on a Google Compute Engine instance, configured as described below.

Create and Configure a Compute Engine Instance

The commands you will be running from this instance do not require a large amount of memory, but can take advantage of multiple cores and ample storage space. So we suggest an 8 core, high CPU instance, with a minimum of 2TB of Standard persistent disk to ensure you have the full amount of persistent disk throughput available. The additional throughput and IOPS provided by Solid-state persistent disk or Local SSD is not needed for this particular application, because the instance will most likely be network bound, not disk bound. If you think you will need to stage more than 2TB of content on the instance, then a larger Standard persistent disk can be used, up to the maximum of 10TB. These instructions will assume 2TB is sufficient for your needs.

> Note: These instructions use **`gcloud compute`** commands, but it is equally valid to create your instance via the Google Developers Console (https://console.developers.google.com)

You can create an 8 core high CPU instance like this:

```
local$ gcloud compute instances create nl-workspace \
    --machine-type n1-highcpu-8 \
    --scopes https://www.googleapis.com/auth/devstorage.read_write \
    --zone us-central1-c
```

Note that we specify the `--scope` to give the instance read/write access to Cloud Storage (i.e. devstorage.read_write). This will allow the instance to write to Cloud Storage Nearline.

These example commands specify the `us-central1-c` zone. You should choose a zone that is co-located with the Cloud Storage Nearline bucket you intend to migrate your data to. If you are using a continental bucket location (i.e. ASIA/EU/USA) then any zone in that continent will do. If you are using a regional bucket location (e.g. us-central1) then you should choose a zone within that region.

Next, we create a 2TB persistent disk:

```
local$ gcloud compute disks create nl-data \
    --size 2000 \
    --zone us-central1-c
```

And attach it to our instance:

```
local$ gcloud compute instances attach-disk nl-workspace \
    --disk nl-data \
    --zone us-central1-c
```

We can now ssh into our instance, mount the persistent disk, install and configure the required software:

```
local$ gcloud compute ssh nearline --zone us-central1-c
```

The following commands will mount the 2TB Persistent Disk at /data:

```
cloud$ sudo mkdir /data
cloud$ sudo /usr/share/google/safe_format_and_mount /dev/sdb /data
cloud$ sudo chmod 777 /data
```

The **`gsutil`** command comes preinstalled on the default Compute Engine instance, and is configured to use the project's service account. The `--scope` you specified when creating the Compute Engine instance ensures that gsutil will be able to read from and write to any bucket owned by that project. However, it is a good idea to make sure you have the latest Cloud SDK tools by running:

```
cloud$ sudo gcloud components update
```

Finally, you will need the **aws** command. This can be obtained with the following series of commands:

```
cloud$ sudo apt-get update
cloud$ sudo apt-get install python-pip
cloud$ sudo pip install awscli
```

Finally, you must configure the **aws** command with your credentials:

```
cloud$ aws configure
```

This will prompt you to enter your Amazon Web Services credentials. Ensure that the credentials you provide are for a user that has at least read only access to Amazon Glacier. This can be accomplished via the `AmazonGlacierReadOnlyAccess` policy. For more detail on how to configure Amazon Identity and Access Management, please consult the Amazon Web Services documentation.

You now have a Compute Engine instance that is configured appropriately for following the rest of the instructions in this document.

Note that if you are not actively using the Compute Engine instance for migrating data, you can reduce costs by stopping the instance as follows:

```
local$ gcloud compute instances stop nl-workspace --zone us-central1-c
```

This is akin to powering the machine down. You cannot do anything with the instance while it is in this state, but all the configuration you just performed is not lost, and you will not be billed for usage while the instance is stopped.

When you are ready to use the instance again, you can start it again like so:

```
local$ gcloud compute instances start nl-workspace --zone us-central1-c
```

At this point you will be able to ssh back into the machine and continue working, but just like with a power cycle of physical machine, you must make sure you have saved any work before stopping the machine if you want access to it after you restart it.

References

[1] http://docs.aws.amazon.com/cli/latest/reference/glacier/initiate-job.html
[2] http://docs.aws.amazon.com/cli/latest/reference/glacier/get-job-output.html
[3] http://aws.amazon.com/glacier/pricing/
[4] http://docs.aws.amazon.com/AmazonS3/latest/dev/restoring-objects-console.html
[5] http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketGET.html