Chapter 4. Big Data Storage

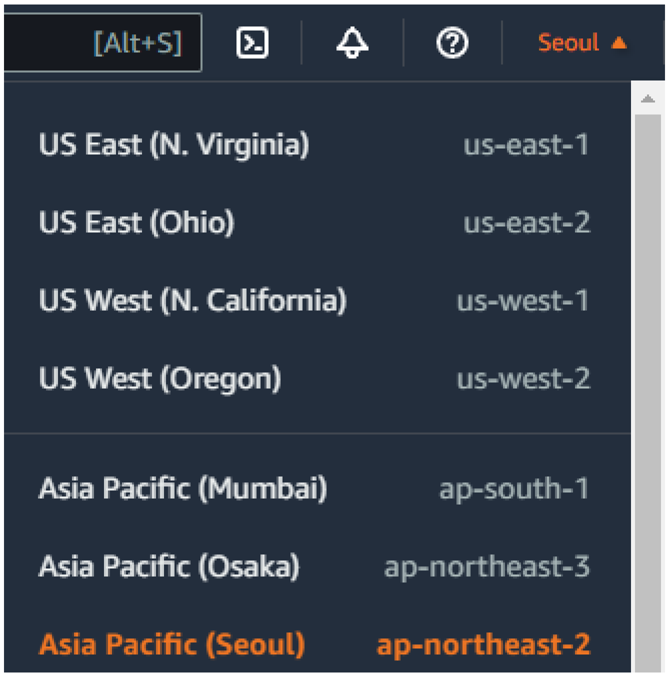
Exercise Workbook

Contents

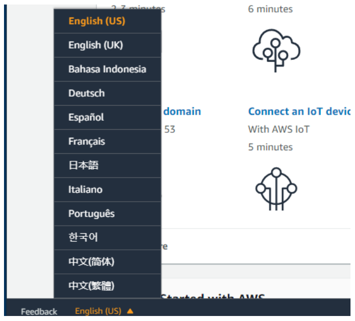
Lab 1: AWS S3 Storage

In this lab, you will create AWS Account and explore the S3 Service. Then you will access and manage S3 Storage using AWS CLI and Cyberduck.

1. Create an AWS Account
    - 1.1 Create a free tier AWS account from https://aws.amazon.com/console/
        - 1.1.1 Creating a free tier account will require a credit card. However, the services that we will be using will all be within free tier eligibility.
    - 1.2 Configure your free account
        - 1.2.1 Change the AWS region to a location nearest to you


[Alt+S]
Seoul
US East (N. Virginia) us-east-1
US East (Ohio) us-east-2
US West (N. California) us-west-1
US West (Oregon) us-west-2
Asia Pacific (Mumbai) ap-south-1
Asia Pacific (Osaka) ap-northeast-3
Asia Pacific (Seoul) ap-northeast-2

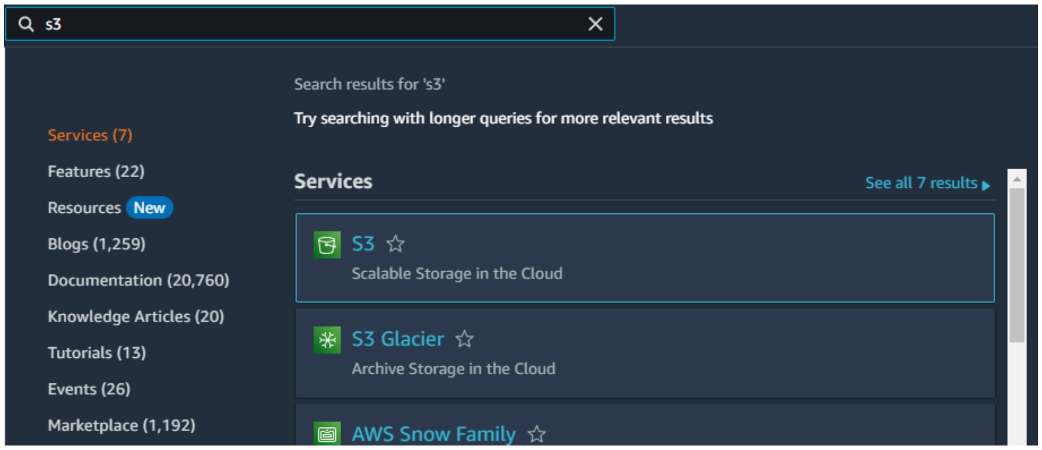
        - 1.2.2 Change the language to English or a language option of your choice. This option is on the bottom left of your browser.



2. Explore the AWS S3 Service

2.1 Open the Amazon S3 console at https://s3.console.aws.amazon.com/

2.1.1 You can get to any Amazon AWS service in several ways. Use the search bar on top and search for the service.



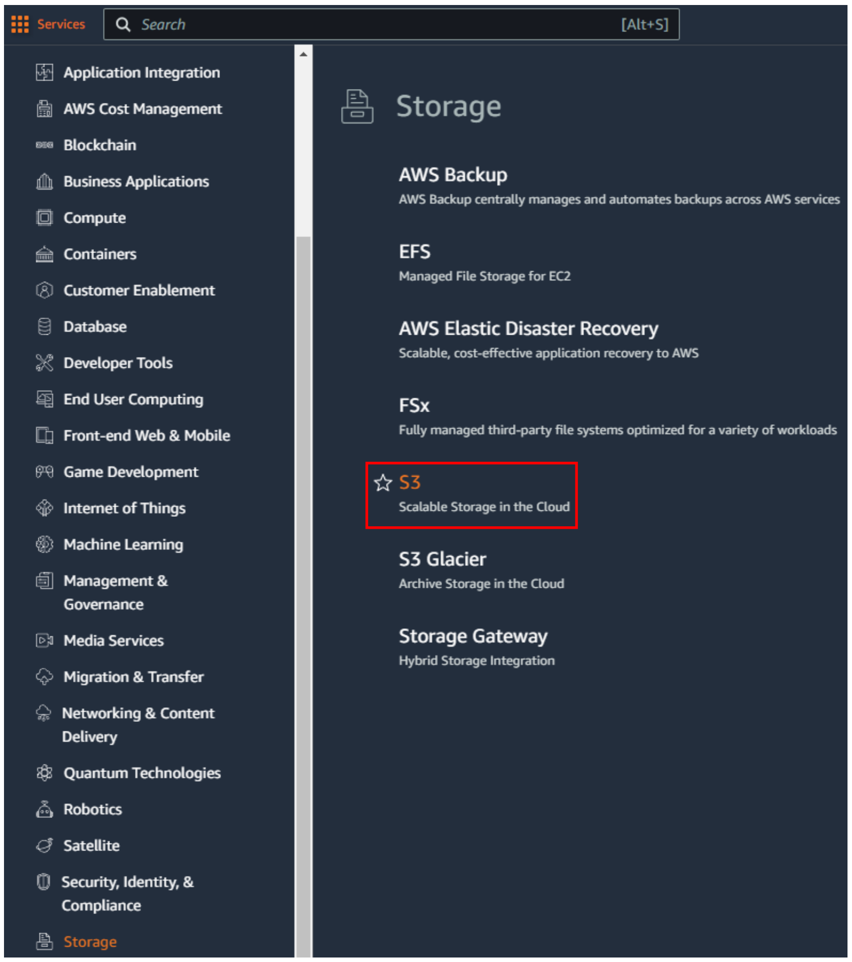
Search results for 's3'

Try searching with longer queries for more relevant results

S3 — Scalable Storage in the Cloud

S3 Glacier — Archive Storage in the Cloud

2.1.2 Click on the Service icon next to the AWS logo to bring up a menu of all services.


Services
Search
[Alt+S]
Application Integration
AWS Cost Management
Blockchain
Business Applications
Compute
Containers
Customer Enablement
Database
Developer Tools
End User Computing
Front-end Web & Mobile
Game Development
Internet of Things
Machine Learning
Management & Governance
Media Services
Migration & Transfer
Networking & Content Delivery
Quantum Technologies
Robotics
Satellite
Security, Identity, & Compliance
Storage
Storage
AWS Backup
AWS Backup centrally manages and automates backups across AWS services
EFS
Managed File Storage for EC2
AWS Elastic Disaster Recovery
Scalable, cost-effective application recovery to AWS
FSx
Fully managed third-party file systems optimized for a variety of workloads
S3
Scalable Storage in the Cloud
S3 Glacier
Archive Storage in the Cloud
Storage Gateway
Hybrid Storage Integration

2.1.3 Directly using link: https://s3.console.aws.amazon.com/

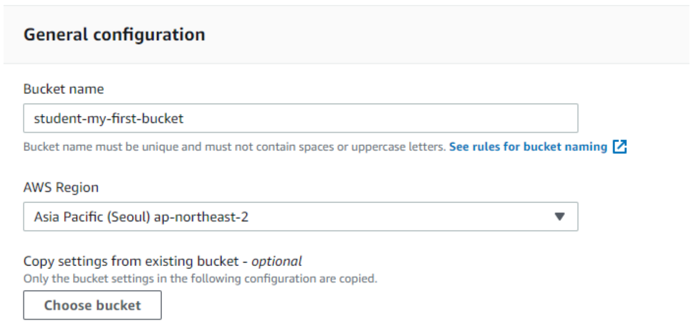
2.2 Create a new s3 bucket

2.2.1 Click on the create bucket icon.

2.2.2 Enter a unique DNS compliant name for your bucket.

- Be unique across all of Amazon S3
- Between 3 and 63 characters
- No uppercase
- Start with lowercase letter or number

2.2.3 Choose a region close to you.



General configuration

Bucket name

student-my-first-bucket

Bucket name must be unique and must not contain spaces or uppercase letters. See rules for bucket naming

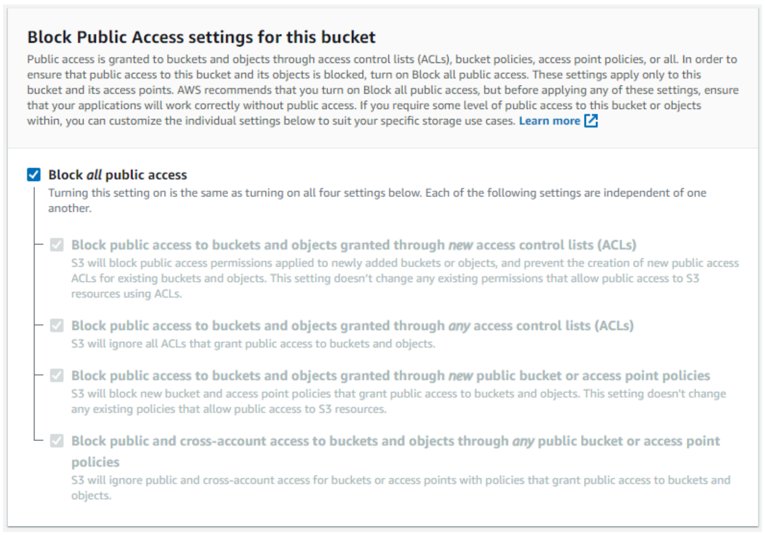
AWS Region

Asia Pacific (Seoul) ap-northeast-2

Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

2.2.4 Keep the checkmark on **Block *all* public access**. By default, Amazon S3 blocks public access to S3 buckets. You can modify this later, even after the bucket has been created.



Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more

✓ Block *all* public access

Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

✓ Block public access to buckets and objects granted through *new* access control lists (ACLs)

S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

✓ Block public access to buckets and objects granted through *any* access control lists (ACLs)

S3 will ignore all ACLs that grant public access to buckets and objects.

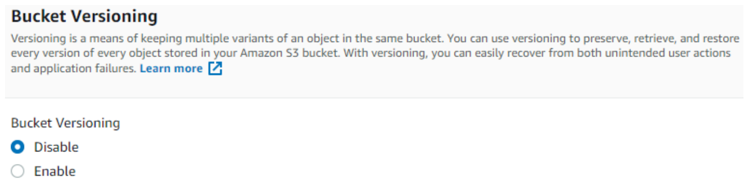
✓ Block public access to buckets and objects granted through *new* public bucket or access point policies

S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

✓ Block public and cross-account access to buckets and objects through *any* public bucket or access point policies

S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

2.2.5 Keep value for **Bucket Versioning** to Disable. If you enable this feature, S3 will keep multiple variants of an object in the bucket. This may be useful against accidental deletes.



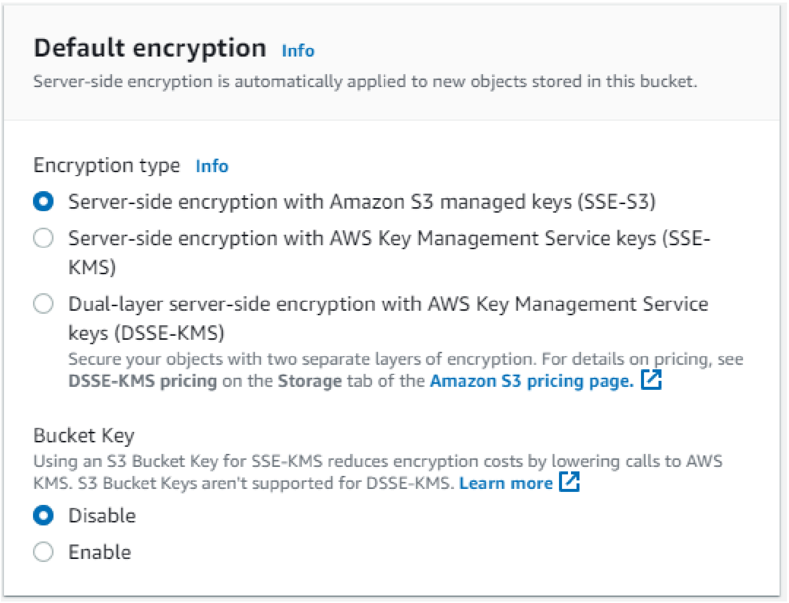
Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more

Bucket Versioning

[✓ Disable] Enable

2.2.6 Keep value for **Default encryption** to Disable. Users can choose to encrypt their data-at-rest by enabling this feature.



Default encryption Info

Server-side encryption is automatically applied to new objects stored in this bucket.

Encryption type Info

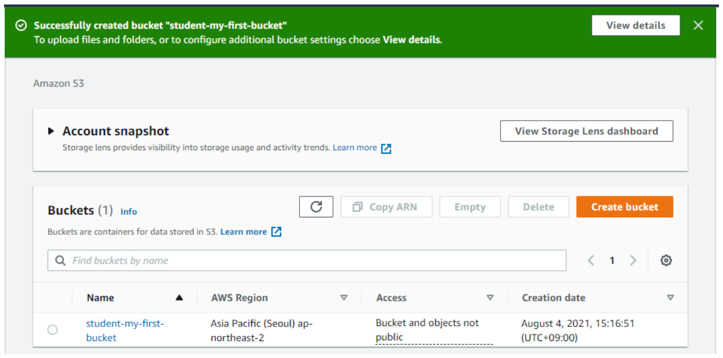
- [✓ Server-side encryption with Amazon S3 managed keys (SSE-S3)]
- Server-side encryption with AWS Key Management Service keys (SSE-KMS)
- Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)
  Secure your objects with two separate layers of encryption. For details on pricing, see **DSSE-KMS pricing** on the **Storage** tab of the **Amazon S3 pricing page.**

Bucket Key

Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for DSSE-KMS. **Learn more**

- [✓ Disable]
- Enable

2.2.7 Click the **Create bucket** button to create your bucket.



Successfully created bucket "student-my-first-bucket"
To upload files and folders, or to configure additional bucket settings choose **View details**.

Amazon S3

Account snapshot
Storage lens provides visibility into storage usage and activity trends. Learn more

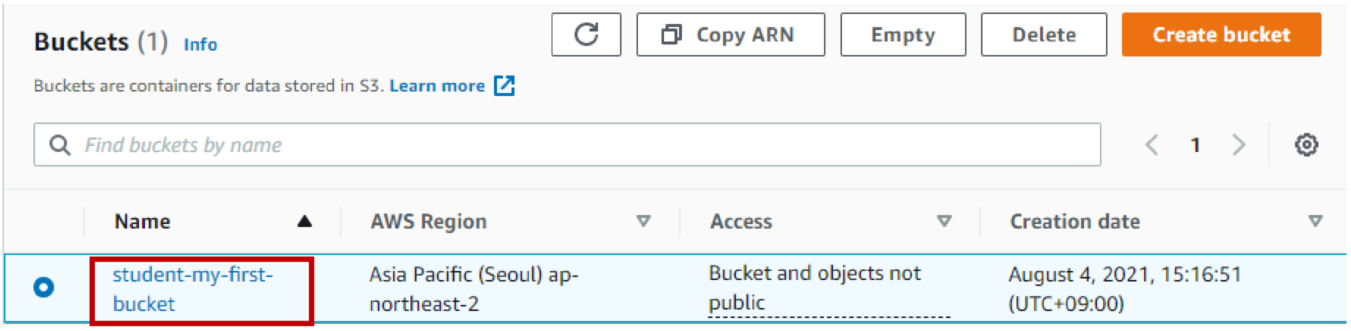
Buckets (1) Info

Buckets are containers for data stored in S3. **Learn more**

	Name	AWS Region	Access	Creation date
○	student-my-first-bucket	Asia Pacific (Seoul) ap-northeast-2	Bucket and objects not public	August 4, 2021, 15:16:51 (UTC+09:00)

2.3 Uploading objects to a bucket

2.3.1 Go to the Amazon S3 console and select the bucket to which you wish to upload object. Select from the list of buckets.



Buckets (1) Info

Buckets are containers for data stored in S3. **Learn more**

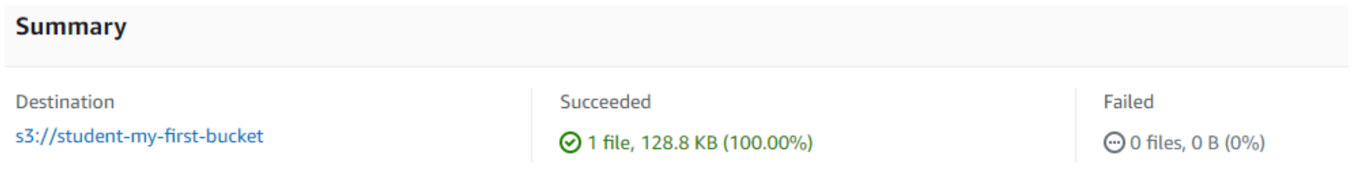
	Name	AWS Region	Access	Creation date
●	student-my-first-bucket	Asia Pacific (Seoul) ap-northeast-2	Bucket and objects not public	August 4, 2021, 15:16:51 (UTC+09:00)

2.3.2 From the **Objects** tab, select the **Upload** icon.

2.3.3 You can choose to upload files or folders. We will upload a file for now. Choose **Add Files** icon.

2.3.4 Select the seoul.jpeg file provided to you by your instructor.

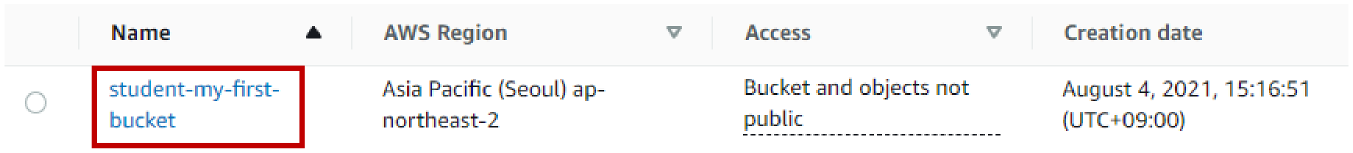
2.3.5 Click on the **Upload** button on the bottom of the screen to complete. When the upload is complete, you will get a success screen below.



Summary		
Destination	Succeeded	Failed
s3://student-my-first-bucket	✓ 1 file, 128.8 KB (100.00%)	0 files, 0 B (0%)

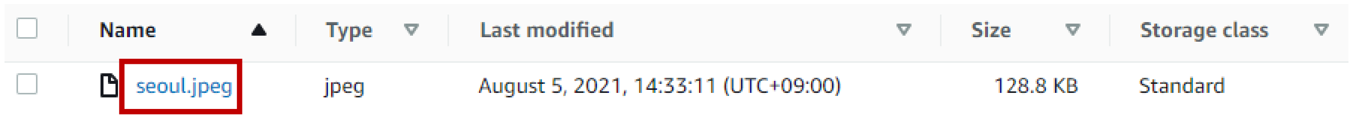
2.4 Downloading an object from an S3 bucket to a local directory

2.4.1 Go to the Amazon S3 console and select the bucket you wish to upload from the list of buckets



	Name	AWS Region	Access	Creation date
○	student-my-first-bucket	Asia Pacific (Seoul) ap-northeast-2	Bucket and objects not public	August 4, 2021, 15:16:51 (UTC+09:00)

2.4.2 From the **Objects** tab, select an object that you would like to download. For now, we will select the seoul.jpeg file that was uploaded.

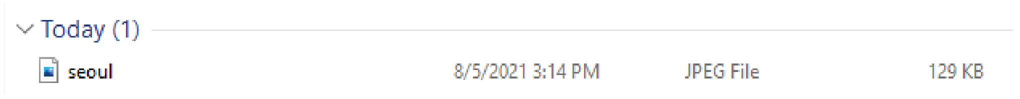


	Name	Type	Last modified	Size	Storage class
☐	seoul.jpeg	jpeg	August 5, 2021, 14:33:11 (UTC+09:00)	128.8 KB	Standard

2.4.3 From the object’s **Properties** tab, review the information for this object

2.4.4 Click on the **Download** icon on the top to download this object. The object will be downloaded to your download directory on your local system.

2.4.5 Confirm that the file has been downloaded



Today (1)

seoul	8/5/2021 3:14 PM	JPEG File	129 KB

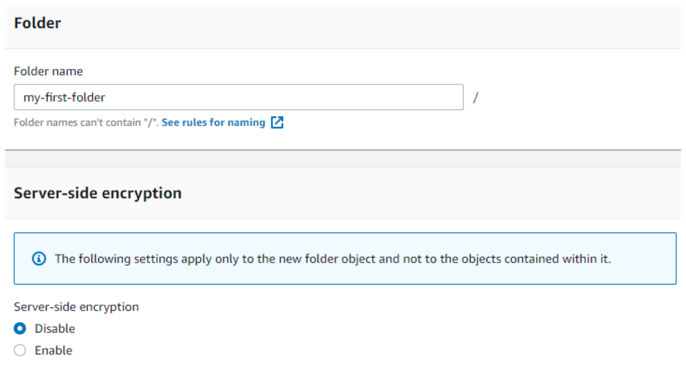
2.5 Create folders in an S3 bucket

2.5.1 Go to the Amazon S3 console and select the bucket you wish to upload from the list of buckets

2.5.2 Choose the **Create Folder** button to create a new folder

2.5.3 Enter a name for your folder

2.5.4 Keep value the default values for the rest



Folder

Folder name

my-first-folder /

Folder names can't contain "/". See rules for naming

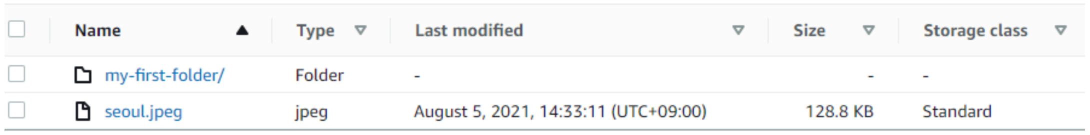
Server-side encryption

The following settings apply only to the new folder object and not to the objects contained within it.

Server-side encryption

[✓ Disable] Enable

2.5.5 Click on **Create Folder** button on the buttom to create your new folder. You should now have a new folder and the seoul.jpeg file you uploaded earlier inside your bucket.

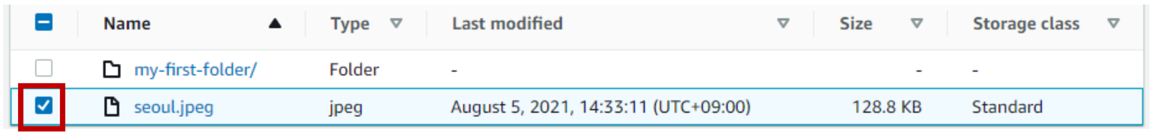


Name	Type	Last modified	Size	Storage class
my-first-folder/	Folder	-	-	-
seoul.jpeg	jpeg	August 5, 2021, 14:33:11 (UTC+09:00)	128.8 KB	Standard

2.6 Copy objects in a S3 bucket

2.6.1 Go to the S3 bucket or folder that contains the object you want to copy. For us, this will be in bucket that we first created and where we uploaded the seoul.jpeg file

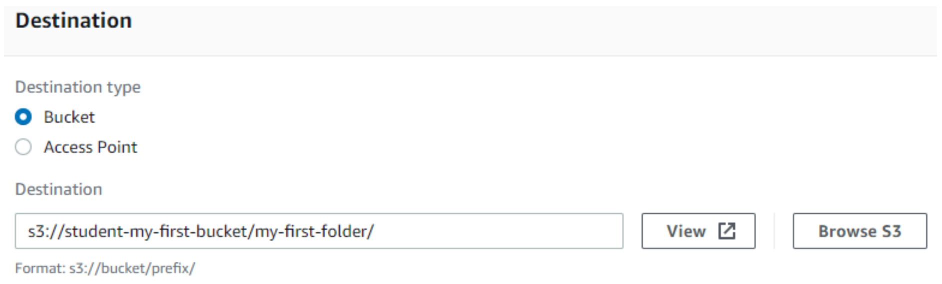
2.6.2 Select the object that you want to perform an action on.



Name	Type	Last modified	Size	Storage class
my-first-folder/	Folder	-	-	-
✓ seoul.jpeg	jpeg	August 5, 2021, 14:33:11 (UTC+09:00)	128.8 KB	Standard

2.6.3 From the **Actions** menu, choose **Copy**

2.6.4 We now have to provide a destination to copy. We can do this in two ways. One is to give the bucket or bucket/folder address. The other is to use the **Browse S3** option to open a GUI window and navigate to the desired bucket or folder. If you choose the GUI method, the Destination address will be automatically filled in for you.



Destination

Destination type

[✓ Bucket] Access Point

Destination

s3://student-my-first-bucket/my-first-folder/

Format: s3://bucket/prefix/

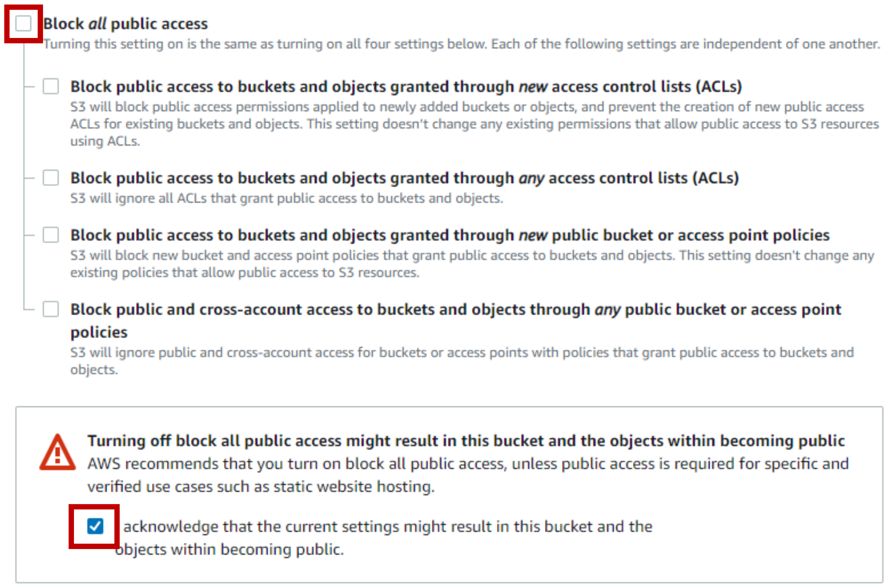
2.6.5 As we copy the object, it is possible to change some of the parameter associated with the object. Leave everything the same and click on the **Copy** button on the bottom of the screen.

2.7 Accessing S3 objects externally

In the bucket that we have already created, recall that we had set the public access to be disabled.

2.7.1 Return to the top level of your s3 account

2.7.2 Create a new bucket but this time, allow public access to it by unchecking the **Block *all* public access**. You must acknowledge that this might expose your object to the public. Make sure that you provide a globally unique name for the bucket.



☐ **Block *all* public access**
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

☐ **Block public access to buckets and objects granted through *new* access control lists (ACLs)**
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

☐ **Block public access to buckets and objects granted through *any* access control lists (ACLs)**
S3 will ignore all ACLs that grant public access to buckets and objects.

☐ **Block public access to buckets and objects granted through *new* public bucket or access point policies**
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

☐ **Block public and cross-account access to buckets and objects through *any* public bucket or access point policies**
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

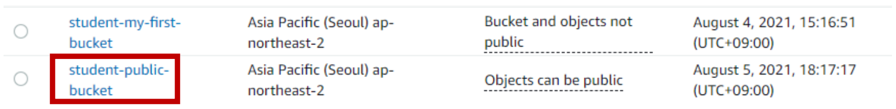
Turning off block all public access might result in this bucket and the objects within becoming public
AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

☑ I acknowledge that the current settings might result in this bucket and the objects within becoming public.

2.7.3 Leave everything as is and click on **Create bucket** to create the bucket.

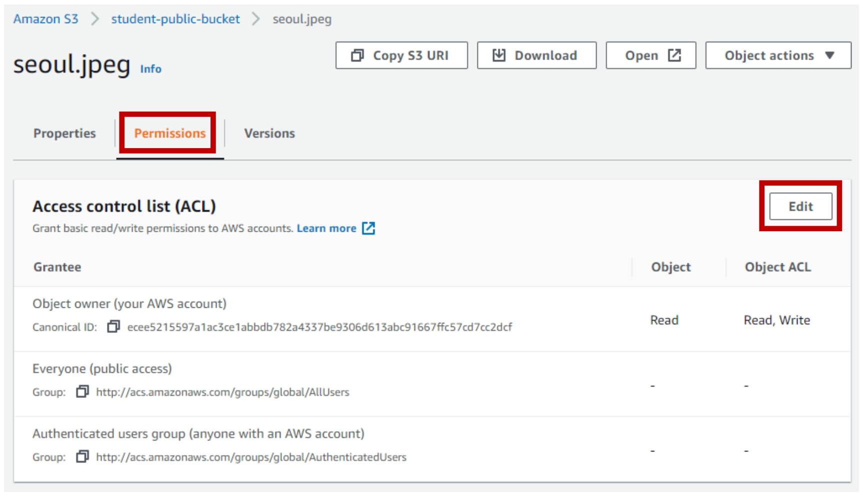
2.7.4 Copy seoul.jpeg file that you uploaded in step 2.3 to the newly created bucket from above.

2.7.5 Go to where you copied the file from above.



○	student-my-first-bucket	Asia Pacific (Seoul) ap-northeast-2	Bucket and objects not public	August 4, 2021, 15:16:51 (UTC+09:00)
○	student-public-bucket	Asia Pacific (Seoul) ap-northeast-2	Objects can be public	August 5, 2021, 18:17:17 (UTC+09:00)

2.7.6 The bucket you copied to has allowed public access, however, you need to set an object's access rights specifically. It does not have public access as of yet. Click on object and go to the **Permissions** tab.



Amazon S3 > student-public-bucket > seoul.jpeg

seoul.jpeg Info

Copy S3 URI | Download | Open | Object actions

Properties | Permissions | Versions

Access control list (ACL)

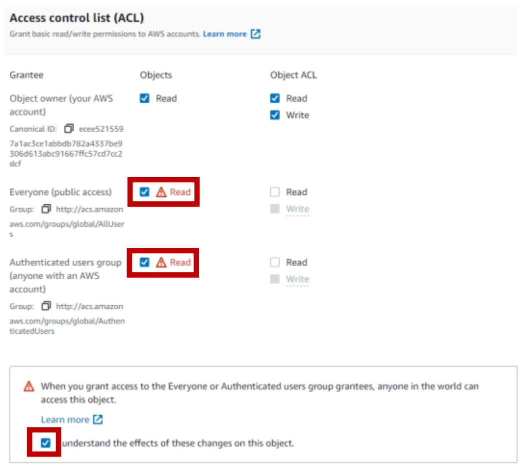
Grant basic read/write permissions to AWS accounts. Learn more

Edit

Grantee	Object	Object ACL
Object owner (your AWS account) Canonical ID: ecee5215597a1ac3ce1abbdb782a4337be9306d613abc91667ffc57cd7cc2dcf	Read	Read, Write
Everyone (public access) Group: http://acs.amazonaws.com/groups/global/AllUsers	-	-
Authenticated users group (anyone with an AWS account) Group: http://acs.amazonaws.com/groups/global/AuthenticatedUsers	-	-

2.7.7 Click on **Edit** button to modify the access control list

2.7.8 Allow read access to all persons (public access). You must also acknowledge that you understand the consequences of this action.



Access control list (ACL)

Grant basic read/write permissions to AWS accounts. Learn more

Grantee	Objects	Object ACL
Object owner (your AWS account) Canonical ID: ecee5215597a1ac3ce1abbdb782a4337be9306d613abc91667ffc57cd7cc2dcf	✓ Read	✓ Read ✓ Write
Everyone (public access) Group: http://acs.amazonaws.com/groups/global/AllUsers	✓ Read	☐ Read ☐ Write
Authenticated users group (anyone with an AWS account) Group: http://acs.amazonaws.com/groups/global/AuthenticatedUsers	✓ Read	☐ Read ☐ Write

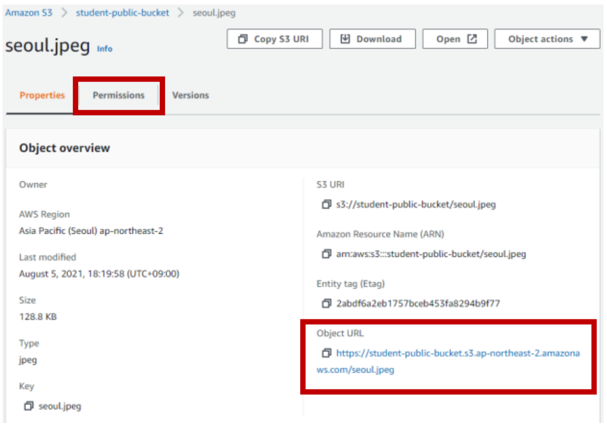
When you grant access to the Everyone or Authenticated users group grantees, anyone in the world can access this object.

Learn more

✓ I understand the effects of these changes on this object.

2.7.9 Save the changes using the **Save changes** button.

2.7.10 From the Properties tab of the object, find the **Object URL** and click to test public access.



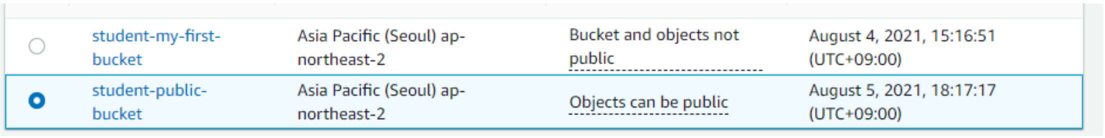
Object overview

Owner	
AWS Region	Asia Pacific (Seoul) ap-northeast-2
Last modified	August 5, 2021, 18:19:58 (UTC+09:00)
Size	128.8 KB
Type	jpeg
Key	seoul.jpeg
S3 URI	s3://student-public-bucket/seoul.jpeg
Amazon Resource Name (ARN)	arn:aws:s3:::student-public-bucket/seoul.jpeg
Entity tag (Etag)	2abdf6a2eb1757bceb453fa8294b9f77
Object URL	https://student-public-bucket.s3.ap-northeast-2.amazonaws.com/seoul.jpeg

2.8 Delete an object in your S3 bucket

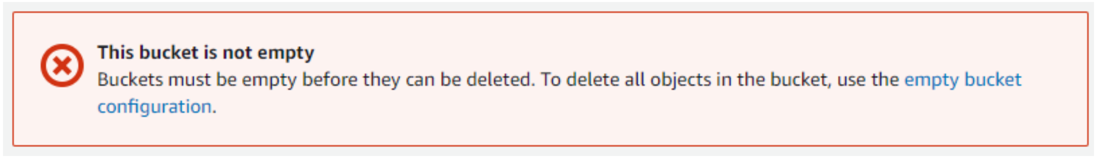
You incur charges while you have objects stored in S3. We will delete all the buckets and all of its contents.

2.8.1 Go to your bucket list and select the bucket you would like to delete



○	student-my-first-bucket	Asia Pacific (Seoul) ap-northeast-2	Bucket and objects not public	August 4, 2021, 15:16:51 (UTC+09:00)
●	student-public-bucket	Asia Pacific (Seoul) ap-northeast-2	Objects can be public	August 5, 2021, 18:17:17 (UTC+09:00)

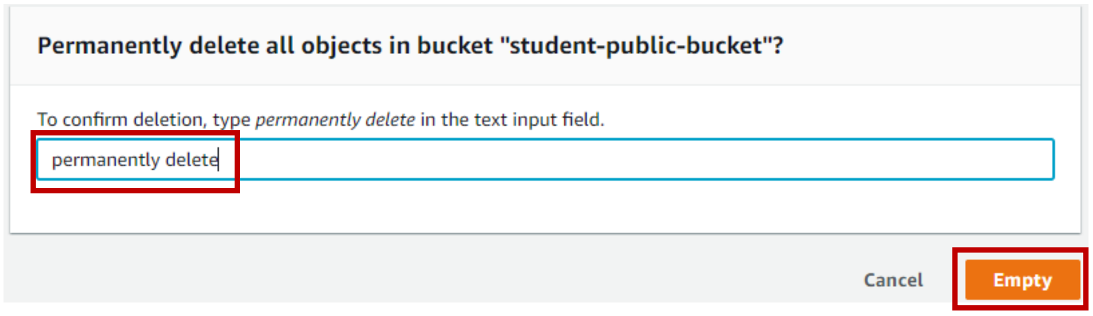
2.8.2 You cannot delete a bucket unless it is empty.



This bucket is not empty
Buckets must be empty before they can be deleted. To delete all objects in the bucket, use the empty bucket configuration.

2.8.3 Click on **Empty** button on the top menu to delete all objects and folders from inside the bucket.

2.8.4 As a caution, you will have to type `permanently delete` in order to empty the bucket. Finalize the operation by clicking on **Empty**.

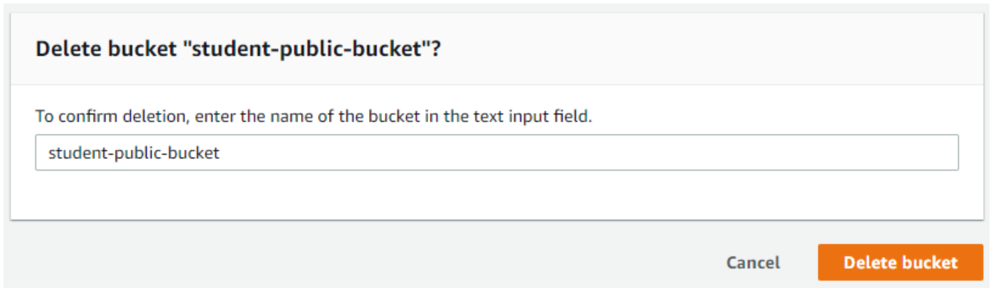


Permanently delete all objects in bucket "student-public-bucket"?

To confirm deletion, type *permanently delete* in the text input field.

permanently delete

2.8.5 Now that the bucket is empty, select it again from your bucket list and select **Delete** from the top buttons. As a caution, you will have to enter the name of the bucket in order to delete it. You will not get the icon to delete the bucket until you correctly spell the name of the bucket.


Delete bucket "student-public-bucket"?

To confirm deletion, enter the name of the bucket in the text input field.

student-public-bucket

2.8.6 In the same manner, delete the remaining bucket as well. When completed, you should not have any buckets remaining in the s3 account.

2.9 Working with lifecycle settings for S3 buckets

AWS S3 have several storage classes with various performance and cost specification. In simple terms, users can select different storage classes depending on whether the data is hot, warm or cold.

You can use lifecycle rules to define what you want Amazon S3 to do during an object's lifecycle, such as moving an object to a different storage class, retaining the object, or deleting the object after a specified period of time has elapsed.

Before setting up your lifecycle configuration, keep the following in mind:

<u>Propagation delay</u>

When you add an S3 lifecycle configuration to a bucket, there is usually some latency until the new or updated lifecycle configuration is fully propagated to all Amazon S3 systems. You will need to wait a few minutes for the configuration to fully take effect. This delay can also occur when deleting the S3 lifecycle configuration.

<u>Disable or Delete Lifecycle Rules</u>

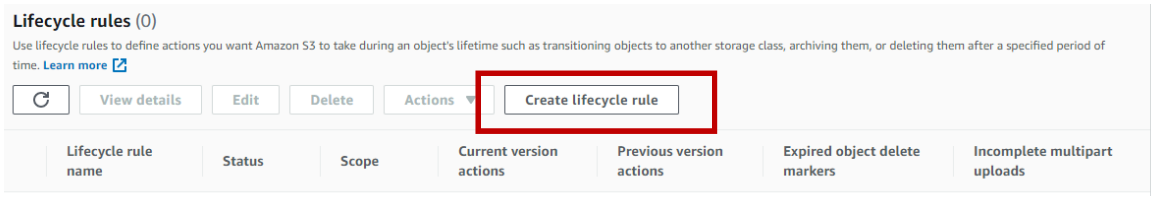
If you disable or delete a lifecycle rule, there is a small delay before Amazon S3 stops scheduling deletion or transition for new objects. Any objects that have already been reserved will be canceled and will not be deleted or converted.

<u>Existing and new objects</u>

When you add a lifecycle configuration to a bucket, the configuration rules apply to existing objects as well as objects you add later. For example, if you add a lifecycle configuration rule today that includes an expiration action that expires 30 days after creation of objects with a specific prefix, Amazon S3 queues all existing objects older than 30 days for deletion.

2.9.1 Make a new bucket and name it <some prefix>-lifecycle-archive. You have to add some prefix to the beginning of the name to assure that you can create a globally uniquely named bucket. You may choose to make it public or closed.

2.9.2 Select the bucket and open the **Management** tab. From there select **Create lifecycle rule**.


Lifecycle rules (0)

Use lifecycle rules to define actions you want Amazon S3 to take during an object's lifetime such as transitioning objects to another storage class, archiving them, or deleting them after a specified period of time. Learn more

Lifecycle rule name	Status	Scope	Current version actions	Previous version actions	Expired object delete markers	Incomplete multipart uploads

2.9.3 In Lifecycle rule name, enter a name for the rule. However, make sure it has a unique name within the bucket.

2.9.4 Select the scope of the lifecycle rule.

To apply this lifecycle rule to all objects with a specific prefix or tag, choose Limit the scope of this rule using one or more filters.

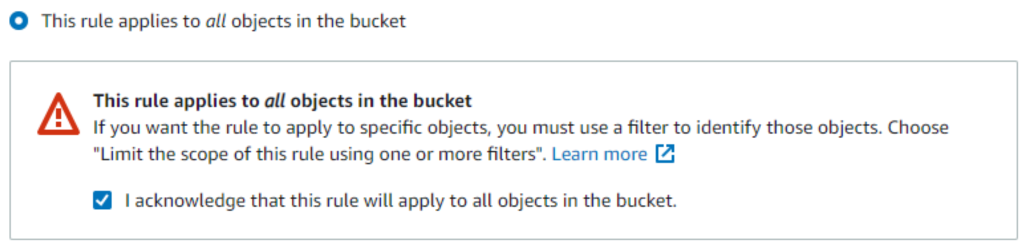
<u>From the Filter Type</u>:

To limit the range by prefix, enter a prefix in **Prefix**.

Objects in s3 can be given a tag. You enter a key and value pair to create the tag. To limit the scope to tags, choose **Add tag** and enter the tag key and value to which this rule should apply

To apply this lifecycle rule to all objects in the bucket, select **This rule applies to all objects in the bucket** and confirm that this rule applies to all objects in the bucket (I acknowledge that this rule applies to all objects in the bucket)

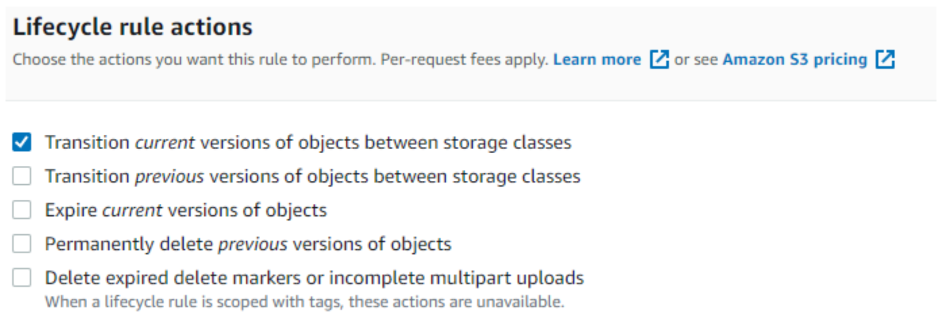
2.9.5 Apply this rule to all objects in the bucket.


● This rule applies to *all* objects in the bucket

This rule applies to *all* objects in the bucket
If you want the rule to apply to specific objects, you must use a filter to identify those objects. Choose "Limit the scope of this rule using one or more filters". Learn more

✓ I acknowledge that this rule will apply to all objects in the bucket.

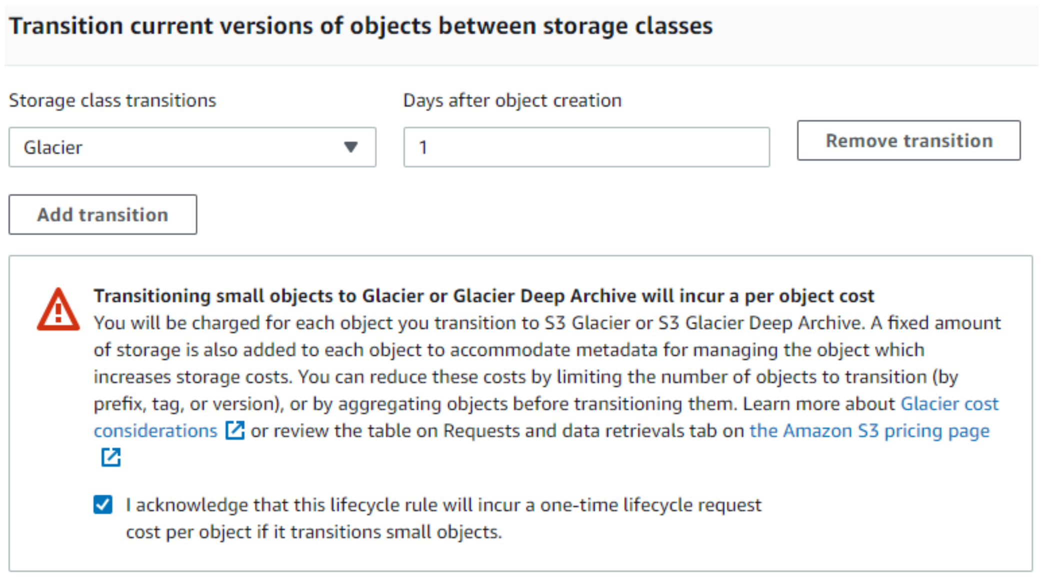
2.9.6 Set a Lifecycle rule actions. You can select more than one rule. Some of the options only applies if versioning is enabled and there are previous versions. In this lab, we will set rules for current versions since we are not versioning and there are no previous versions. **Select the Transition current versions of objects between storage classes**.


Lifecycle rule actions
Choose the actions you want this rule to perform. Per-request fees apply. Learn more or see Amazon S3 pricing

- [✓] Transition *current* versions of objects between storage classes
- [ ] Transition *previous* versions of objects between storage classes
- [ ] Expire *current* versions of objects
- [ ] Permanently delete *previous* versions of objects
- [ ] Delete expired delete markers or incomplete multipart uploads
  When a lifecycle rule is scoped with tags, these actions are unavailable.

2.9.7 When the above option is selected, a new section will appear for you to enter the transition policy. Here you enter the Days after object creation and the Storage class transitions. To view the cost and storage classes, refer to https://docs.aws.amazon.com/AmazonS3/latest/userguide/lifecycle-transition-general-considerations.html#glacier-pricing-considerations.

2.9.8 Choose Glacier storage class and set to 1 day after object creation. You must also acknowledge your understanding of this rule.


Transition current versions of objects between storage classes

Storage class transitions: Glacier

Days after object creation: 1

Transitioning small objects to Glacier or Glacier Deep Archive will incur a per object cost
You will be charged for each object you transition to S3 Glacier or S3 Glacier Deep Archive. A fixed amount of storage is also added to each object to accommodate metadata for managing the object which increases storage costs. You can reduce these costs by limiting the number of objects to transition (by prefix, tag, or version), or by aggregating objects before transitioning them. Learn more about Glacier cost considerations or review the table on Requests and data retrievals tab on the Amazon S3 pricing page

☑ I acknowledge that this lifecycle rule will incur a one-time lifecycle request cost per object if it transitions small objects.

2.9.9 Click the **Create rule** button on the bottom.

2.9.10 Upload the seoul.jpeg file to <prefix>-lifecycle-archive bucket. After more than a day has passed come back to check that the file has been moved to Glacier storage.

3. Using AWS CLI to access and manage S3 storage

3.1 Add Git Bash (**Windows environment only)**

Git Bash is a nice Windows terminal program that provides many of the basic functionalities of a Linux terminal. Many of the command we will execute below can be easily accomplished from Git Bash without installing further programs. To install Git Bash, go to https://git-scm.com/downloads and download the Windows 64 bit version. If you Windows environment is 32 bits, please upgrade your operating system.

You will use Git Bash as the terminal program for the rest of the lab.

3.2 Add AWS CLI for **Windows environment**

3.2.1 Install AWS CLI version 2 on Windows from https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-windows.html

3.2.2 Follow the instructions. The recommended method is to install the latest version from https://awscli.amazonaws.com/AWSCLIV2.msi

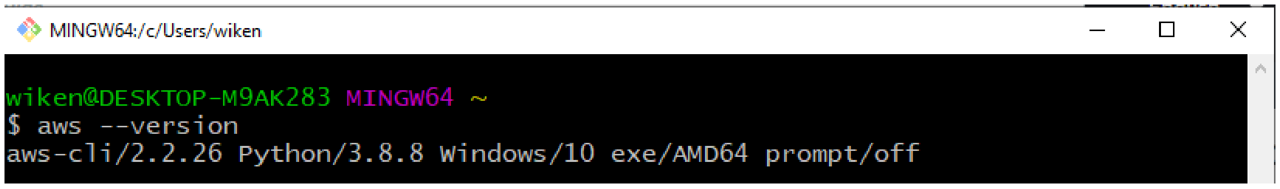
3.3 Add AWS CLI for **Max OS environment**

3.3.1 Install AWS CLI version 2 for Mac OS from https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-mac.html

3.3.2 Follow the instructions. The recommended method is to install the latest version from https://awscli.amazonaws.com/AWSCLIV2.pkg

3.4 Verify that AWS CLI was properly installed by opening up a terminal (Git Bash for Windows and a terminal in Mac OS. Both environments will give same results. The lab instructions will limit the screenshot to Git Bash for brevity) and executing the following:

```
aws --version
```

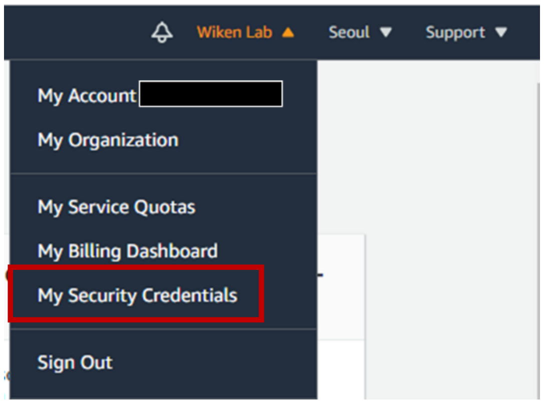


```
wiken@DESKTOP-M9AK283 MINGW64 ~
$ aws --version
aws-cli/2.2.26 Python/3.8.8 Windows/10 exe/AMD64 prompt/off
```

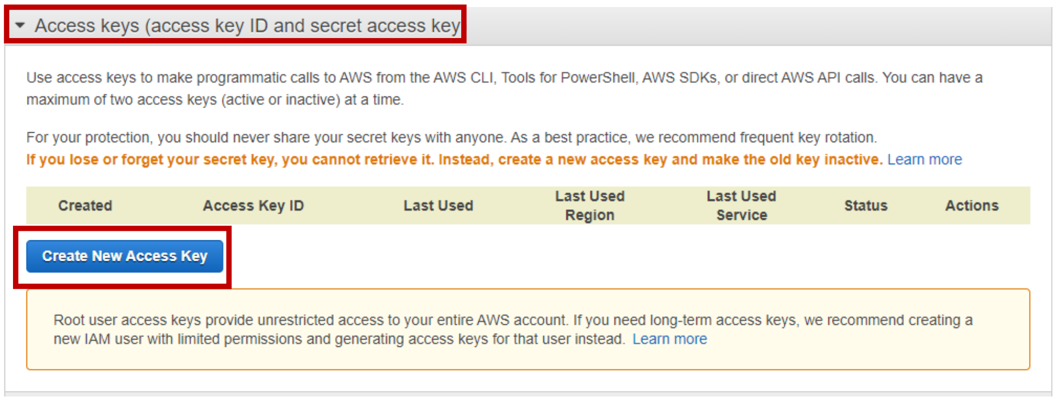
3.5 Set up your AWS CLI configuration

In order to execute commands from the AWS CLI, it will need to be configured with the proper authorizations to access your AWS account.

3.5.1 From the AWS console, click on the account name and select **My Security Credentials** from the drop-down menu



3.5.2 From the new page, select **Access Keys** and click on the **Create New Access Key button**



Access keys (access key ID and secret access key)

Use access keys to make programmatic calls to AWS from the AWS CLI, Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time.

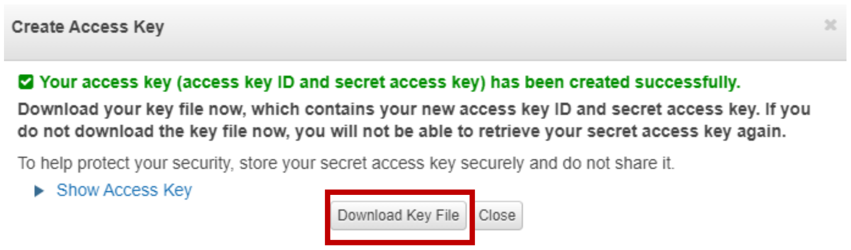
For your protection, you should never share your secret keys with anyone. As a best practice, we recommend frequent key rotation.
If you lose or forget your secret key, you cannot retrieve it. Instead, create a new access key and make the old key inactive. Learn more

Created	Access Key ID	Last Used	Last Used Region	Last Used Service	Status	Actions

Create New Access Key

Root user access keys provide unrestricted access to your entire AWS account. If you need long-term access keys, we recommend creating a new IAM user with limited permissions and generating access keys for that user instead. Learn more

3.5.3 From the pop-up window, select Download Key File



Create Access Key

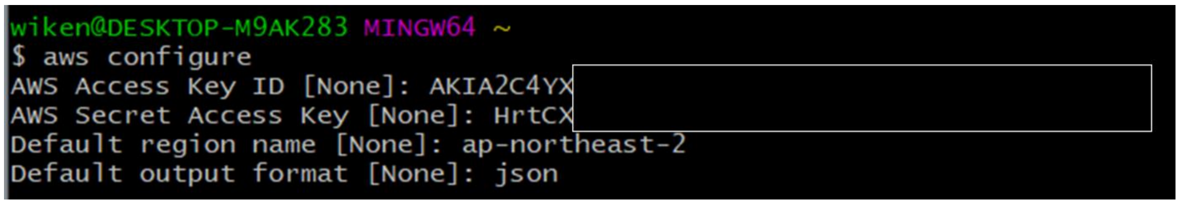
Your access key (access key ID and secret access key) has been created successfully.
Download your key file now, which contains your new access key ID and secret access key. If you do not download the key file now, you will not be able to retrieve your secret access key again.

To help protect your security, store your secret access key securely and do not share it.
▶ Show Access Key

Download Key File Close

3.5.4 Navigate to the download directory and open the downloaded access key file. Depending on your environment, it might open as an Excel document or as a Text document. In either case, there will be an AWS Access Key ID and AWS Secret Access Key. You will need these two pieces of information in the next step.

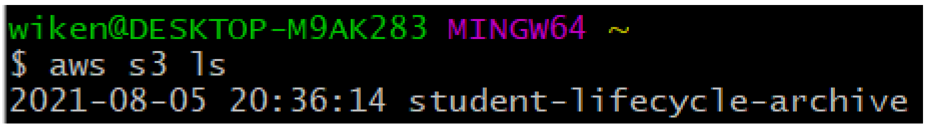
3.5.5 From Git Bash (terminal), run aws configure command and enter the AWS Access Key ID and AWS Secret Access Key when prompted. For the default region, enter the region name where you created your buckets in step 2.2. Finally, enter json for the output format.



```
wiken@DESKTOP-M9AK283 MINGW64 ~
$ aws configure
AWS Access Key ID [None]: AKIA2C4YX[redacted]
AWS Secret Access Key [None]: HrtCX
Default region name [None]: ap-northeast-2
Default output format [None]: json
```

3.5.6 Test the connection by issuing the following command to get a listing of your s3 buckets.

```
aws s3 ls
```



```
wiken@DESKTOP-M9AK283 MINGW64 ~
$ aws s3 ls
2021-08-05 20:36:14 student-lifecycle-archive
```

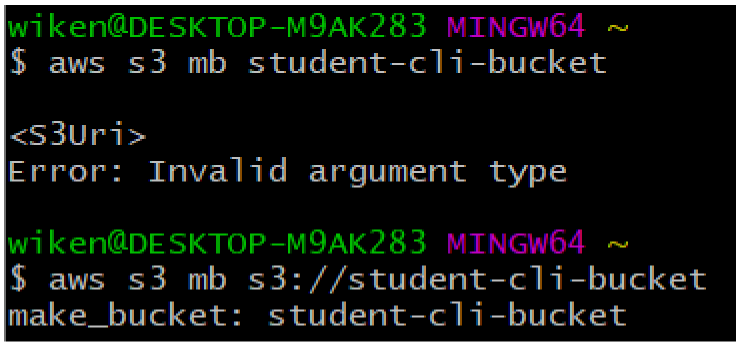
3.6 Create a new bucket with AWS CLI

3.6.1 First, get a list of available command with:

```
aws s3 help
```

3.6.2 Create a new bucket and name it <prefix>-cli-bucket. In creating a bucket, you must use an S3 URI path notation

```
aws s3 mb s3://student-cli-bucket
```



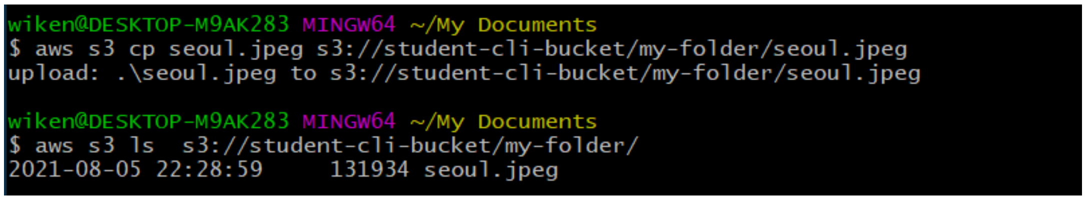
```
wiken@DESKTOP-M9AK283 MINGW64 ~
$ aws s3 mb student-cli-bucket

<S3Uri>
Error: Invalid argument type

wiken@DESKTOP-M9AK283 MINGW64 ~
$ aws s3 mb s3://student-cli-bucket
make_bucket: student-cli-bucket
```

3.6.3 List your buckets to confirm a new bucket has been created.

3.6.4 Now, copy the seoul.jpeg file to the newly created bucket with the following command. Notice that as we copy, we are also creating a folder named my-folder.

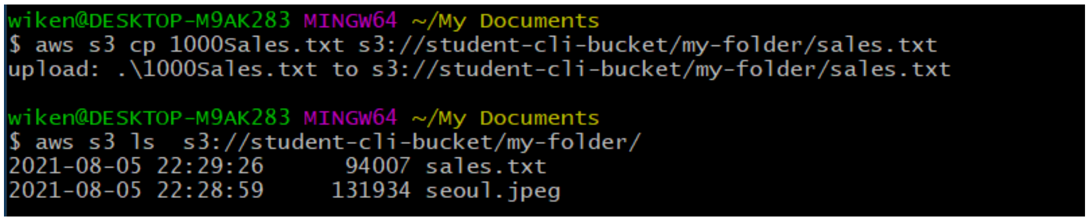
```
aws s3 cp seoul.jpeg s3://student-cli-bucket/my-folder/seoul.jpeg
```

3.6.5 Confirm that the file was uploaded



```
wiken@DESKTOP-M9AK283 MINGW64 ~/My Documents
$ aws s3 cp seoul.jpeg s3://student-cli-bucket/my-folder/seoul.jpeg
upload: .\seoul.jpeg to s3://student-cli-bucket/my-folder/seoul.jpeg

wiken@DESKTOP-M9AK283 MINGW64 ~/My Documents
$ aws s3 ls  s3://student-cli-bucket/my-folder/
2021-08-05 22:28:59     131934 seoul.jpeg
```

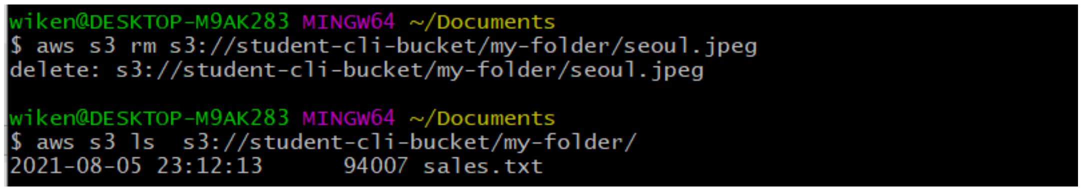
3.6.6 Now add another file of **your choice** to the my-folder directory in the <prefix>-cli-bucket bucket. Confirm that you have added the file by listing the bucket and directory again. Notice that we can change the name of the file at the destination.



```
wiken@DESKTOP-M9AK283 MINGW64 ~/My Documents
$ aws s3 cp 1000Sales.txt s3://student-cli-bucket/my-folder/sales.txt
upload: .\1000Sales.txt to s3://student-cli-bucket/my-folder/sales.txt

wiken@DESKTOP-M9AK283 MINGW64 ~/My Documents
$ aws s3 ls  s3://student-cli-bucket/my-folder/
2021-08-05 22:29:26      94007 sales.txt
2021-08-05 22:28:59     131934 seoul.jpeg
```

3.6.7 We have already uploaded the seoul.jpeg file previously, so let's delete it from the new bucket and folder. Execute the following command:

```
aws s3 rm s3://student-cli-bucket/my-folder/seoul.jpeg

aws s3 ls  s3://student-cli-bucket/my-folder/
```



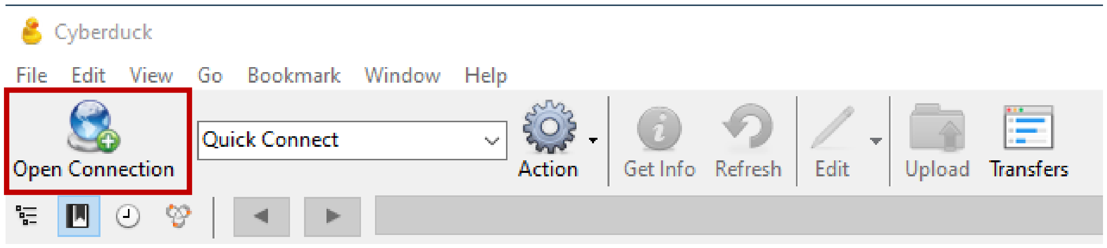
```
wiken@DESKTOP-M9AK283 MINGW64 ~/Documents
$ aws s3 rm s3://student-cli-bucket/my-folder/seoul.jpeg
delete: s3://student-cli-bucket/my-folder/seoul.jpeg

wiken@DESKTOP-M9AK283 MINGW64 ~/Documents
$ aws s3 ls  s3://student-cli-bucket/my-folder/
2021-08-05 23:12:13      94007 sales.txt
```

4. Using Cyberduck to explore AWS S3

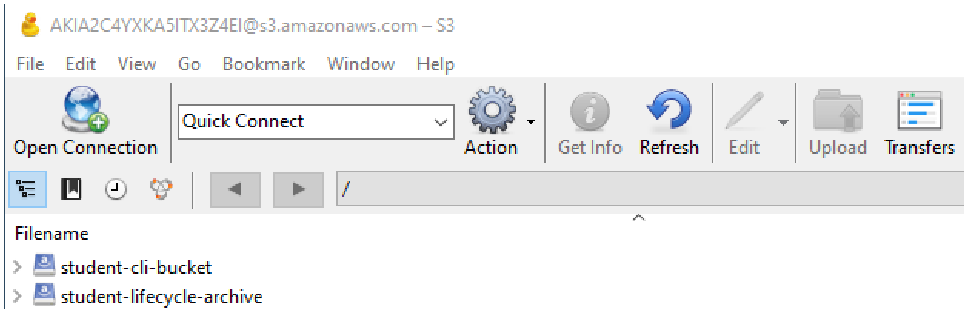
Cyberduck is a popular graphical user interface based FTP / file explorer. It can connect to various services including AWS s3.

4.1 Install cyberduck for your platform from https://cyberduck.io/download/

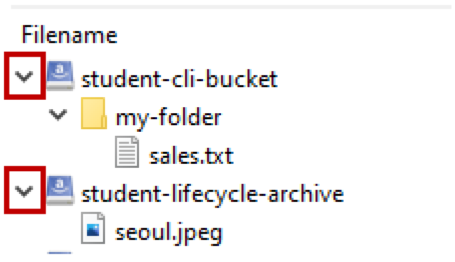
4.2 Start cyberduck and select Open Connection

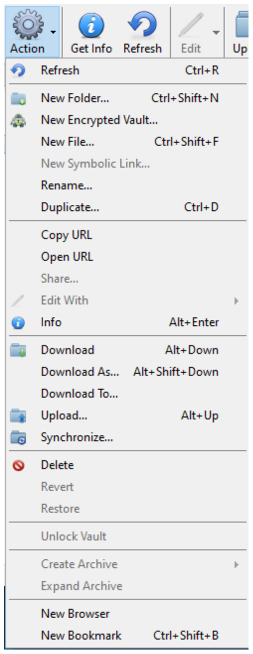


4.3 Select Amazon S3 from the drop down menu and enter your Access Key ID and Secret Access Key, if necessary, that was downloaded from step **3.5.1**. Cyberduck is smart and will read the AWS CLI setting for the Access Key ID and Secret Access Key. If it was unable to read this information from your system, you will have to enter it manually. Connect to your account.


AKIA2C4YXKA5ITX3Z4EI@s3.amazonaws.com – S3
File Edit View Go Bookmark Window Help
Open Connection
Quick Connect
Action
Get Info
Refresh
Edit
Upload
Transfers
/
Filename
student-cli-bucket
student-lifecycle-archive

4.4 Open each of the buckets by clicking on the arrow (>) next to the bucket names. Check that the files we had placed are displayed properly.





4.5 Because Cyberduck is a GUI based tool, all of the usage is very intuitive. Click on the cogwheel to see a list of actions available.

We can create new folders, rename files and folders. duplicate files and folders, download and upload files from s3 buckets and to s3 buckets, delete files and folders, and much more.

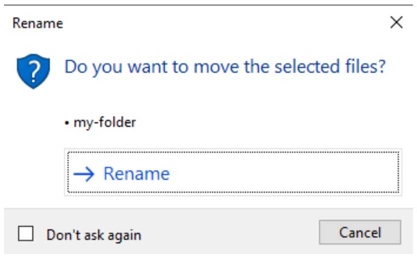
4.6 Rename the my-folder in student-cli-bucket to Q1-Sales.

4.6.1 Click on the my-folder icon.

4.6.2 Select Rename from the cogwheel menu

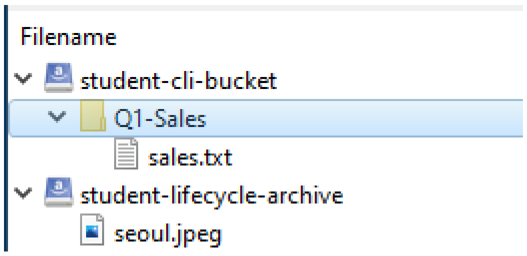
4.6.3 Change the my-folder name to Q1-Sales and hit enter

4.6.4 You will be prompted if you want to move the selected files and will be given an option to Rename. Click on the Rename


Rename

Do you want to move the selected files?

• my-folder


Filename
student-cli-bucket
Q1-Sales
sales.txt
student-lifecycle-archive
seoul.jpeg

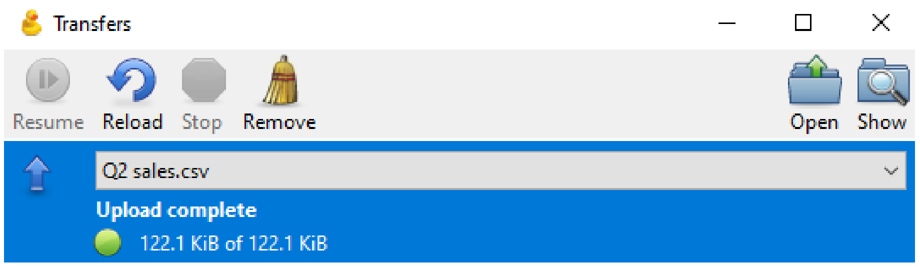
4.7 Upload a file to the Q1-Sales folder

4.7.1 Select the Q1-Sales folder

4.7.2 Select Upload from the cogwheel menu or from the top menu

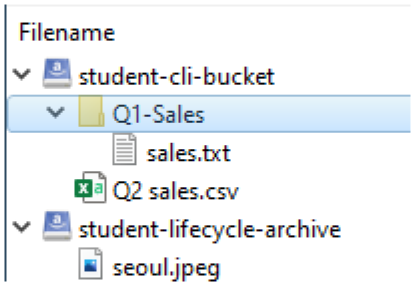
4.7.3 Navigate to the directory containing the source file and select it.

4.7.4 Click **Choose** to begin uploading the file. A new pop up window will display the progress of the upload.



Q2 sales.csv
Upload complete
122.1 KiB of 122.1 KiB

4.7.5 Notice that Cyberduck displays different icons depending on the file type


Filename
student-cli-bucket
Q1-Sales
sales.txt
Q2 sales.csv
student-lifecycle-archive
seoul.jpeg

4.8 Try renaming the student-cli-bucket. Will Cyberduck allow this operation?

Unfortunately, it will not. Remember, bucket names must be globally unique, therefore it is beyond the scope of Cyberduck to check for its uniqueness.

4.9 Select the Q2 sales.csv file and drag it into the <prefix>-lifecycle-archive bucket. Does it work as expected?

Yes, Cyberduck prompt you to verify that you want to move the file. When you click on the move button, the file is moved to the destination bucket.

4.10 Now try creating a new folder on your own. Create the folder in student-cli-bucket bucket.

4.11 Download the Q1 sales.csv file and then delete it from S3.

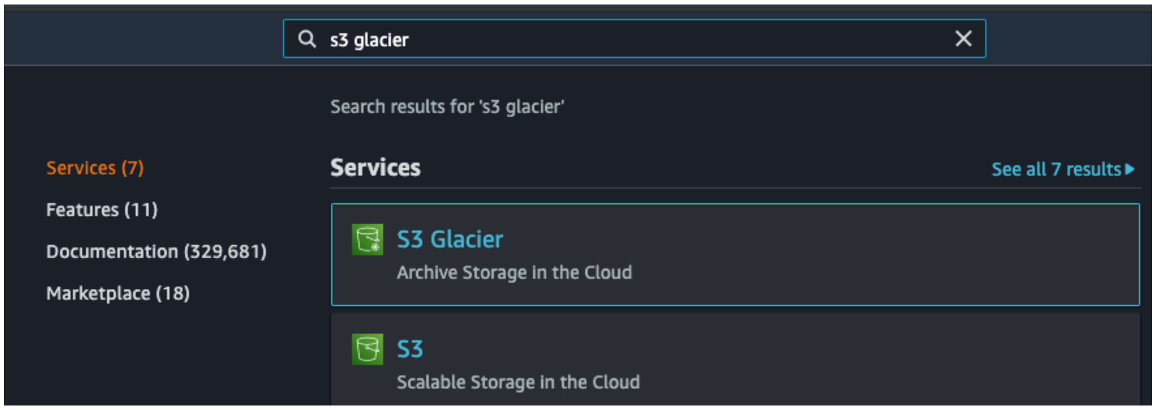
Lab 2: AWS S3 Glacier Storage

AWS S3 Glacier storage is a cost-effective alternative to S3 standard storage for files that are cold and not expected to be accessed for a long time.

1. Create an AWS Glacier Vault

   1.1 Login to the AWS Management Console and navigate to the Glacier service

      1.1.1 You can get to any Amazon AWS service in several ways.
      Use the search bar on top and search for the service



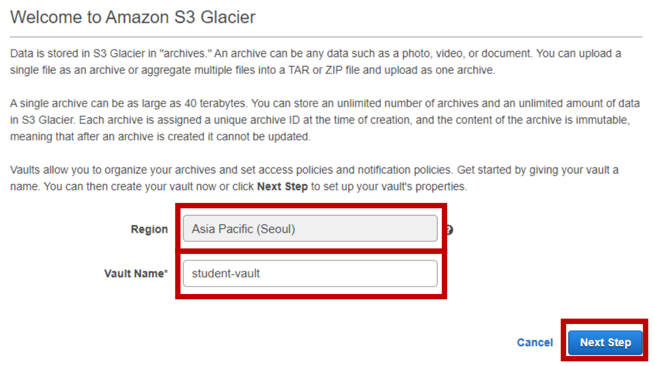
      1.1.2 Go direct to https://console.aws.amazon.com/glacier/

   1.2 Select **Create Vault** from the splash page to create a S3 Glacier vault

      1.2.1 Select an AWS Region close to your current location

      1.2.2 Provide a name for the vault

      1.2.3 Click **Next Step** to continue



Welcome to Amazon S3 Glacier

Data is stored in S3 Glacier in "archives." An archive can be any data such as a photo, video, or document. You can upload a single file as an archive or aggregate multiple files into a TAR or ZIP file and upload as one archive.

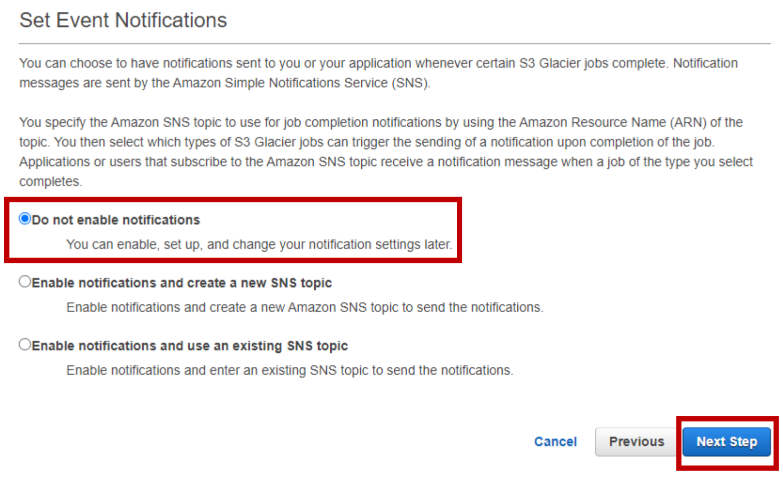
A single archive can be as large as 40 terabytes. You can store an unlimited number of archives and an unlimited amount of data in S3 Glacier. Each archive is assigned a unique archive ID at the time of creation, and the content of the archive is immutable, meaning that after an archive is created it cannot be updated.

Vaults allow you to organize your archives and set access policies and notification policies. Get started by giving your vault a name. You can then create your vault now or click **Next Step** to set up your vault's properties.

Region: Asia Pacific (Seoul)

Vault Name*: student-vault

   1.3 Keep the setting to **Do not enable notification** and click continue


Set Event Notifications

You can choose to have notifications sent to you or your application whenever certain S3 Glacier jobs complete. Notification messages are sent by the Amazon Simple Notifications Service (SNS).

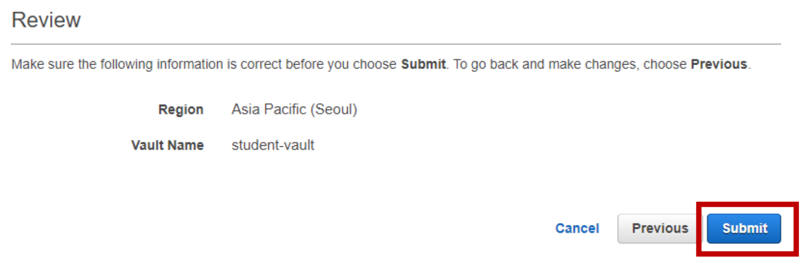
You specify the Amazon SNS topic to use for job completion notifications by using the Amazon Resource Name (ARN) of the topic. You then select which types of S3 Glacier jobs can trigger the sending of a notification upon completion of the job. Applications or users that subscribe to the Amazon SNS topic receive a notification message when a job of the type you select completes.

[✓ **Do not enable notifications**]
You can enable, set up, and change your notification settings later.

Enable notifications and create a new SNS topic
Enable notifications and create a new Amazon SNS topic to send the notifications.

Enable notifications and use an existing SNS topic
Enable notifications and enter an existing SNS topic to send the notifications.

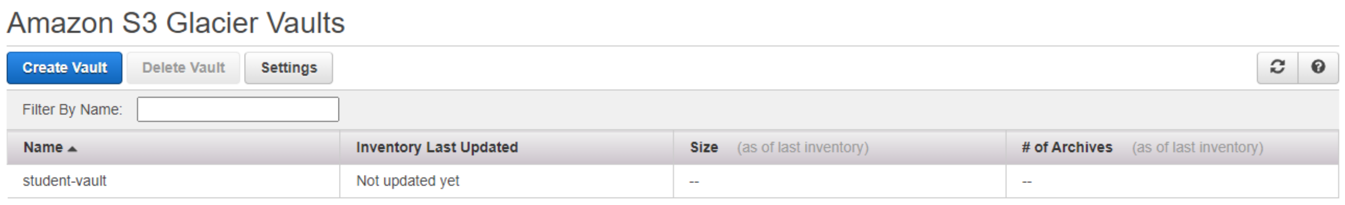
1.4 Check the information and **Submit**


Review

Make sure the following information is correct before you choose **Submit**. To go back and make changes, choose **Previous**.

Region	Asia Pacific (Seoul)
Vault Name	student-vault

1.5 Your new vault is listed in the Amazon S3 Glacier Vault page


Amazon S3 Glacier Vaults

Filter By Name:

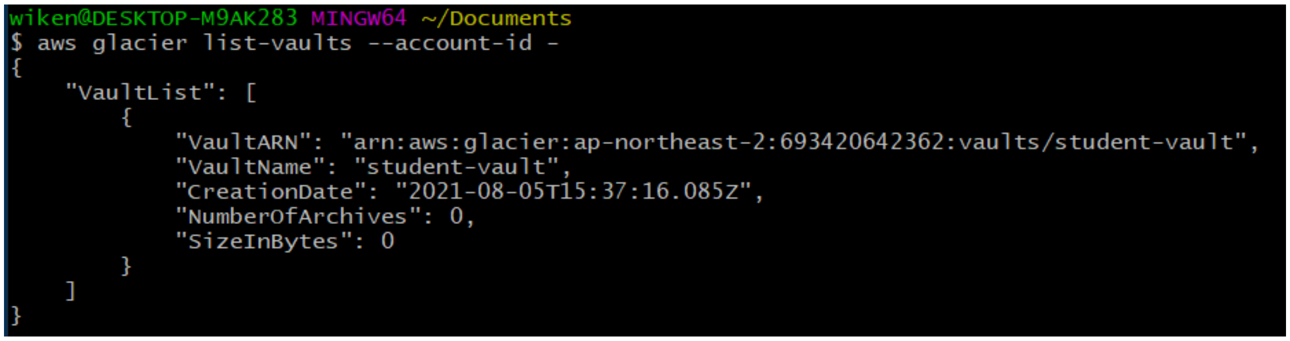
Name ▲	Inventory Last Updated	Size (as of last inventory)	# of Archives (as of last inventory)
student-vault	Not updated yet	--	--

2. Using AWS CLI to work with S3 Glacier vaults

2.1 Check to see that the Glacier vault was properly created. From Git Bash (terminal) issue the following command. You must include the hyphen at the end of the command. This tells AWS CLI to use the account id associated with the information we entered in step **0** as part of configuring AWS CLI.

```
aws glacier list-vaults --account-id -
```

This command will return the name of that vault in json format.

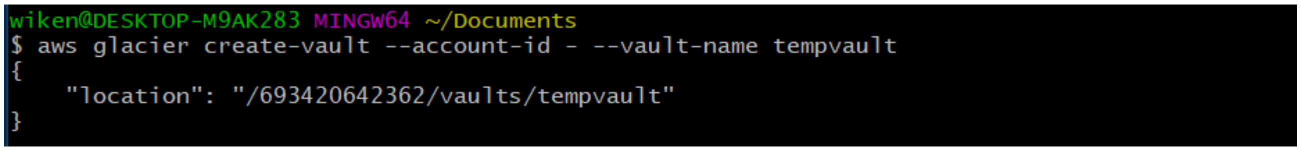


```
wiken@DESKTOP-M9AK283 MINGW64 ~/Documents
$ aws glacier list-vaults --account-id -
{
    "VaultList": [
        {
            "VaultARN": "arn:aws:glacier:ap-northeast-2:693420642362:vaults/student-vault",
            "VaultName": "student-vault",
            "CreationDate": "2021-08-05T15:37:16.085Z",
            "NumberOfArchives": 0,
            "SizeInBytes": 0
        }
    ]
}
```

2.2 Uploading a large file using multi-upload to S3 Glacier using AWS CLI

2.2.1 Create a S3 Glacier vault with the create-vault command

```
aws glacier create-vault --account-id - --vault-name tempvault
```

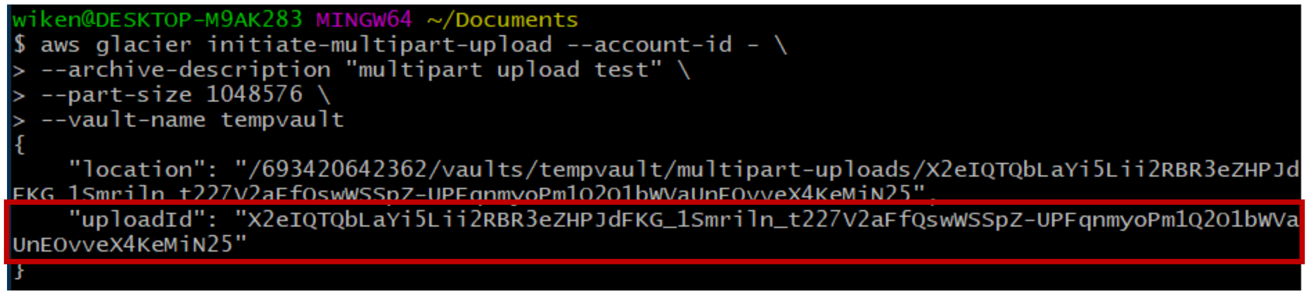


```
wiken@DESKTOP-M9AK283 MINGW64 ~/Documents
$ aws glacier create-vault --account-id - --vault-name tempvault
{
    "location": "/693420642362/vaults/tempvault"
}
```

2.2.2 Create a multipart upload to S3 Glacier with the initiate-multipart-upload command

```
aws glacier initiate-multipart-upload --account-id - \
--archive-description "multipart upload test" \
--part-size 1048576 \
--vault-name tempvault
```

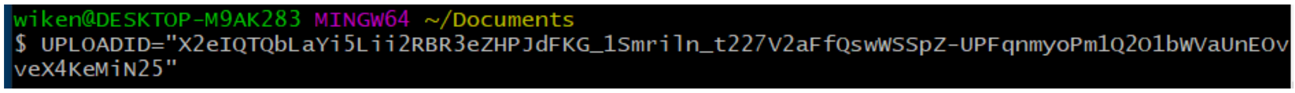
This command initiates a multipart upload and requires the size of each part in bytes. Our chunks are 1MiB each. We also have to give it a vault name and our account-id information. We will need the **uploadId** information that is returned after running this command



```
wiken@DESKTOP-M9AK283 MINGW64 ~/Documents
$ aws glacier initiate-multipart-upload --account-id - \
> --archive-description "multipart upload test" \
> --part-size 1048576 \
> --vault-name tempvault
{
    "location": "/693420642362/vaults/tempvault/multipart-uploads/X2eIQTQbLaYi5Lii2RBR3eZHPJd
FKG_1Smriln_t227V2aFfQswWSSpZ-UPFqnmyoPm1Q2O1bWVaUnEOvveX4KeMiN25",
    "uploadId": "X2eIQTQbLaYi5Lii2RBR3eZHPJdFKG_1Smriln_t227V2aFfQswWSSpZ-UPFqnmyoPm1Q2O1bWVa
UnEOvveX4KeMiN25"
}
```

2.3 Setup an environmental variable named UPLOADID and set it to the **uploadId** information returned above. Be careful that you don't introduce any extra characters or space within the command and especially the **uploadId** string.

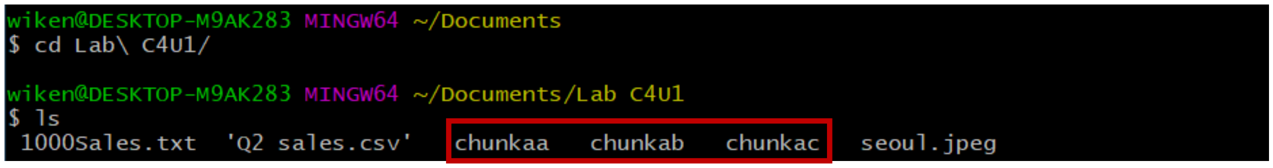
```
UPLOADID="<your upload id from above step>"
```



```
wiken@DESKTOP-M9AK283 MINGW64 ~/Documents
$ UPLOADID="X2eIQTQbLaYi5Lii2RBR3eZHPJdFKG_1Smriln_t227V2aFfQswWSSpZ-UPFqnmyoPm1Q2O1bWVaUnEOv
veX4KeMiN25"
```

2.4 Use the upload-multipart-part command to upload each part. These commands can be run in parallel for faster throughput.

2.4.1 Make sure that files named 3 parts are in the current directory. If they are not, either copy them to the current directory or move to the directory that has the 3 files.



```
wiken@DESKTOP-M9AK283 MINGW64 ~/Documents
$ cd Lab\ C4U1/

wiken@DESKTOP-M9AK283 MINGW64 ~/Documents/Lab C4U1
$ ls
 1000Sales.txt  'Q2 sales.csv'   chunkaa   chunkab   chunkac   seoul.jpeg
```

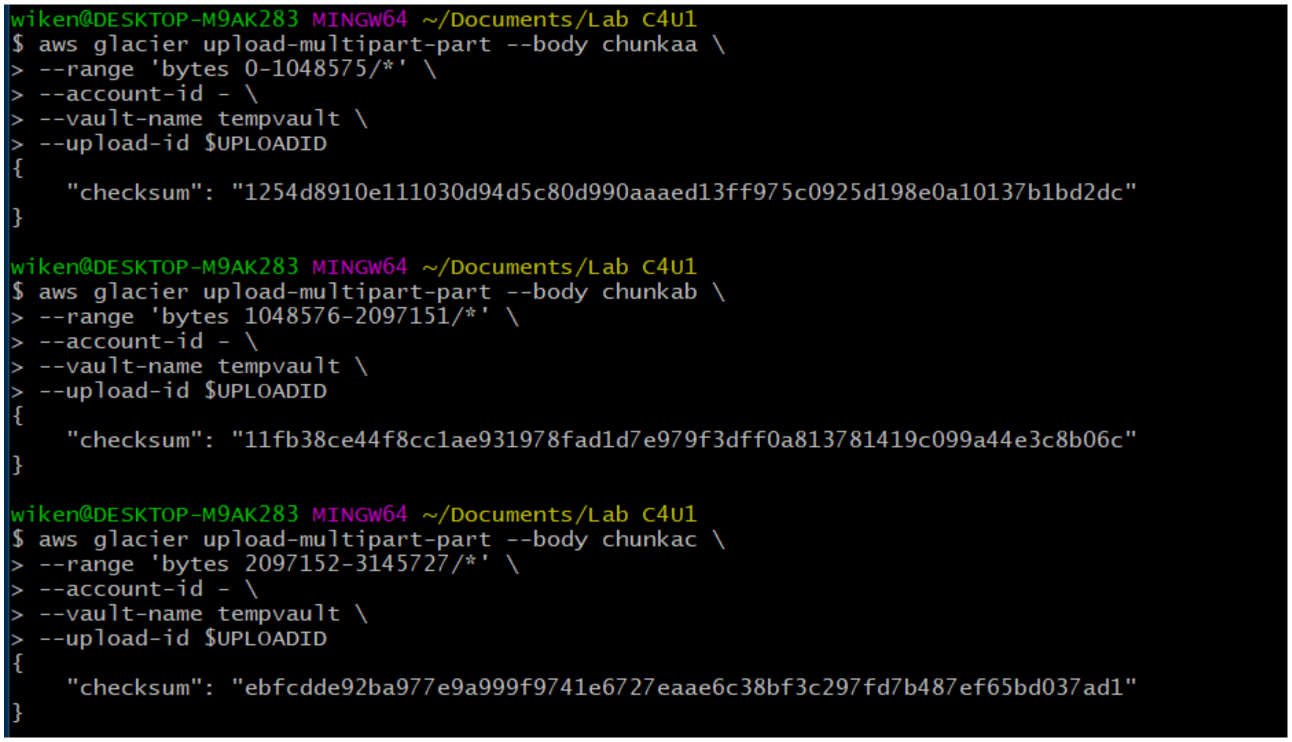
2.4.2 Execute the following command:

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

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

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

You will get 3 results similar to:



```
wiken@DESKTOP-M9AK283 MINGW64 ~/Documents/Lab C4U1
$ aws glacier upload-multipart-part --body chunkaa \
> --range 'bytes 0-1048575/*' \
> --account-id - \
> --vault-name tempvault \
> --upload-id $UPLOADID
{
    "checksum": "1254d8910e111030d94d5c80d990aaaed13ff975c0925d198e0a10137b1bd2dc"
}

wiken@DESKTOP-M9AK283 MINGW64 ~/Documents/Lab C4U1
$ aws glacier upload-multipart-part --body chunkab \
> --range 'bytes 1048576-2097151/*' \
> --account-id - \
> --vault-name tempvault \
> --upload-id $UPLOADID
{
    "checksum": "11fb38ce44f8cc1ae931978fad1d7e979f3dff0a813781419c099a44e3c8b06c"
}

wiken@DESKTOP-M9AK283 MINGW64 ~/Documents/Lab C4U1
$ aws glacier upload-multipart-part --body chunkac \
> --range 'bytes 2097152-3145727/*' \
> --account-id - \
> --vault-name tempvault \
> --upload-id $UPLOADID
{
    "checksum": "ebfcdde92ba977e9a999f9741e6727eaae6c38bf3c297fd7b487ef65bd037ad1"
}
```

2.5 Checking for proper transmission

Amazon S3 Glacier requires a tree hash of the original file to ensure that all uploaded fragments have reached AWS unchanged.

To compute the tree hash, split the file into 1 MiB parts and compute the binary SHA-256 hash of each piece. Then split the hash list into pairs, combine the two binary hashes into each pair, and get the hash of the result. Repeat this process until only one hash remains. If there is an odd hash at any level, it is promoted to the next level without modification.

The key to correctly calculating tree hashes when using command line utilities is to store each hash in binary format and only convert it to hexadecimal in the last step. Combining or hashing hashes of the hexadecimal version of the tree can produce incorrect results.

2.5.1 To calculate tree hash
Compute and store a binary SHA-256 hash of each chunk

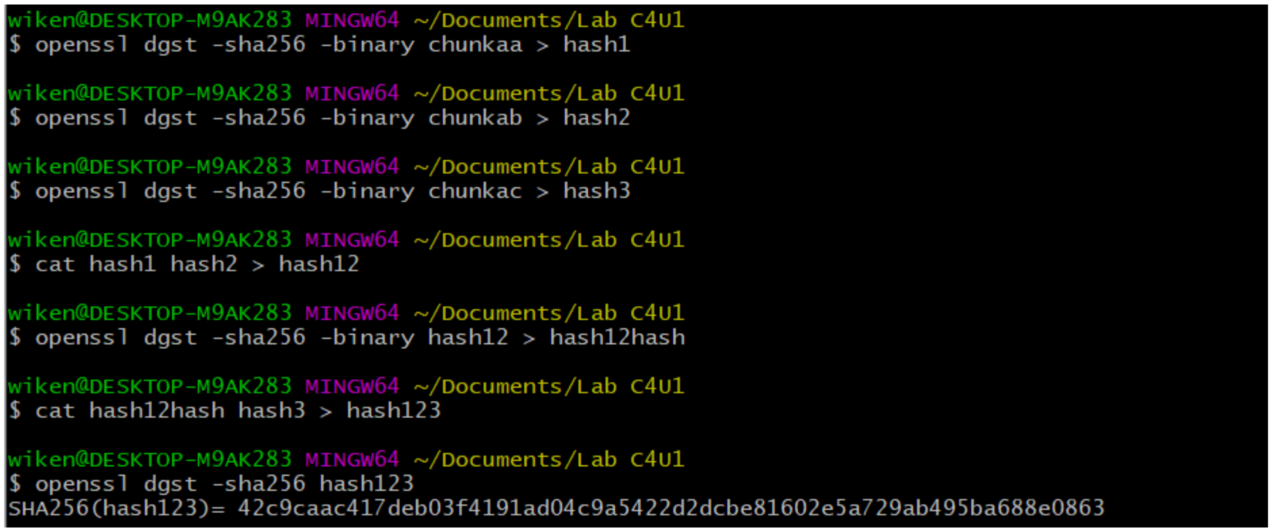
```
openssl dgst -sha256 -binary chunkaa > hash1
openssl dgst -sha256 -binary chunkab > hash2
openssl dgst -sha256 -binary chunkac > hash3
```

2.5.2 Combine the first 2 hashes and get the binary hash of the result.

```
cat hash1 hash2 > hash12
openssl dgst -sha256 -binary hash12 > hash12hash
```

2.5.3 Combine the parent hashes of chunks aa and ab with the hashes of chunk ac and hash the result again. At this time, the hexadecimal number is produced.

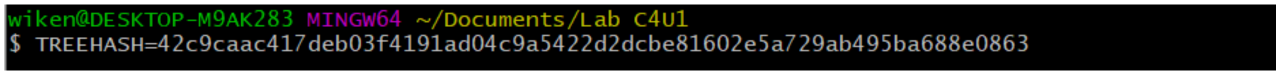
```
cat hash12hash hash3 > hash123
openssl dgst -sha256 hash123
```



```
wiken@DESKTOP-M9AK283 MINGW64 ~/Documents/Lab C4U1
$ openssl dgst -sha256 -binary chunkaa > hash1

wiken@DESKTOP-M9AK283 MINGW64 ~/Documents/Lab C4U1
$ openssl dgst -sha256 -binary chunkab > hash2

wiken@DESKTOP-M9AK283 MINGW64 ~/Documents/Lab C4U1
$ openssl dgst -sha256 -binary chunkac > hash3

wiken@DESKTOP-M9AK283 MINGW64 ~/Documents/Lab C4U1
$ cat hash1 hash2 > hash12

wiken@DESKTOP-M9AK283 MINGW64 ~/Documents/Lab C4U1
$ openssl dgst -sha256 -binary hash12 > hash12hash

wiken@DESKTOP-M9AK283 MINGW64 ~/Documents/Lab C4U1
$ cat hash12hash hash3 > hash123

wiken@DESKTOP-M9AK283 MINGW64 ~/Documents/Lab C4U1
$ openssl dgst -sha256 hash123
SHA256(hash123)= 42c9caac417deb03f4191ad04c9a5422d2dcbe81602e5a729ab495ba688e0863
```

2.5.4 Store the result in a shell variable.

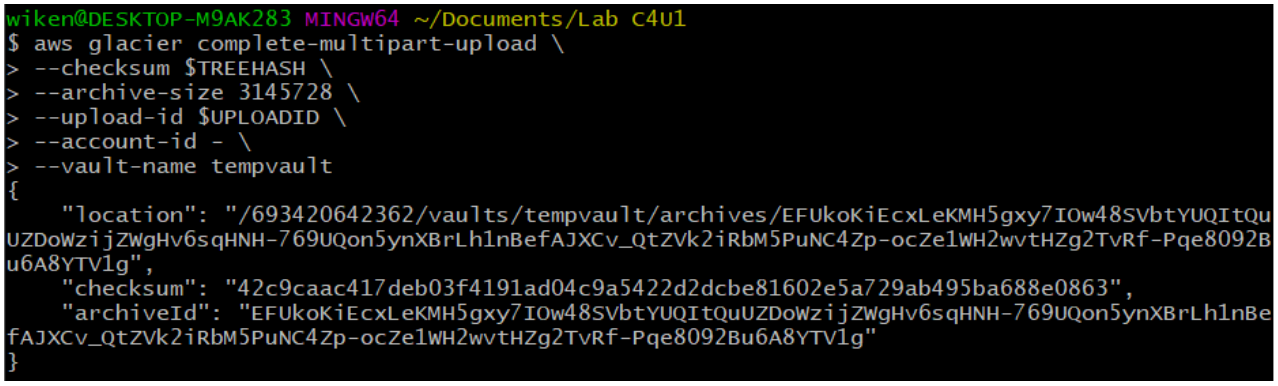
```
TREEHASH=<hexadecimal number produced above>
```



```
wiken@DESKTOP-M9AK283 MINGW64 ~/Documents/Lab C4U1
$ TREEHASH=42c9caac417deb03f4191ad04c9a5422d2dcbe81602e5a729ab495ba688e0863
```

2.5.5 Finally, use the complete-multipart-upload command to complete the upload. This command takes the size (in bytes) of the original file, the final tree hash value (in hexadecimal), and the account ID and vault name.

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



```
wiken@DESKTOP-M9AK283 MINGW64 ~/Documents/Lab C4U1
$ aws glacier complete-multipart-upload \
> --checksum $TREEHASH \
> --archive-size 3145728 \
> --upload-id $UPLOADID \
> --account-id - \
> --vault-name tempvault
{
    "location": "/693420642362/vaults/tempvault/archives/EFUkoKiEcxLeKMH5gxy7IOw48SVbtYUQItQu
UZDoWzijZWgHv6sqHNH-769UQon5ynXBrLh1nBefAJXCv_QtZVk2iRbM5PuNC4Zp-ocZe1WH2wvtHZg2TvRf-Pqe8092B
u6A8YTV1g",
    "checksum": "42c9caac417deb03f4191ad04c9a5422d2dcbe81602e5a729ab495ba688e0863",
    "archiveId": "EFUkoKiEcxLeKMH5gxy7IOw48SVbtYUQItQuUZDoWzijZWgHv6sqHNH-769UQon5ynXBrLh1nBe
fAJXCv_QtZVk2iRbM5PuNC4Zp-ocZe1WH2wvtHZg2TvRf-Pqe8092Bu6A8YTV1g"
}
```

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

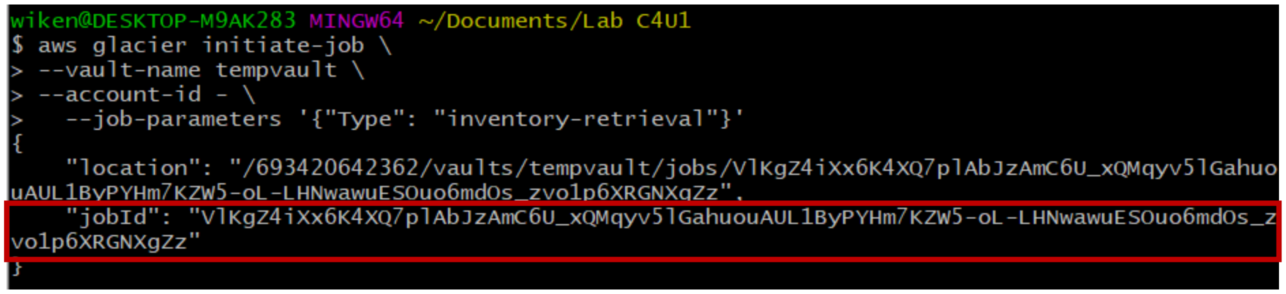
```
aws glacier describe-vault \
--account-id - \
--vault-name tempvault
```

Vault status is updated about once per day. You may not be able to see the files you have uploaded yet.

2.6 Downloading a Vault Inventory

2.6.1 Initiate an inventory retrieval job using the initiate-job command.

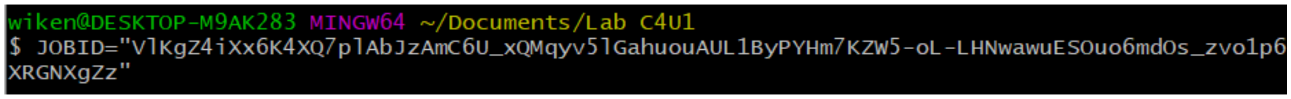
```
aws glacier initiate-job \
--vault-name tempvault \
--account-id - \
 --job-parameters '{"Type": "inventory-retrieval"}'
```



```
wiken@DESKTOP-M9AK283 MINGW64 ~/Documents/Lab C4U1
$ aws glacier initiate-job \
> --vault-name tempvault \
> --account-id - \
>   --job-parameters '{"Type": "inventory-retrieval"}'
{
    "location": "/693420642362/vaults/tempvault/jobs/VlKgZ4iXx6K4XQ7plAbJzAmC6U_xQMqyv5lGahuo
uAUL1ByPYHm7KZW5-oL-LHNwawuESOuo6mdOs_zvo1p6XRGNXgZz",
    "jobId": "VlKgZ4iXx6K4XQ7plAbJzAmC6U_xQMqyv5lGahuouAUL1ByPYHm7KZW5-oL-LHNwawuESOuo6mdOs_z
vo1p6XRGNXgZz"
}
```

2.6.2 Save the **jobId** from above in an environmental variable

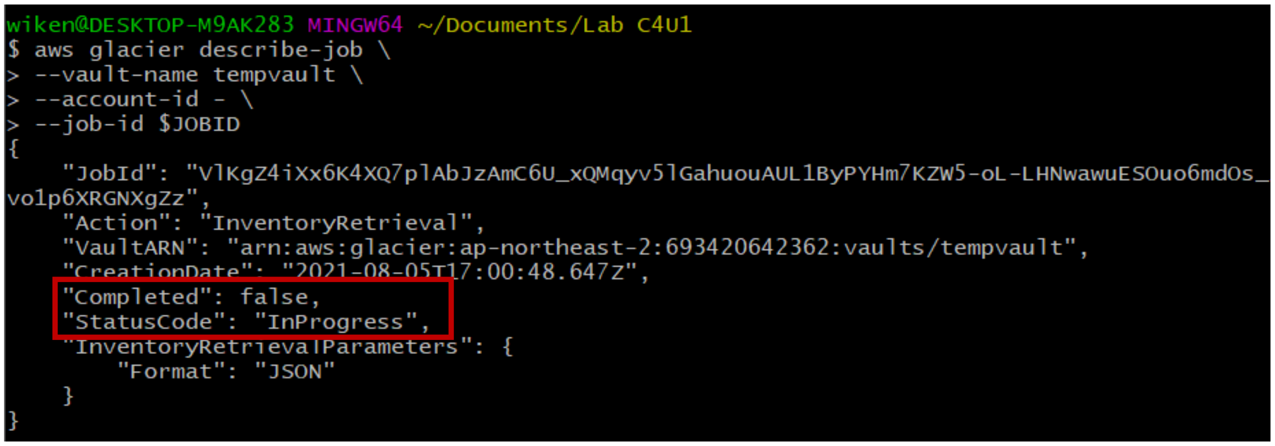
```
JOBID=<jobId from above>
```



```
wiken@DESKTOP-M9AK283 MINGW64 ~/Documents/Lab C4U1
$ JOBID="VlKgZ4iXx6K4XQ7plAbJzAmC6U_xQMqyv5lGahuouAUL1ByPYHm7KZW5-oL-LHNwawuESOuo6mdOs_zvo1p6
XRGNXgZz"
```

2.6.3 Use the describe-job command to check the status of a previous search job

```
aws glacier describe-job \
--vault-name tempvault \
--account-id - \
--job-id $JOBID
```



```
wiken@DESKTOP-M9AK283 MINGW64 ~/Documents/Lab C4U1
$ aws glacier describe-job \
> --vault-name tempvault \
> --account-id - \
> --job-id $JOBID
{
    "JobId": "VlKgZ4iXx6K4XQ7plAbJzAmC6U_xQMqyv5lGahuouAUL1ByPYHm7KZW5-oL-LHNwawuESOuo6mdOs_
vo1p6XRGNXgZz",
    "Action": "InventoryRetrieval",
    "VaultARN": "arn:aws:glacier:ap-northeast-2:693420642362:vaults/tempvault",
    "CreationDate": "2021-08-05T17:00:48.647Z",
    "Completed": false,
    "StatusCode": "InProgress",
    "InventoryRetrievalParameters": {
        "Format": "JSON"
    }
}
```

2.6.4 Wait for the download operation to complete. You can rerun the above command to keep checking on the completion status.
Job IDs don't expire for at least 24 hours after S3 Glacier completes the job

2.6.5 When finished, use the get-job-output command to download the status as output.json file.

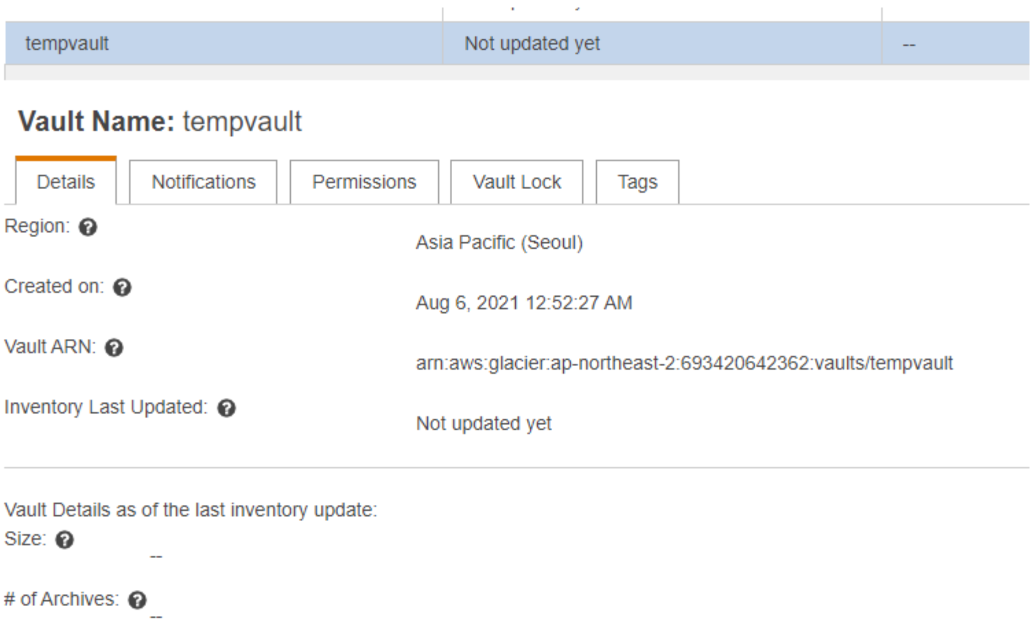
```
aws glacier get-job-output \

--vault-name tempvault \

--account-id - \

--job-id $JOBID output.json
```

3. Configuring Vault Notifications Using the Amazon S3 Glacier Console

3.1 From the AWS Management Console, navigate to the S3 Glacier console

3.1.1 Navigate to https://console.aws.amazon.com/glacier

3.2 We have been working with the tempvault in the previous steps. Click and select the tempvault. This will open up the following screen:



tempvault	Not updated yet	--

Vault Name: tempvault

Details | Notifications | Permissions | Vault Lock | Tags

Region:	Asia Pacific (Seoul)
Created on:	Aug 6, 2021 12:52:27 AM
Vault ARN:	arn:aws:glacier:ap-northeast-2:693420642362:vaults/tempvault
Inventory Last Updated:	Not updated yet

Vault Details as of the last inventory update:

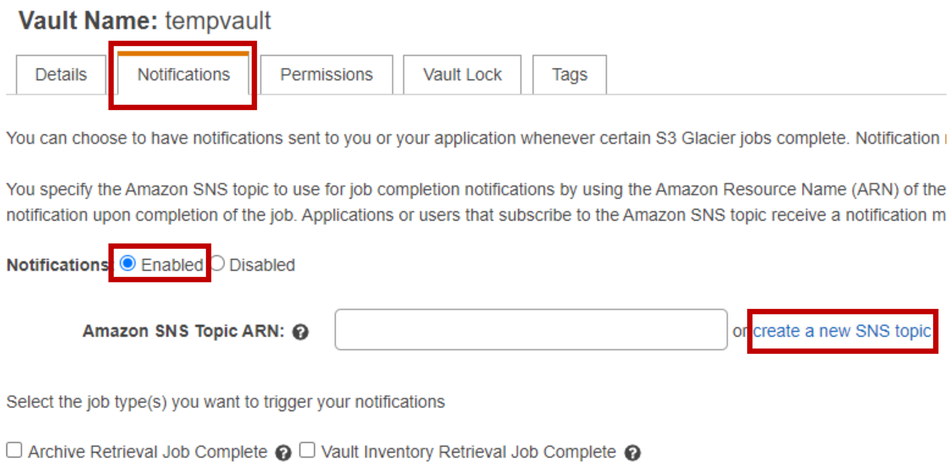
Size: --

# of Archives: --

3.3 Select the Notifications tab from the top to configure the notification

3.3.1 Enable the Notification

3.3.2 Select create a new SNS topic



Vault Name: tempvault

Details | Notifications | Permissions | Vault Lock | Tags

You can choose to have notifications sent to you or your application whenever certain S3 Glacier jobs complete. Notification

You specify the Amazon SNS topic to use for job completion notifications by using the Amazon Resource Name (ARN) of the notification upon completion of the job. Applications or users that subscribe to the Amazon SNS topic receive a notification m

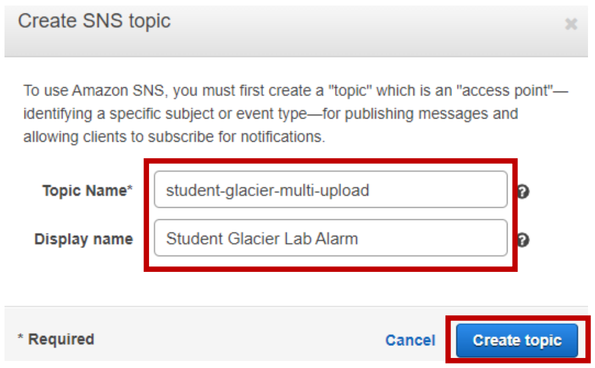
Notifications [✓ Enabled] Disabled

Amazon SNS Topic ARN: or create a new SNS topic

Select the job type(s) you want to trigger your notifications

☐ Archive Retrieval Job Complete ☐ Vault Inventory Retrieval Job Complete

3.3.3 Enter a topic name and display name and click **Create topic**


Create SNS topic

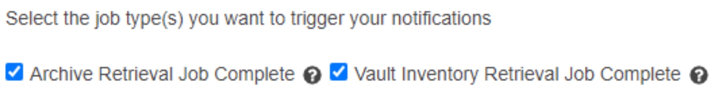
To use Amazon SNS, you must first create a "topic" which is an "access point"—identifying a specific subject or event type—for publishing messages and allowing clients to subscribe for notifications.

Topic Name*: student-glacier-multi-upload

Display name: Student Glacier Lab Alarm

* Required

3.3.4 Select the notification job type. Select both options.

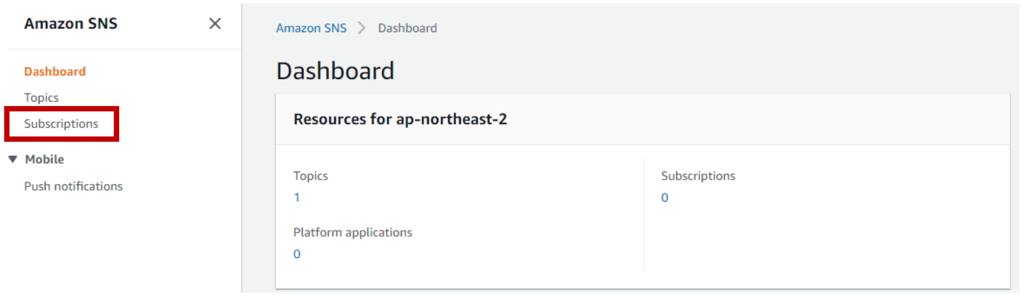

Select the job type(s) you want to trigger your notifications

☑ Archive Retrieval Job Complete ☑ Vault Inventory Retrieval Job Complete

3.3.5 Finally, **Save** the notification job

3.4 Create a topic subscription

3.4.1 Navigate to the Amazon Simple Notification Service console by searching for it from the AWS Management Console



Dashboard

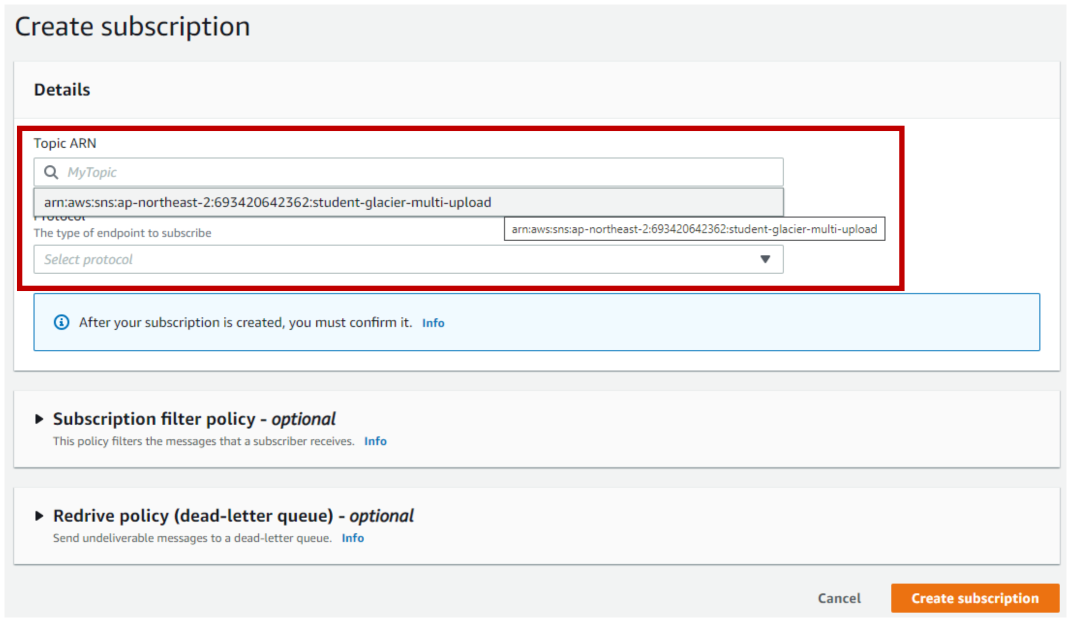
Resources for ap-northeast-2

Topics	Subscriptions
1	0
Platform applications	
0	

3.4.2 Select **Subscription** from the tab menu on the left

3.4.3 Click Create subscription

3.4.4 Select the Topic ARN that we created in the previous step



Create subscription

Details

Topic ARN

MyTopic

arn:aws:sns:ap-northeast-2:693420642362:student-glacier-multi-upload

The type of endpoint to subscribe

arn:aws:sns:ap-northeast-2:693420642362:student-glacier-multi-upload

Select protocol

After your subscription is created, you must confirm it. Info

Subscription filter policy - *optional*

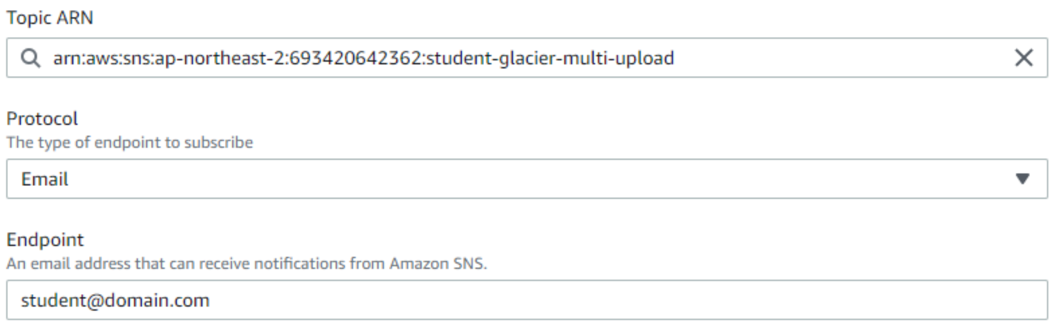
This policy filters the messages that a subscriber receives. Info

Redrive policy (dead-letter queue) - *optional*

Send undeliverable messages to a dead-letter queue. Info

3.4.5 Select Email as the **Protocol**

3.4.6 Enter an email to receive notification in the **Endpoint** section



Topic ARN

arn:aws:sns:ap-northeast-2:693420642362:student-glacier-multi-upload

Protocol

The type of endpoint to subscribe

Email

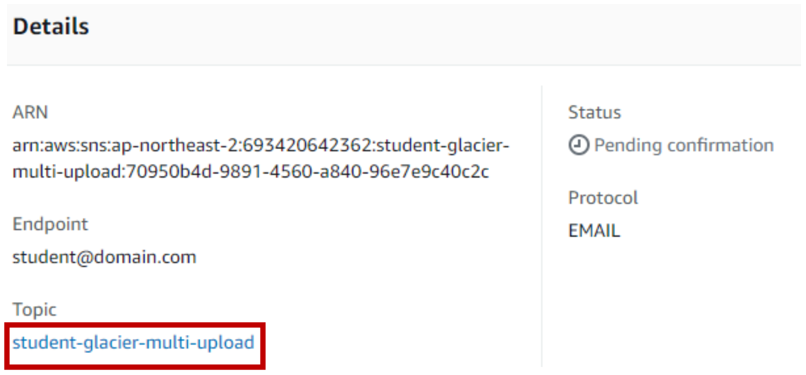
Endpoint

An email address that can receive notifications from Amazon SNS.

student@domain.com

3.4.7 Click **Create subscription** on the bottom right

3.4.8 A Details page will display the details of the subscription. Select on the topic that we created in the previous section.



Details

ARN

arn:aws:sns:ap-northeast-2:693420642362:student-glacier-multi-upload:70950b4d-9891-4560-a840-96e7e9c40c2c

Endpoint

student@domain.com

Topic

student-glacier-multi-upload

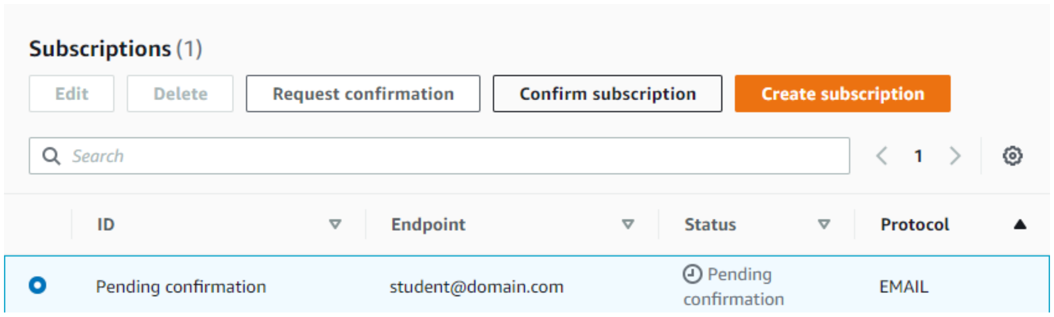
Status

Pending confirmation

Protocol

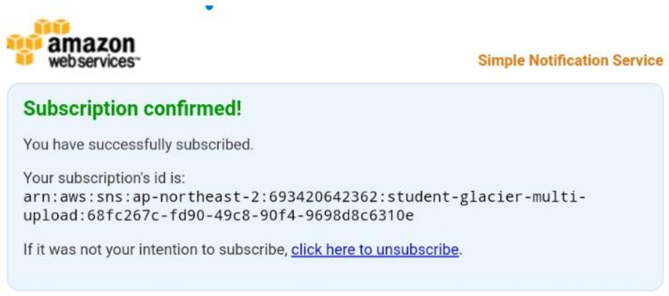
EMAIL

3.4.9 You will see that there is now a pending subscription waiting for confirmation. Check the email address entered in the subscription. There should be an email requesting confirmation.



ID	Endpoint	Status	Protocol
Pending confirmation	student@domain.com	Pending confirmation	EMAIL

3.4.10 Confirm the subscription. A new pop-up window will confirm that you are now subscribed.



Subscription confirmed!

You have successfully subscribed.

Your subscription's id is:
```
arn:aws:sns:ap-northeast-2:693420642362:student-glacier-multi-upload:68fc267c-fd90-49c8-90f4-9698d8c6310e
```

If it was not your intention to subscribe, click here to unsubscribe.

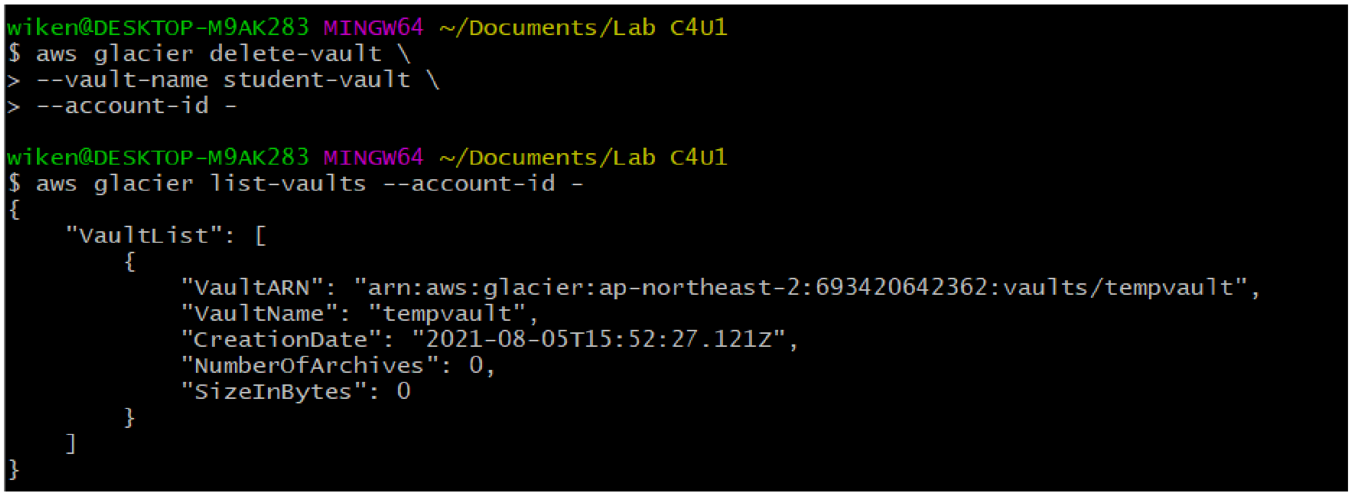
4. Deleting AWS S3 Glacier Vault

4.1 Using AWS CLI to a vault

4.1.1 Use the delete-vault command to remove a vault that does not have any archives

4.1.2 Delete the student-vault that we created in step 1.2.

```
aws glacier delete-vault \
--vault-name student-vault \
--account-id -
```


```
wiken@DESKTOP-M9AK283 MINGW64 ~/Documents/Lab C4U1
$ aws glacier delete-vault \
> --vault-name student-vault \
> --account-id -

wiken@DESKTOP-M9AK283 MINGW64 ~/Documents/Lab C4U1
$ aws glacier list-vaults --account-id -
{
    "VaultList": [
        {
            "VaultARN": "arn:aws:glacier:ap-northeast-2:693420642362:vaults/tempvault",
            "VaultName": "tempvault",
            "CreationDate": "2021-08-05T15:52:27.121Z",
            "NumberOfArchives": 0,
            "SizeInBytes": 0
        }
    ]
}
```

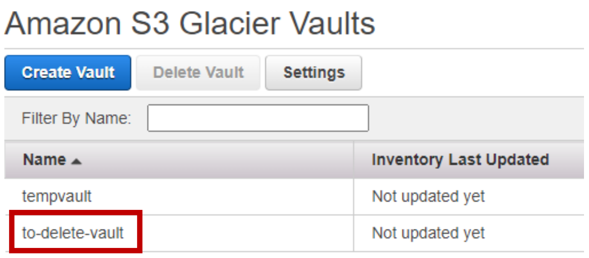
The student-vault is now deleted, however, we will keep the tempvault until all the pending jobs are completed. Since we have subscribed to a notification, wait for the confirmation notification that the job is complete.

4.2 Using the AWS Glacier console to delete a vault

4.2.1 Create a vault using AWS CLI so that we can delete it from the console

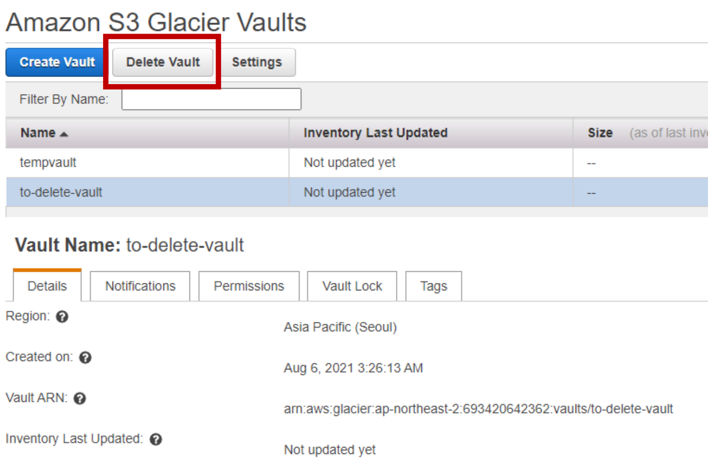
```
aws glacier create-vault \
--account-id - \
--vault-name to-delete-vault
```

4.2.2 Return to the console and refresh the screen to view all your vaults.



Name ▲	Inventory Last Updated
tempvault	Not updated yet
to-delete-vault	Not updated yet

4.2.3 Select the to-delete-vault. This will enable the Delete Vault option on the top menu


Amazon S3 Glacier Vaults

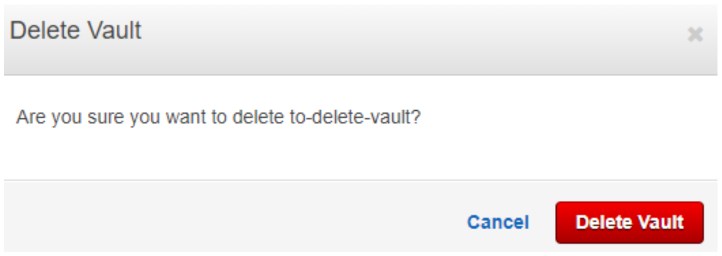
Filter By Name:

Name ▲	Inventory Last Updated	Size (as of last inv
tempvault	Not updated yet	--
to-delete-vault	Not updated yet	--

Vault Name: to-delete-vault

Region:	Asia Pacific (Seoul)
Created on:	Aug 6, 2021 3:26:13 AM
Vault ARN:	arn:aws:glacier:ap-northeast-2:693420642362:vaults/to-delete-vault
Inventory Last Updated:	Not updated yet

4.2.4 Click **Delete Vault** to delete the selected vault. Confirm the deletion.


Delete Vault

Are you sure you want to delete to-delete-vault?

Note: If the vault is not empty, you must first delete all archives. In order to delete the archive, use aws cli to start an "inventory scan" job. You can get a list of archive IDs from there. Delete all archives using aws glacier delete archive <archive id>.

Lab 3: Data Access with HBase

In this lab, you run to basic operation with creating, deleting, and altering a table in HBase shell. And you will use the shell to put and get data in HBase.

1. Start with HBase Shell

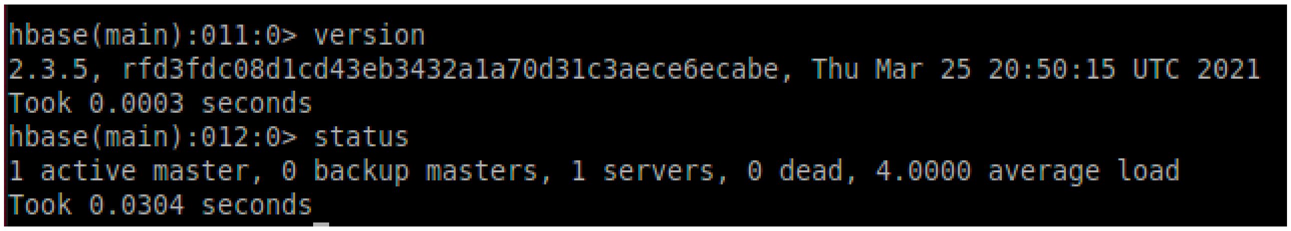
   1.1 Run the HBase shell. And execute the help command and view the basic usage information for HBase Shell.

```
$hbase shell
hbase(main):001:0> help
```

Note: The HBase shell prompt ends with a “>” character.

   1.2 Display the version and status for basic usage

```
hbase(main):001:0> version
hbase(main):001:0> status
```



```
hbase(main):011:0> version
2.3.5, rfd3fdc08d1cd43eb3432a1a70d31c3aece6ecabe, Thu Mar 25 20:50:15 UTC 2021
Took 0.0003 seconds
hbase(main):012:0> status
1 active master, 0 backup masters, 1 servers, 0 dead, 4.0000 average load
Took 0.0304 seconds
```

You can check the Hbase version (2.3.5) and standalone execution (1 active master, 1 servers).

   1.3 Use the create command to create a new table. You must specify the table name and the column family name.

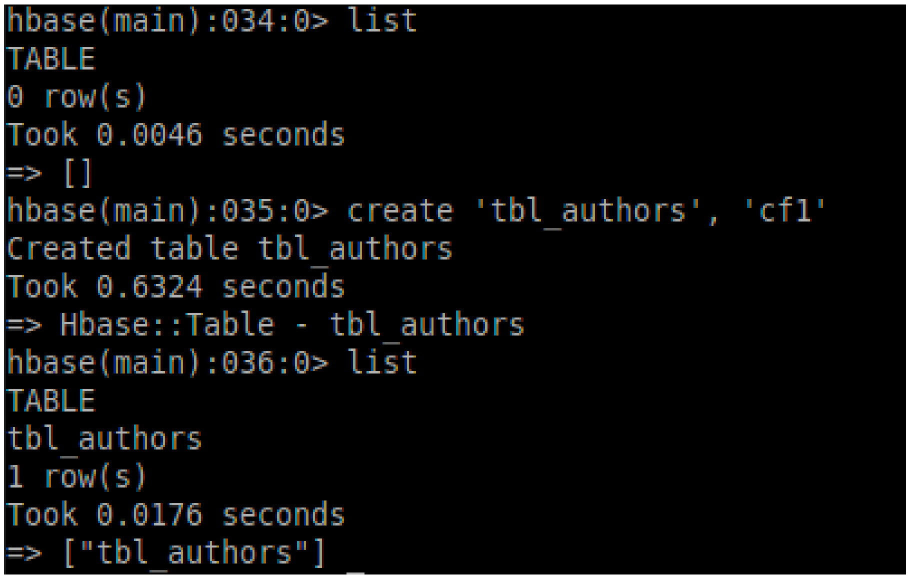
      1.3.1 Table name: tbl_authors, Column Family: cf1

```
hbase(main):001:0> create ‘tbl_authors’, ‘cf1’
```

Note: Table names, rows, columns all must be enclosed in quote mark.

   1.4 List table to verify table tbl_authors was created using list. You can either give the List command alone or give the table name along with List .

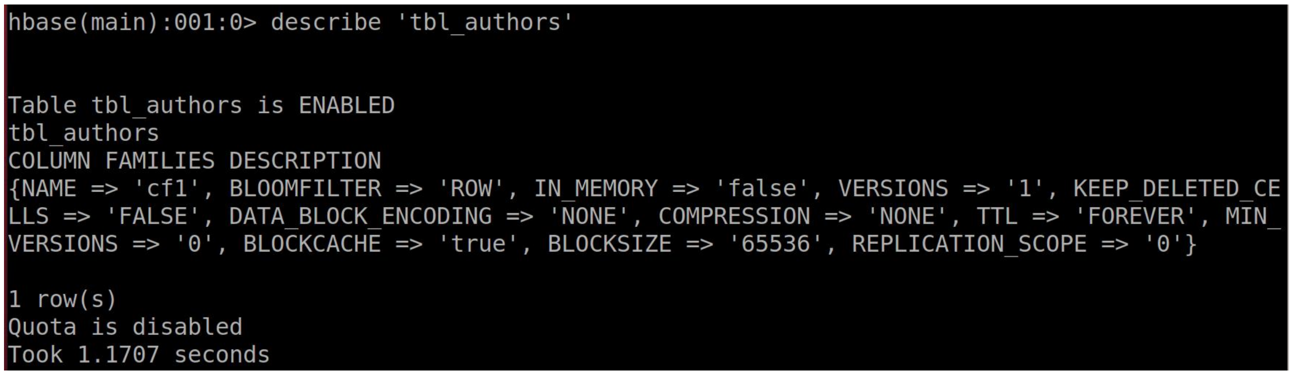
```
hbase(main):002:0> list

hbase(main):003:0> list ‘tbl_authors’
```



```
hbase(main):034:0> list
TABLE
0 row(s)
Took 0.0046 seconds
=> []
hbase(main):035:0> create 'tbl_authors', 'cf1'
Created table tbl_authors
Took 0.6324 seconds
=> Hbase::Table - tbl_authors
hbase(main):036:0> list
TABLE
tbl_authors
1 row(s)
Took 0.0176 seconds
=> ["tbl_authors"] _
```

1.5 Use the describe command to see details, including configuration defaults.

```
hbase(main):001:0> describe ‘tbl_authors’

hbase(main):002:0> describe
```



```
hbase(main):001:0> describe 'tbl_authors'


Table tbl_authors is ENABLED
tbl_authors
COLUMN FAMILIES DESCRIPTION
{NAME => 'cf1', BLOOMFILTER => 'ROW', IN_MEMORY => 'false', VERSIONS => '1', KEEP_DELETED_CE
LLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', COMPRESSION => 'NONE', TTL => 'FOREVER', MIN_
VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}

1 row(s)
Quota is disabled
Took 1.1707 seconds
```

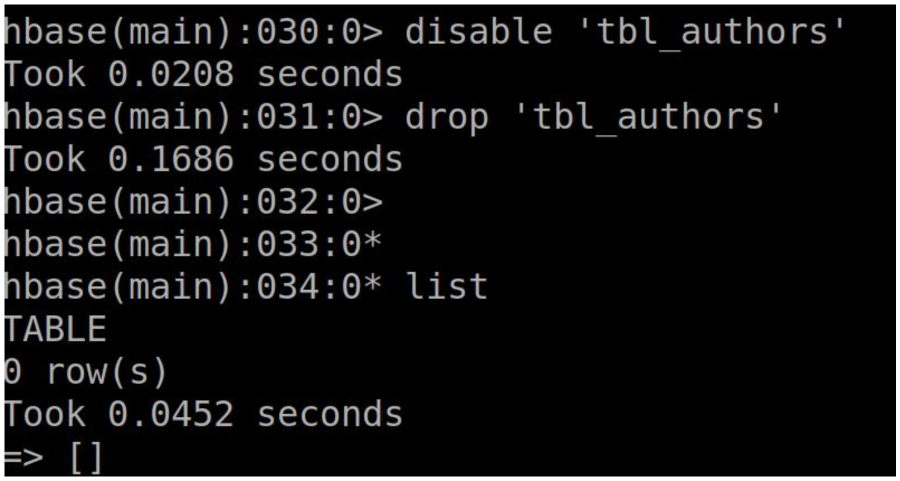
1.6 Delete the table you just created. And list all tables to verify table was successfully deleted

```
hbase(main):001:0> drop  ‘tbl_authors’
```

Note: Table tbl_authors is enabled now. Disable it first to delete it.

1.7 Disable the tbl_authors table for drop.

```
hbase(main):003:0> disable  ‘tbl_authors’

hbase(main):004:0> drop ‘tbl_authors’

hbase(main):005:0> list
```



```
hbase(main):030:0> disable 'tbl_authors'
Took 0.0208 seconds
hbase(main):031:0> drop 'tbl_authors'
Took 0.1686 seconds
hbase(main):032:0>
hbase(main):033:0*
hbase(main):034:0* list
TABLE
0 row(s)
Took 0.0452 seconds
=> []
```

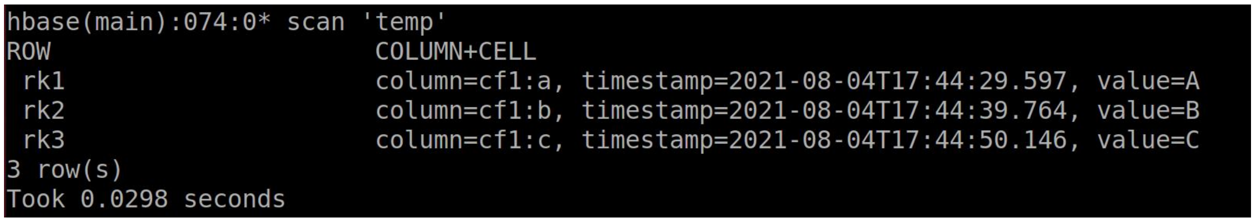
1.8 Create a test table, and use the put command. Here, we insert three values, one at a time.

1.8.1 The input data is in the following format.

1.8.2 Table name: temp, Column Family: cf1

	Column family		
Rowkey	cf1:a	cf1:b	cf1:c
rk1	A		
rk2		B	
rk3			C

```
hbase(main):003:0> create 'temp', 'cf1'

hbase(main):004:0> put 'temp', 'rk1', 'cf1:a', 'A'

hbase(main):005:0> put 'temp', 'rk2', 'cf1:b', 'B'

hbase(main):006:0> put 'temp', 'rk3', 'cf1:c', 'C'
```


```
hbase(main):074:0* scan 'temp'
ROW                     COLUMN+CELL
 rk1                    column=cf1:a, timestamp=2021-08-04T17:44:29.597, value=A
 rk2                    column=cf1:b, timestamp=2021-08-04T17:44:39.764, value=B
 rk3                    column=cf1:c, timestamp=2021-08-04T17:44:50.146, value=C
3 row(s)
Took 0.0298 seconds
```

1.9 Count the number of rows in the HBase temp table that you created in the previous lab

```
hbase(main):003:0> count ‘temp’
```

1.10 What is the expected number of rows after running the following command?

1.10.1 Input data under the following conditions into temp table.

Column Family: cf1

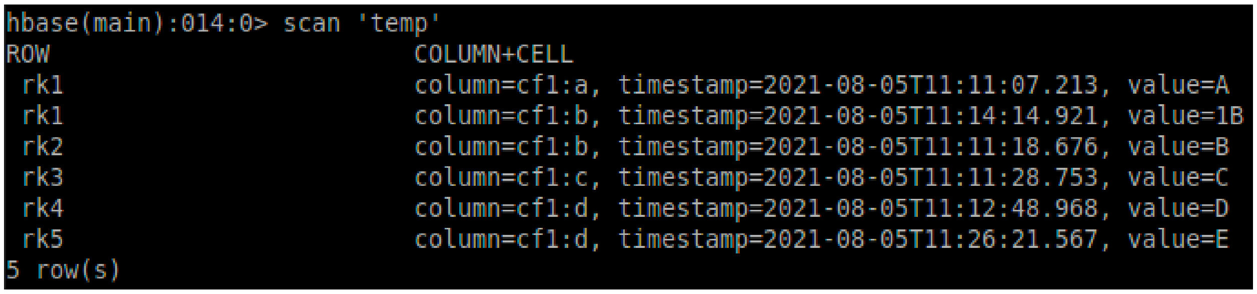
RowKey:rk4, column descriptor : d and value: D

RowKey:rk1, column descriptor : b and value: 1B

RowKey:rk5, column descriptor : d and value: E

```
hbase(main):003:0>  put 'temp', 'rk4', 'cf1:d', 'D'
hbase(main):003:0>  put 'temp', 'rk1', 'cf1:b', '1B'
hbase(main):003:0>  put 'temp', 'rk5', 'cf1:d', 'E'
hbase(main):003:0> scan  ‘temp’
```

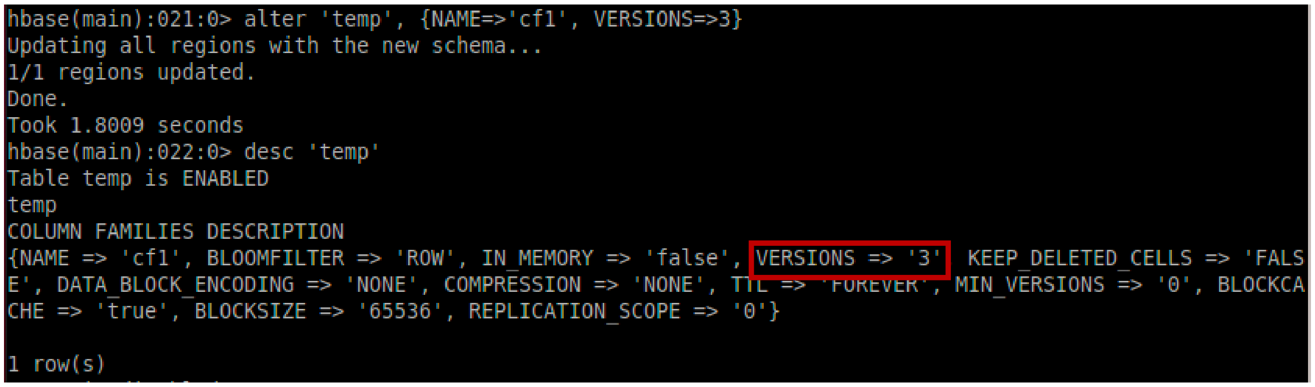
The following is the result of scanning the data in the TEMP table. The total list is 6, but the actual number of rows is 5.


```
hbase(main):014:0> scan 'temp'
ROW                        COLUMN+CELL
 rk1                       column=cf1:a, timestamp=2021-08-05T11:11:07.213, value=A
 rk1                       column=cf1:b, timestamp=2021-08-05T11:14:14.921, value=1B
 rk2                       column=cf1:b, timestamp=2021-08-05T11:11:18.676, value=B
 rk3                       column=cf1:c, timestamp=2021-08-05T11:11:28.753, value=C
 rk4                       column=cf1:d, timestamp=2021-08-05T11:12:48.968, value=D
 rk5                       column=cf1:d, timestamp=2021-08-05T11:26:21.567, value=E
5 row(s)
```

Note: There are two data in cf1 with ROWKEY rk=1.

1.11 Change the version attribute of cf1 in the temp table to 3, and check whether the change is reflected correctly.

```
hbase(main):003:0>  alter ‘temp’, {NAME=>'cf1', VERSIONS=>3}

hbase(main):004:0> desc ‘temp’
```

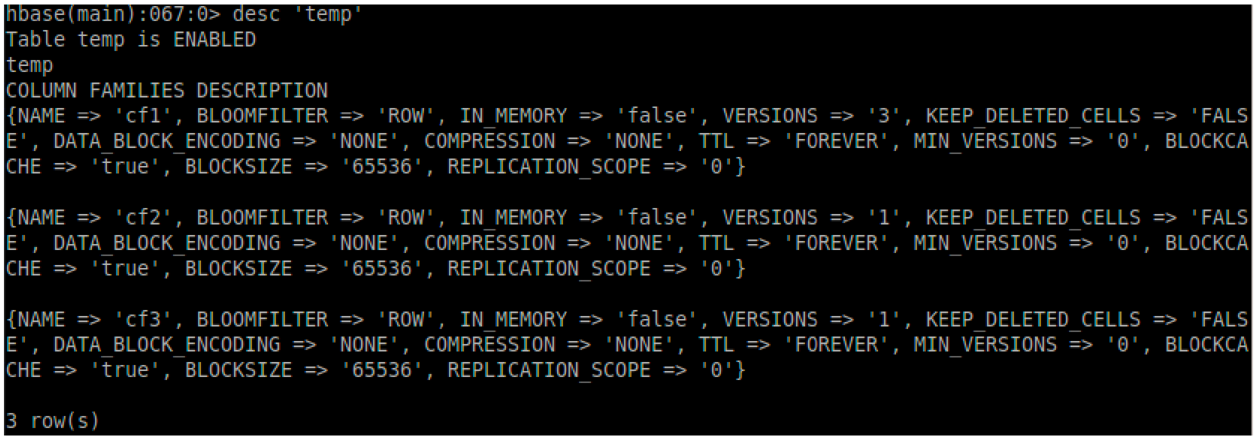


```
hbase(main):021:0> alter 'temp', {NAME=>'cf1', VERSIONS=>3}
Updating all regions with the new schema...
1/1 regions updated.
Done.
Took 1.8009 seconds
hbase(main):022:0> desc 'temp'
Table temp is ENABLED
temp
COLUMN FAMILIES DESCRIPTION
{NAME => 'cf1', BLOOMFILTER => 'ROW', IN_MEMORY => 'false', VERSIONS => '3', KEEP_DELETED_CELLS => 'FALS
E', DATA_BLOCK_ENCODING => 'NONE', COMPRESSION => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', BLOCKCA
CHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}

1 row(s)
```

1.12 Change the temp table to add the cf2 and cf3 column family using the ALTER command.

```
hbase(main):003:0>  alter 'temp', 'cf2', 'cf3'
```

1.13 Execute the command to check whether cf2 and cf3 have been added.



```
hbase(main):067:0> desc 'temp'
Table temp is ENABLED
temp
COLUMN FAMILIES DESCRIPTION
{NAME => 'cf1', BLOOMFILTER => 'ROW', IN_MEMORY => 'false', VERSIONS => '3', KEEP_DELETED_CELLS => 'FALS
E', DATA_BLOCK_ENCODING => 'NONE', COMPRESSION => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', BLOCKCA
CHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}

{NAME => 'cf2', BLOOMFILTER => 'ROW', IN_MEMORY => 'false', VERSIONS => '1', KEEP_DELETED_CELLS => 'FALS
E', DATA_BLOCK_ENCODING => 'NONE', COMPRESSION => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', BLOCKCA
CHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}

{NAME => 'cf3', BLOOMFILTER => 'ROW', IN_MEMORY => 'false', VERSIONS => '1', KEEP_DELETED_CELLS => 'FALS
E', DATA_BLOCK_ENCODING => 'NONE', COMPRESSION => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', BLOCKCA
CHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}

3 row(s)
```

1.14 Remove the newly added cf3 and check the result.

```
hbase(main):003:0> alter 'temp', {'delete' => 'cf3'}

hbase(main):004:0> desc ‘temp’
```

1.15 Delete the temp table. And list all tables to verify table was successfully deleted.

1.16 Exit HBase shell with the quit command

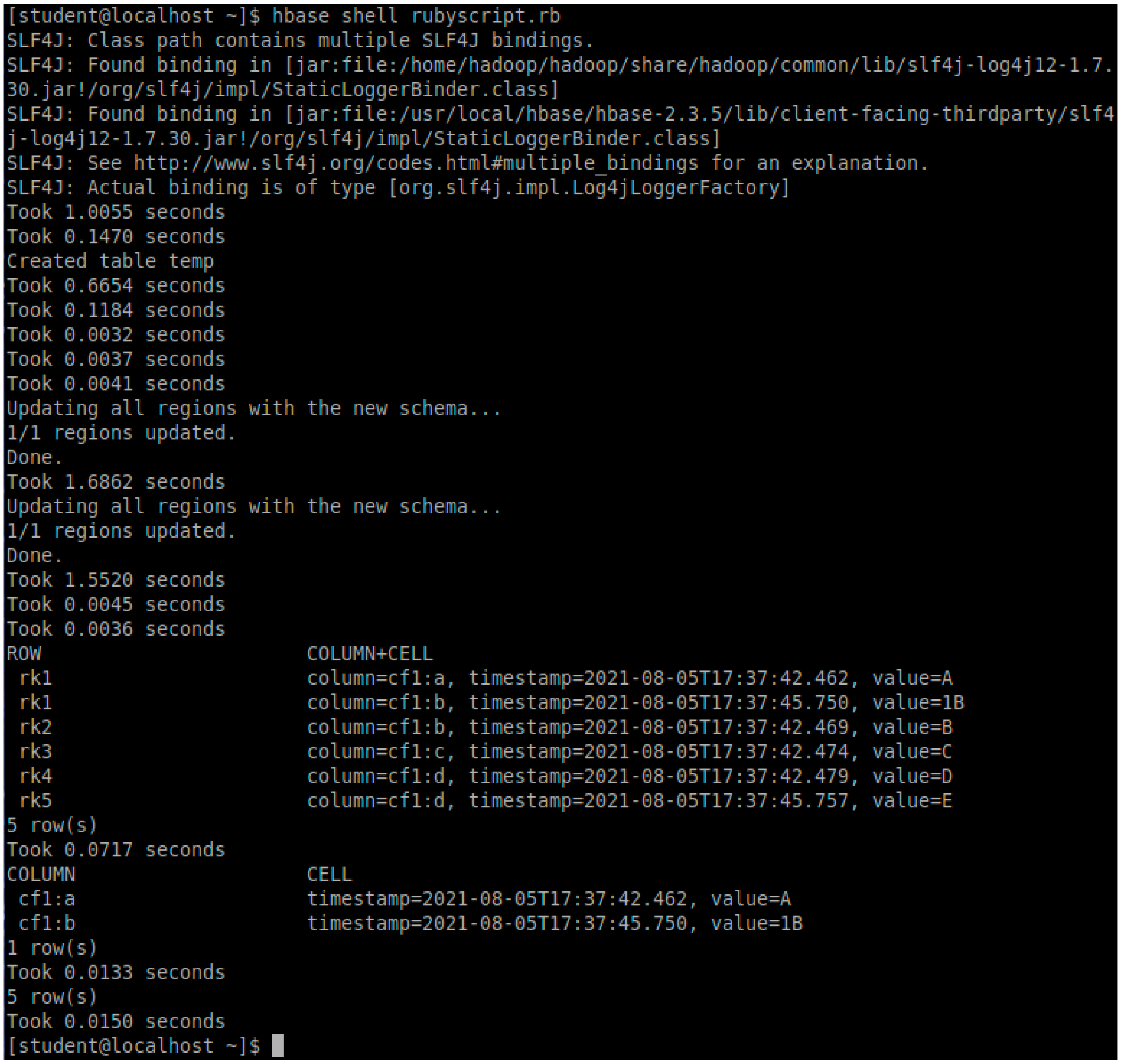
```
hbase(main):004:0> quit
```

1.17 Run HBase shell commands from a scripts file.

```
$vi rubyscript.rb
disable ‘temp’
drop ‘temp’

create 'temp', 'cf1'
put 'temp', 'rk1', 'cf1:a', 'A'
put 'temp', 'rk2', 'cf1:b', 'B'
put 'temp', 'rk3', 'cf1:c', 'C'
put 'temp', 'rk4', 'cf1:d', 'D'

alter 'temp', {NAME=>'cf1', VERSIONS=>3}
alter 'temp', 'cf2', 'cf3'
put 'temp', 'rk1', 'cf1:b', '1B'
put 'temp', 'rk5', 'cf1:d', 'E'

scan 'temp'
get 'temp', 'rk1'
count 'temp'

exit
```

1.18 You can enter HBase shell commands into a text file, one command per line, and pass the file to the shell.

```
hbase shell rubyscript.rb

hbase shell

hbase(main):001:0> require './rubyscript.rb'
```



```
[student@localhost ~]$ hbase shell rubyscript.rb
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.
30.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hbase/hbase-2.3.5/lib/client-facing-thirdparty/slf4
j-log4j12-1.7.30.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Took 1.0055 seconds
Took 0.1470 seconds
Created table temp
Took 0.6654 seconds
Took 0.1184 seconds
Took 0.0032 seconds
Took 0.0037 seconds
Took 0.0041 seconds
Updating all regions with the new schema...
1/1 regions updated.
Done.
Took 1.6862 seconds
Updating all regions with the new schema...
1/1 regions updated.
Done.
Took 1.5520 seconds
Took 0.0045 seconds
Took 0.0036 seconds
ROW                        COLUMN+CELL
 rk1                       column=cf1:a, timestamp=2021-08-05T17:37:42.462, value=A
 rk1                       column=cf1:b, timestamp=2021-08-05T17:37:45.750, value=1B
 rk2                       column=cf1:b, timestamp=2021-08-05T17:37:42.469, value=B
 rk3                       column=cf1:c, timestamp=2021-08-05T17:37:42.474, value=C
 rk4                       column=cf1:d, timestamp=2021-08-05T17:37:42.479, value=D
 rk5                       column=cf1:d, timestamp=2021-08-05T17:37:45.757, value=E
5 row(s)
Took 0.0717 seconds
COLUMN                     CELL
 cf1:a                     timestamp=2021-08-05T17:37:42.462, value=A
 cf1:b                     timestamp=2021-08-05T17:37:45.750, value=1B
1 row(s)
Took 0.0133 seconds
5 row(s)
Took 0.0150 seconds
[student@localhost ~]$
```

Note: This is a rubyscript.rb execution result screen, and if there is no exit statement in the script, it remains in the shell state.

1.19 You can also pass commands to the HBase Shell in non-interactive mode using the echo command and the | (pipe) operator.

```
echo "describe 'temp'" | hbase shell -n
```

Lab 4: Data Accessing Using DML commands

In this lab, you will use command to inserting, retrieving, scanning, and removing rows.

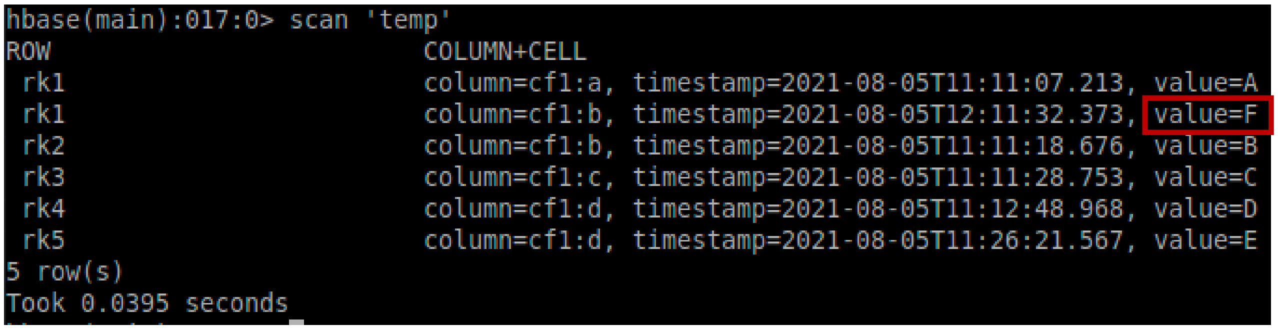
1. CRUD(Insert, select, update, delete) operations.
   1.1 If you have not finished lab1, run the ruby.rb script first and proceed with this lab2.
   1.2 Run the HBase shell:

```
hbase shell
```

   1.3 Enter data with the following conditions and display the results.

   RowKey:rk1, column descriptor : b and value: F

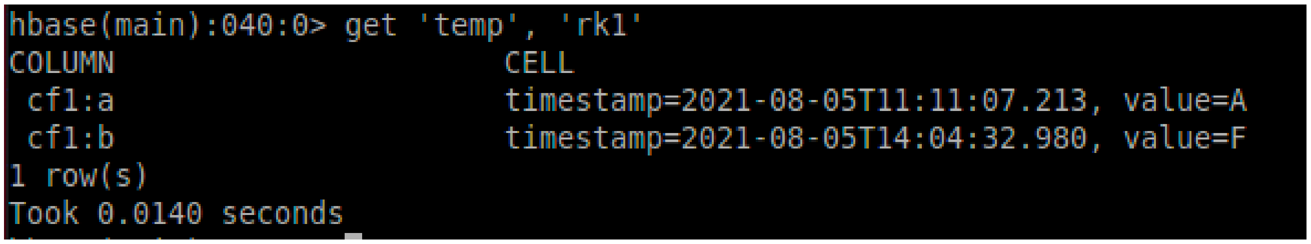
```
hbase(main):003:0>  put ‘temp’, ‘rk1’, ‘cf1:b’, ‘F’
hbase(main):004:0> scan ‘temp’
```



```
hbase(main):017:0> scan 'temp'
ROW                       COLUMN+CELL
 rk1                      column=cf1:a, timestamp=2021-08-05T11:11:07.213, value=A
 rk1                      column=cf1:b, timestamp=2021-08-05T12:11:32.373, value=F
 rk2                      column=cf1:b, timestamp=2021-08-05T11:11:18.676, value=B
 rk3                      column=cf1:c, timestamp=2021-08-05T11:11:28.753, value=C
 rk4                      column=cf1:d, timestamp=2021-08-05T11:12:48.968, value=D
 rk5                      column=cf1:d, timestamp=2021-08-05T11:26:21.567, value=E
5 row(s)
Took 0.0395 seconds
```

   1.4 Get the row with rowkey rk1 from temp table.

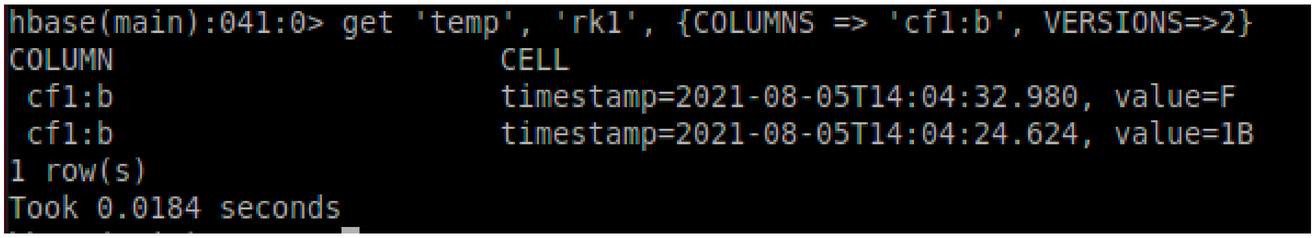
```
Hbase(main):003:0>  get ‘temp’, ‘rk1’
```



```
hbase(main):040:0> get 'temp', 'rk1'
COLUMN                    CELL
 cf1:a                    timestamp=2021-08-05T11:11:07.213, value=A
 cf1:b                    timestamp=2021-08-05T14:04:32.980, value=F
1 row(s)
Took 0.0140 seconds
```

   1.5 Get two values of the previous versions of the ‘b’ column with the rk1.

```
Hbase(main):003:0>  get 'temp', 'rk1', {COLUMNS => 'cf1:b', VERSIONS=>2}
hbase(main):004:0> get 'temp', 'rk1', {COLUMNS => 'cf1:b'}
```



```
hbase(main):041:0> get 'temp', 'rk1', {COLUMNS => 'cf1:b', VERSIONS=>2}
COLUMN                     CELL
 cf1:b                     timestamp=2021-08-05T14:04:32.980, value=F
 cf1:b                     timestamp=2021-08-05T14:04:24.624, value=1B
1 row(s)
Took 0.0184 seconds
```

Note: The first input 1B value and the last input F value are displayed. If the version is not specified, only the final value is displayed.

1.6 View the all table, but only show the 'a', 'b', 'c' columns.

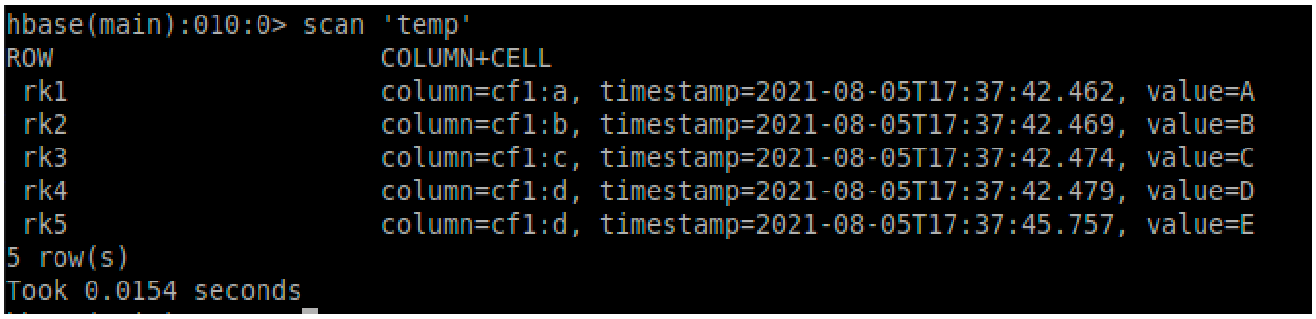
```
Hbase(main):003:0>  scan 'temp', {COLUMNS => ['cf1:b', 'cf1:a', 'cf1:c']}
```

1.7 Delete the 'b' column from the temp table with the rowkey = rk1.

```
Hbase(main):003:0>   delete 'temp', 'rk1', 'cf1:b'
```

1.8 Verify that 'b' column has been deleted.

```
Hbase(main):004:0> scan 'temp'
```

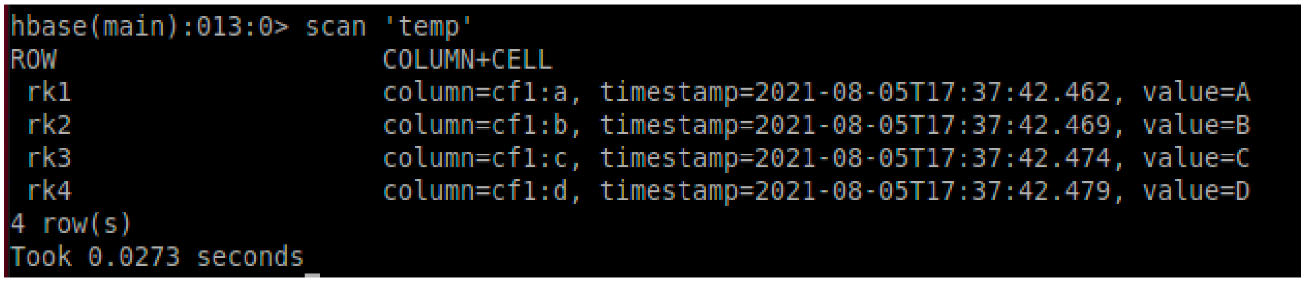


```
hbase(main):010:0> scan 'temp'
ROW                    COLUMN+CELL
 rk1                   column=cf1:a, timestamp=2021-08-05T17:37:42.462, value=A
 rk2                   column=cf1:b, timestamp=2021-08-05T17:37:42.469, value=B
 rk3                   column=cf1:c, timestamp=2021-08-05T17:37:42.474, value=C
 rk4                   column=cf1:d, timestamp=2021-08-05T17:37:42.479, value=D
 rk5                   column=cf1:d, timestamp=2021-08-05T17:37:45.757, value=E
5 row(s)
Took 0.0154 seconds
```

1.9 Delete the entire row form the temp table in rk1 rowkey.

```
Hbase(main):012:0> deleteall 'temp', 'rk5'
```

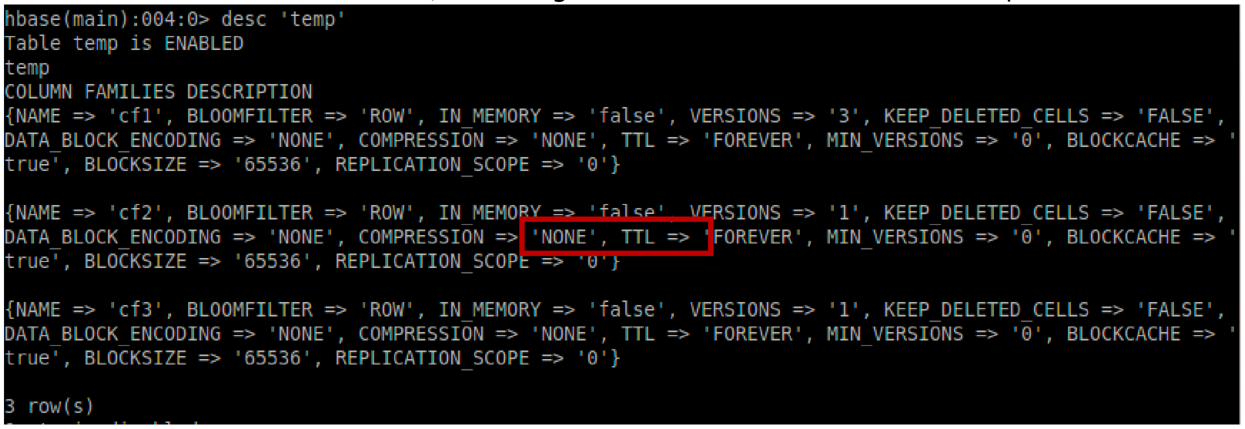
1.10 Show the row with rk1 has been deleted from temp table.



```
hbase(main):013:0> scan 'temp'
ROW                    COLUMN+CELL
 rk1                   column=cf1:a, timestamp=2021-08-05T17:37:42.462, value=A
 rk2                   column=cf1:b, timestamp=2021-08-05T17:37:42.469, value=B
 rk3                   column=cf1:c, timestamp=2021-08-05T17:37:42.474, value=C
 rk4                   column=cf1:d, timestamp=2021-08-05T17:37:42.479, value=D
4 row(s)
Took 0.0273 seconds
```

2. Using MIN_VERSIONS and Time-To-Live

   2.1 Describe the temp table and check the TTL value of cf2 column. The default TTL(Time-To-Live) is FOREVER, meaning that versions of a cell never expire.

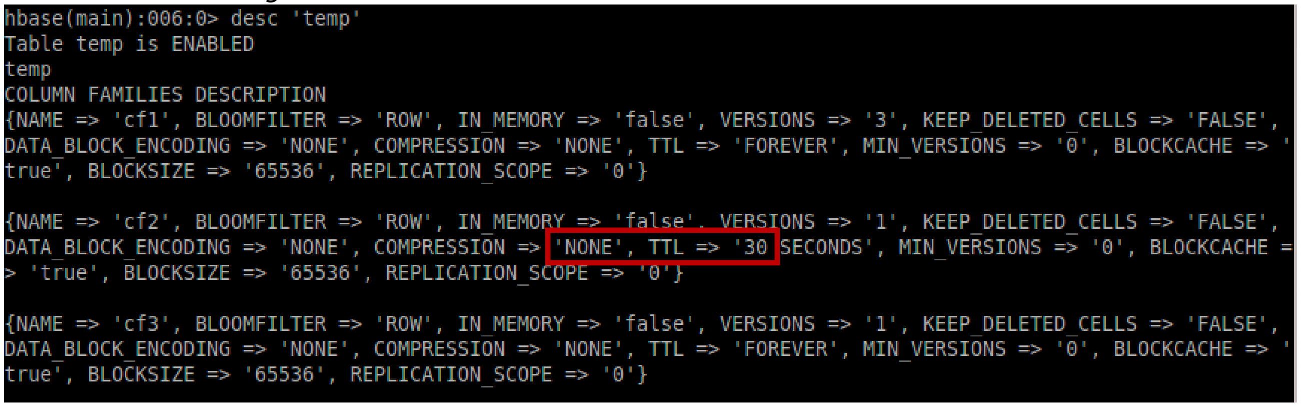


```
hbase(main):004:0> desc 'temp'
Table temp is ENABLED
temp
COLUMN FAMILIES DESCRIPTION
{NAME => 'cf1', BLOOMFILTER => 'ROW', IN_MEMORY => 'false', VERSIONS => '3', KEEP_DELETED_CELLS => 'FALSE',
DATA_BLOCK_ENCODING => 'NONE', COMPRESSION => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', BLOCKCACHE => '
true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}

{NAME => 'cf2', BLOOMFILTER => 'ROW', IN_MEMORY => 'false', VERSIONS => '1', KEEP_DELETED_CELLS => 'FALSE',
DATA_BLOCK_ENCODING => 'NONE', COMPRESSION => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', BLOCKCACHE => '
true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}

{NAME => 'cf3', BLOOMFILTER => 'ROW', IN_MEMORY => 'false', VERSIONS => '1', KEEP_DELETED_CELLS => 'FALSE',
DATA_BLOCK_ENCODING => 'NONE', COMPRESSION => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', BLOCKCACHE => '
true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}

3 row(s)
```

   2.2 Change the value from FOREVER to 30 seconds. This means the version will be deleted 30 seconds after inserting data into the table.

```
Base(main):005:0> alter 'temp', NAME => 'cf2', TTL=> 30
```

   2.3 Show the TTL changed from FOREVER to 30 secs.

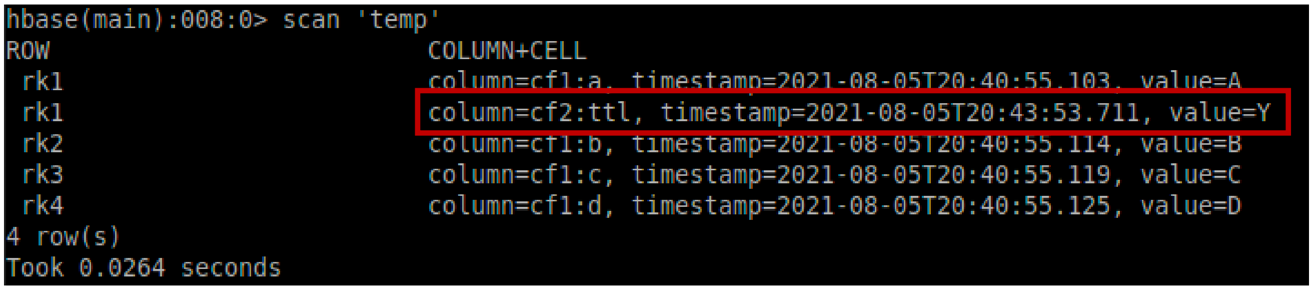


```
hbase(main):006:0> desc 'temp'
Table temp is ENABLED
temp
COLUMN FAMILIES DESCRIPTION
{NAME => 'cf1', BLOOMFILTER => 'ROW', IN_MEMORY => 'false', VERSIONS => '3', KEEP_DELETED_CELLS => 'FALSE',
DATA_BLOCK_ENCODING => 'NONE', COMPRESSION => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', BLOCKCACHE => '
true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}

{NAME => 'cf2', BLOOMFILTER => 'ROW', IN_MEMORY => 'false', VERSIONS => '1', KEEP_DELETED_CELLS => 'FALSE',
DATA_BLOCK_ENCODING => 'NONE', COMPRESSION => 'NONE', TTL => '30 SECONDS', MIN_VERSIONS => '0', BLOCKCACHE =
> 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}

{NAME => 'cf3', BLOOMFILTER => 'ROW', IN_MEMORY => 'false', VERSIONS => '1', KEEP_DELETED_CELLS => 'FALSE',
DATA_BLOCK_ENCODING => 'NONE', COMPRESSION => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', BLOCKCACHE => '
true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
```

   2.4 Insert a data with the following conditions and display the results.

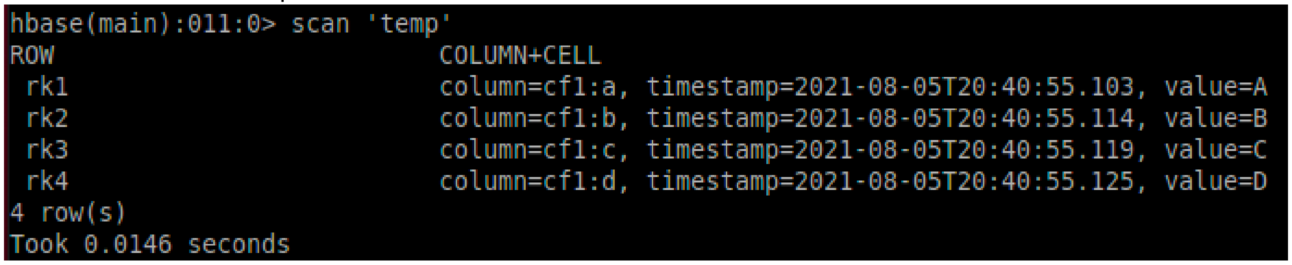
RowKey:rk1, Column Family: cf2, column descriptor : ttl and value: Y

```
hbase(main):007:0> put ‘temp’, ‘rk1’, ‘cf2:ttl’, ‘Y’
```



```
hbase(main):008:0> scan 'temp'
ROW                        COLUMN+CELL
 rk1                       column=cf1:a, timestamp=2021-08-05T20:40:55.103, value=A
 rk1                       column=cf2:ttl, timestamp=2021-08-05T20:43:53.711, value=Y
 rk2                       column=cf1:b, timestamp=2021-08-05T20:40:55.114, value=B
 rk3                       column=cf1:c, timestamp=2021-08-05T20:40:55.119, value=C
 rk4                       column=cf1:d, timestamp=2021-08-05T20:40:55.125, value=D
4 row(s)
Took 0.0264 seconds
```

2.5 After waiting at least 30 seconds, run the scan command again to see if the inserted row has expired and has been deleted.



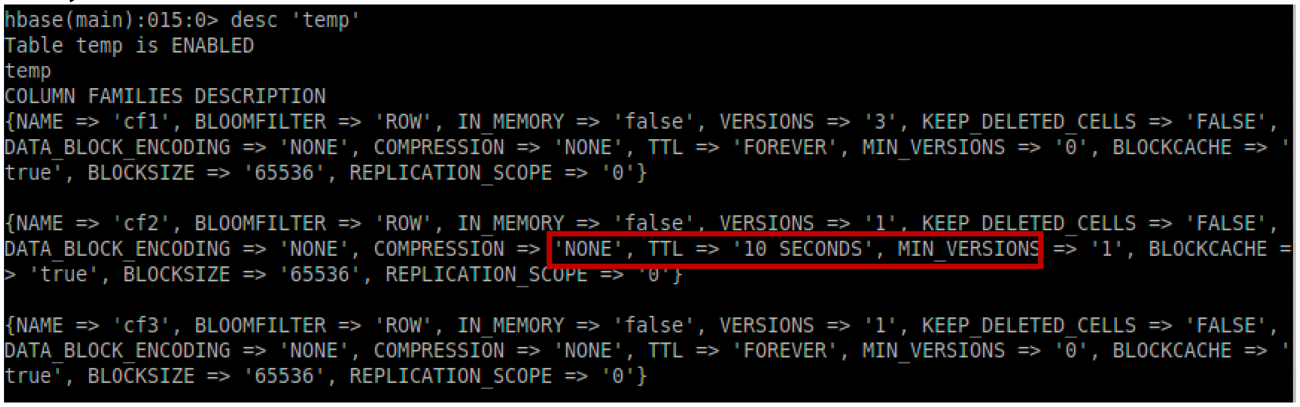
```
hbase(main):011:0> scan 'temp'
ROW                        COLUMN+CELL
 rk1                       column=cf1:a, timestamp=2021-08-05T20:40:55.103, value=A
 rk2                       column=cf1:b, timestamp=2021-08-05T20:40:55.114, value=B
 rk3                       column=cf1:c, timestamp=2021-08-05T20:40:55.119, value=C
 rk4                       column=cf1:d, timestamp=2021-08-05T20:40:55.125, value=D
4 row(s)
Took 0.0146 seconds
```

2.6 Set TTL to 10 seconds and set MIN_VERSIONS to 1 with cf2.

```
Hbase(main):014:0> alter ‘temp’, NAME => ‘cf2’, TTL=> 10, MIN_VERSIONS => 1
```

Note: When a version expires with MIN_VERSION option, it will not be deleted if it is the only remaining version of the cell.

2.7 Verify the TTL and MIN_VERSIONS value in cf2.

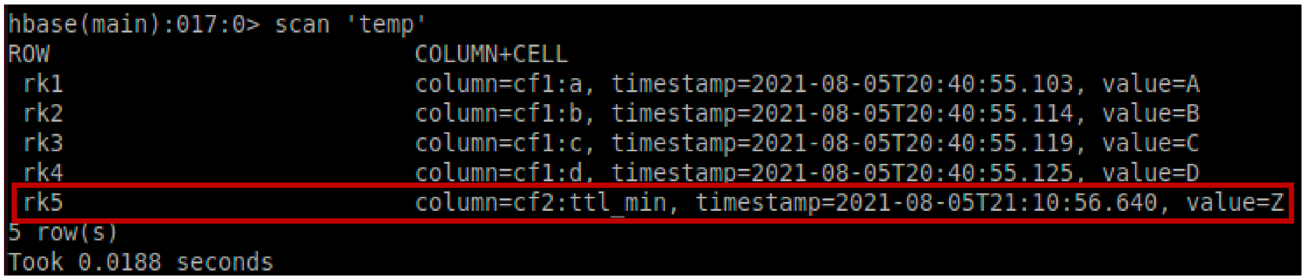


```
hbase(main):015:0> desc 'temp'
Table temp is ENABLED
temp
COLUMN FAMILIES DESCRIPTION
{NAME => 'cf1', BLOOMFILTER => 'ROW', IN_MEMORY => 'false', VERSIONS => '3', KEEP_DELETED_CELLS => 'FALSE',
DATA_BLOCK_ENCODING => 'NONE', COMPRESSION => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', BLOCKCACHE => '
true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}

{NAME => 'cf2', BLOOMFILTER => 'ROW', IN_MEMORY => 'false', VERSIONS => '1', KEEP_DELETED_CELLS => 'FALSE',
DATA_BLOCK_ENCODING => 'NONE', COMPRESSION => 'NONE', TTL => '10 SECONDS', MIN_VERSIONS => '1', BLOCKCACHE =
> 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}

{NAME => 'cf3', BLOOMFILTER => 'ROW', IN_MEMORY => 'false', VERSIONS => '1', KEEP_DELETED_CELLS => 'FALSE',
DATA_BLOCK_ENCODING => 'NONE', COMPRESSION => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', BLOCKCACHE => '
true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
```

2.8 Insert a data with the following conditions and display the results.

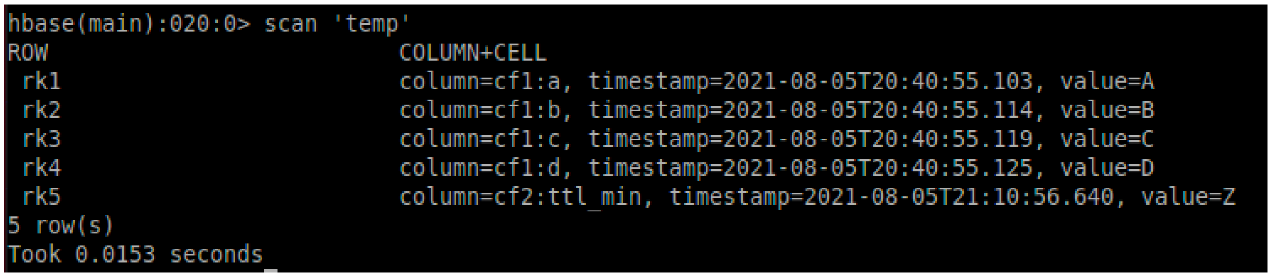
RowKey:rk5, Column Family: cf2, column descriptor : ttl_min and value: Z

```
hbase(main):016:0> put ‘temp’, ‘rk5’, ‘cf2:ttl_min’, ‘Z’
```



```
hbase(main):017:0> scan 'temp'
ROW                             COLUMN+CELL
 rk1                            column=cf1:a, timestamp=2021-08-05T20:40:55.103, value=A
 rk2                            column=cf1:b, timestamp=2021-08-05T20:40:55.114, value=B
 rk3                            column=cf1:c, timestamp=2021-08-05T20:40:55.119, value=C
 rk4                            column=cf1:d, timestamp=2021-08-05T20:40:55.125, value=D
 rk5                            column=cf2:ttl_min, timestamp=2021-08-05T21:10:56.640, value=Z
5 row(s)
Took 0.0188 seconds
```

2.9 After waiting at least 10 seconds, run the scan command again to see if the inserted row has expired and has been remained.



```
hbase(main):020:0> scan 'temp'
ROW                             COLUMN+CELL
 rk1                            column=cf1:a, timestamp=2021-08-05T20:40:55.103, value=A
 rk2                            column=cf1:b, timestamp=2021-08-05T20:40:55.114, value=B
 rk3                            column=cf1:c, timestamp=2021-08-05T20:40:55.119, value=C
 rk4                            column=cf1:d, timestamp=2021-08-05T20:40:55.125, value=D
 rk5                            column=cf2:ttl_min, timestamp=2021-08-05T21:10:56.640, value=Z
5 row(s)
Took 0.0153 seconds
```

Lab 5: Working with HBase

In this lab, you will work with HBase.

1. Write a command for creating table that meets the following conditions
   - 1.1 Create a table with the following conditions:
     - 1.1.1 Table name: movie
       Column family: info with 3 versions
       Column family: media with 3 versions
     - 1.1.2 Table name: ranking
       Column family: info with 3 versions
       Column family: versions with 3 versions
   - 1.2 Create a table with the following conditions:
     - 1.2.1 Table name: movie
       Column family: info with 3 versions
       Column family: media with 3 versions
     - 1.2.2 Table name: ranking
       Column family: info with 3 versions
       Column family: versions with 3 versions
   - 1.3 Show the number of rows in the movie and user tables you entered.
   - 1.4 Use alter command to change the movie table to add the title column family.
   - 1.5 Run the command to verify that the stitle column family has been added.
   - 1.6 Remove the added stitle.
   - 1.7 Check the movie table to ensure that the stitle column family has been removed.
   - 1.8 Modify the media column family in the movie table to keep 4 versions.
   - 1.9 Show the media change in the movie table.
2. Write a command for CRUD that meets the following conditions
   - 2.1 Show the user table to check the column family names.
   - 2.2 Get the row with row key 100 from the user table.
   - 2.3 Add a row with the following properties.
     - 2.3.1 Table name: user
     - 2.3.2 Row key: 100

- 2.3.3 Column Family: info
- 2.3.4 Column Descriptor: age and value 20
- 2.3.5 Column Descriptor: gender and value F
- 2.3.6 Column Descriptor: zip and value 18730

2.4 Just show the row with row key 100 from the user table.

2.5 In addition, insert a row with the following attributes:

- 2.5.1 Table name: user
- 2.5.2 Row key: 100
- 2.5.3 Column Family: info
- 2.5.4 Column Descriptor: age and value 30
- 2.5.5 Table name: user
- 2.5.6 Row key: 100
- 2.5.7 Column Family: info
- 2.5.8 Column Descriptor: age and value 40

2.6 Show the row with row key 100 from the user table.

2.7 In the user table, get all the old columns of the age column version in the row with row key 100.

2.8 Show the entire table with scan command, but only display the age column.

2.9 In the user table, delete the info:age column of the row with row key 100.

2.10 Verify that the age column has been deleted.

2.11 Delete the entire row with row key 100 from the user table.

2.12 Verify that the row with row key 100 has been removed from the user table.

Lab 6: Data Access with Cassandra

In this lab, you will start and use the Cassandra NoSQL for our labs.

1. Installing the Cassandra NoSQL
   check the latest version in https://cassandra.apache.org/_/download.html

   1.1 Create the Cassandra of repo file for installation.

```
sudo vi /etc/yum.repos.d/cassandra.repo
[cassandra]
name=Apache Cassandra
baseurl=https://downloads.apache.org/cassandra/redhat/40x/
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://downloads.apache.org/cassandra/KEYS
```

   1.2 Install the Cassandra NoSQL

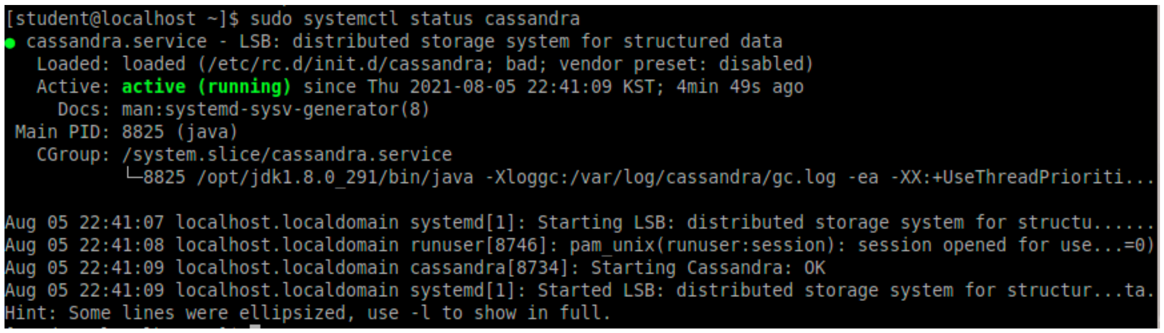
```
sudo yum install cassandra
```

   1.3 Start the Cassandra service

```
sudo systemctl daemon-reload
sudo systemctl start cassandra
```

   1.4 Verify the Cassandra service

```
sudo systemctl status cassandra
```



```
[student@localhost ~]$ sudo systemctl status cassandra
● cassandra.service - LSB: distributed storage system for structured data
   Loaded: loaded (/etc/rc.d/init.d/cassandra; bad; vendor preset: disabled)
   Active: active (running) since Thu 2021-08-05 22:41:09 KST; 4min 49s ago
     Docs: man:systemd-sysv-generator(8)
 Main PID: 8825 (java)
   CGroup: /system.slice/cassandra.service
           └─8825 /opt/jdk1.8.0_291/bin/java -Xloggc:/var/log/cassandra/gc.log -ea -XX:+UseThreadPrioriti...

Aug 05 22:41:07 localhost.localdomain systemd[1]: Starting LSB: distributed storage system for structu......
Aug 05 22:41:08 localhost.localdomain runuser[8746]: pam_unix(runuser:session): session opened for use...=0)
Aug 05 22:41:09 localhost.localdomain cassandra[8734]: Starting Cassandra: OK
Aug 05 22:41:09 localhost.localdomain systemd[1]: Started LSB: distributed storage system for structur...ta.
Hint: Some lines were ellipsized, use -l to show in full.
```

2. Starting the CQL shell

    2.1 Run the CQL shell

```
cqlsh
```

    2.2 List all keyspaces in the cluster

```
cqlsh> desc keyspaces;
```

    2.3 Create a keyspace called “ks_work1” on a single node.

```
cqlsh> CREATE KEYSPACE ks_work1
  WITH REPLICATION = {
   'class' : 'SimpleStrategy',
   'replication_factor' : 1
  };
```

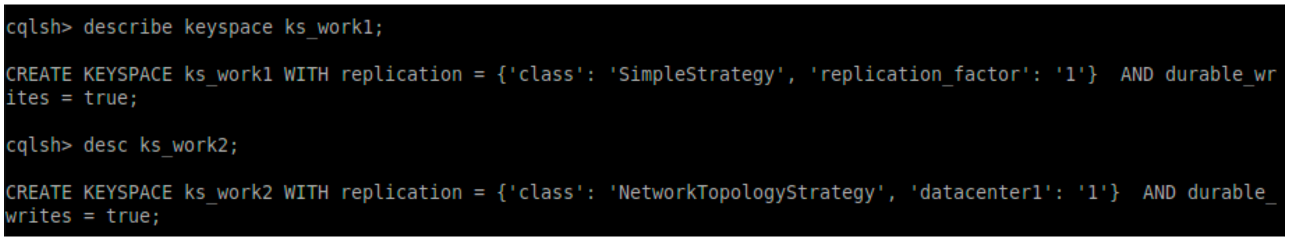
    2.4 If you want to use multiple data centers, Use NetworkTopologyStrategy on an evaluation cluster.

    2.5 This example shows how to create a keyspace with network topology in a single node evaluation cluster.

```
cqlsh> CREATE KEYSPACE ks_work2
  WITH REPLICATION = {
   'class' : 'NetworkTopologyStrategy',
   'datacenter1' : 1
  };
```

2.6 Verify the generated keyspace information.

```
cqlsh> describe keyspace ks_work1;
cqlsh> desc ks_work2;
```

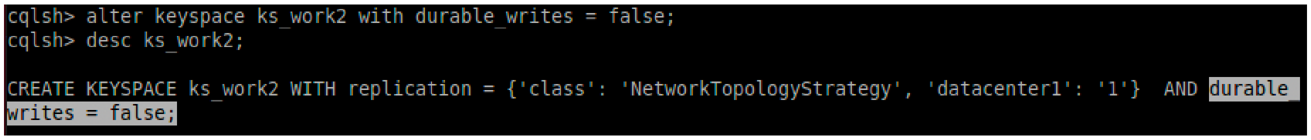


```
cqlsh> describe keyspace ks_work1;

CREATE KEYSPACE ks_work1 WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '1'}  AND durable_wr
ites = true;

cqlsh> desc ks_work2;

CREATE KEYSPACE ks_work2 WITH replication = {'class': 'NetworkTopologyStrategy', 'datacenter1': '1'}  AND durable_
writes = true;
```

2.7 Change and delete keyspace settings

```
cqlsh> alter keyspace ks_work2 with durable_writes = false;
cqlsh> desc ks_work2;
```

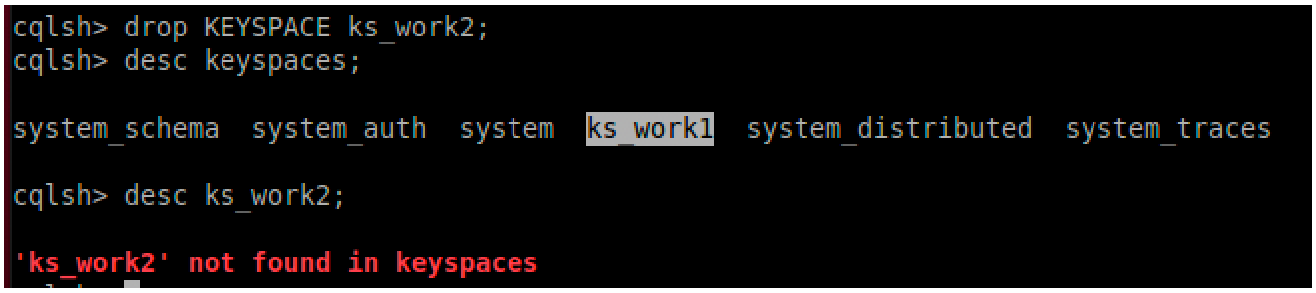
2.7.1 Verify the changed setting in ks_work2



```
cqlsh> alter keyspace ks_work2 with durable_writes = false;
cqlsh> desc ks_work2;

CREATE KEYSPACE ks_work2 WITH replication = {'class': 'NetworkTopologyStrategy', 'datacenter1': '1'}  AND durable_
writes = false;
```

```
cqlsh> drop KEYSPACE ks_work2;
cqlsh> desc ks_work2;
```

2.7.2 Verify the keyspace, ks_work2.



```
cqlsh> drop KEYSPACE ks_work2;
cqlsh> desc keyspaces;

system_schema  system_auth  system  ks_work1  system_distributed  system_traces

cqlsh> desc ks_work2;

'ks_work2' not found in keyspaces
```

3. Data Accessing using DDL

3.1 Create tables with the following characteristics (create table including key space without use command)

3.1.1 Keyspace: ks_work1

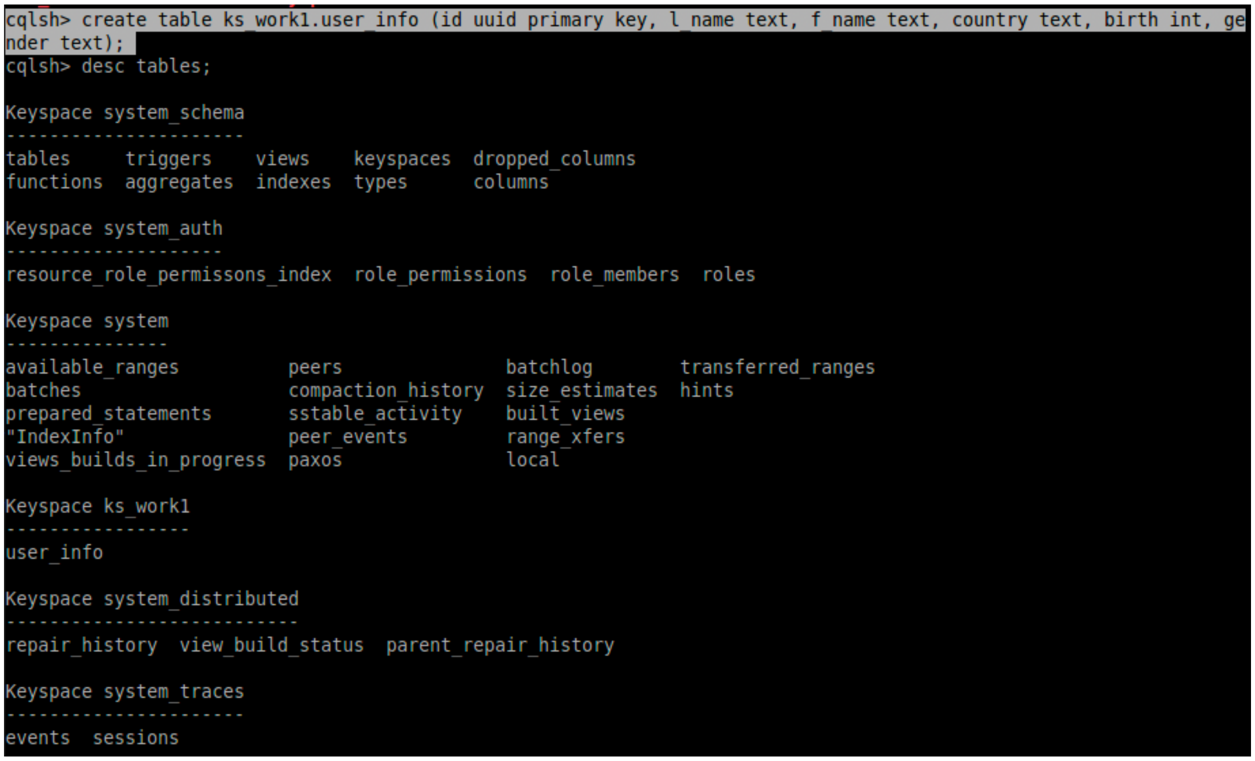
Table name: user_info

Column name: id uuid, l_name text, f_name text, country text, birth int, gender text

Partition Key: id

```
cqlsh> create table ks_work1.user_info (id uuid primary key, l_name text, f_name text, country text, birth int, gender text);
```

3.1.2 Verify the user_info table.



```
cqlsh> create table ks_work1.user_info (id uuid primary key, l_name text, f_name text, country text, birth int, gender text);
cqlsh> desc tables;

Keyspace system_schema
----------------------
tables     triggers    views    keyspaces  dropped_columns
functions  aggregates  indexes  types      columns

Keyspace system_auth
--------------------
resource_role_permissons_index  role_permissions  role_members  roles

Keyspace system
---------------
available_ranges          peers               batchlog        transferred_ranges
batches                   compaction_history  size_estimates  hints
prepared_statements       sstable_activity    built_views
"IndexInfo"               peer_events         range_xfers
views_builds_in_progress  paxos               local

Keyspace ks_work1
-----------------
user_info

Keyspace system_distributed
---------------------------
repair_history  view_build_status  parent_repair_history

Keyspace system_traces
----------------------
events  sessions
```

3.1.3 Keyspace: ks_work1

Table name: user_info2

Column name: id uuid, l_name text, f_name text, country text, birth int, gender text, s text, num int

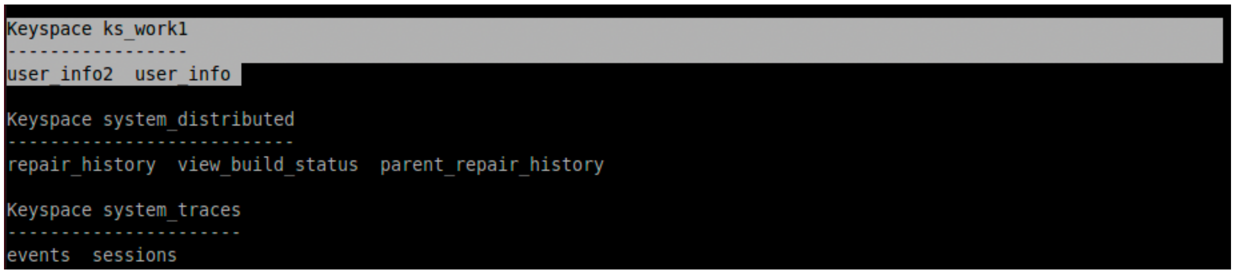
static column: s

primary key : country, birth, id

composite partition key : country, birth

```
cqlsh> create table ks_work1.user_info2 (id int, l_name text, f_name text, country text, birth text, gender text, s text, num int, primary key((country,birth), id));

csqsh> desc tables;
```

3.1.4 Verify the user_info2 table.



```
Keyspace ks_work1
-----------------
user_info2  user_info

Keyspace system_distributed
---------------------------
repair_history  view_build_status  parent_repair_history

Keyspace system_traces
----------------------
events  sessions
```

3.1.5 Keyspace: ks_work1

Table name: user_info3

Column name: id int, l_name text, f_name text, country text, birth int, age int

Compound Primary Key : country, id, birth

Clustering key: birth

```
cqlsh>  create table ks_work1.user_info3 (id int, l_name text, f_name text, country text, birth text, gender text, s text static, num int, primary key(country, id, birth));
```

3.1.6 Keyspace: ks_work1

Table name: user_info33

Column name: id int, l_name text, f_name text, country text, birth int, age int

Compound Primary Key : country, id, birth (descending order)

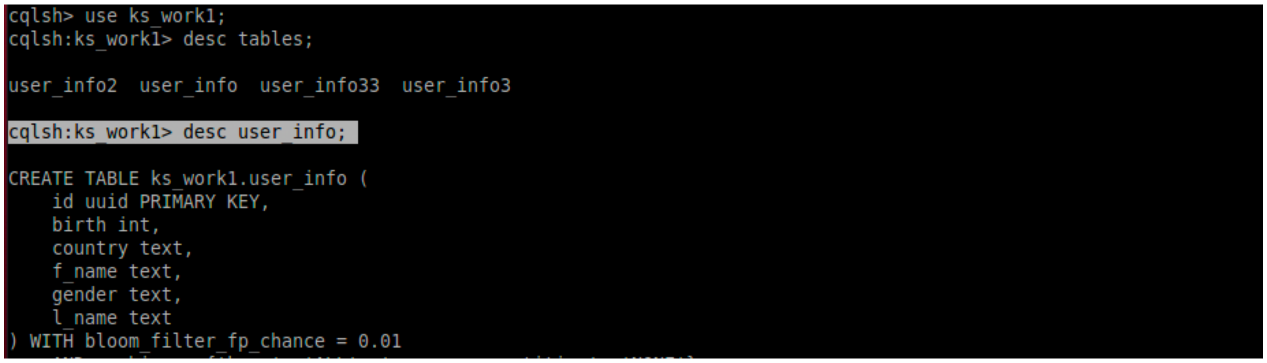
Clustering key: birth (store data in descending order)

```
cqlsh>  create table ks_work1.user_info33 (id int, l_name text, f_name text, country text, birth text, gender text, s text static, age int, primary key(id, birth)) with clustering order by (birth desc);

csqsh> desc tables;
```

3.2 Change the keyspace to ks_work1 and verify whether the tables user_info, user_info2 and user_info3 were created successfully.

```
cqlsh> use ks_work1;
cqlsh:ks_work1> desc tables;
cqlsh:ks_work1> desc user_info;
```

3.2.1 Verify the user_info

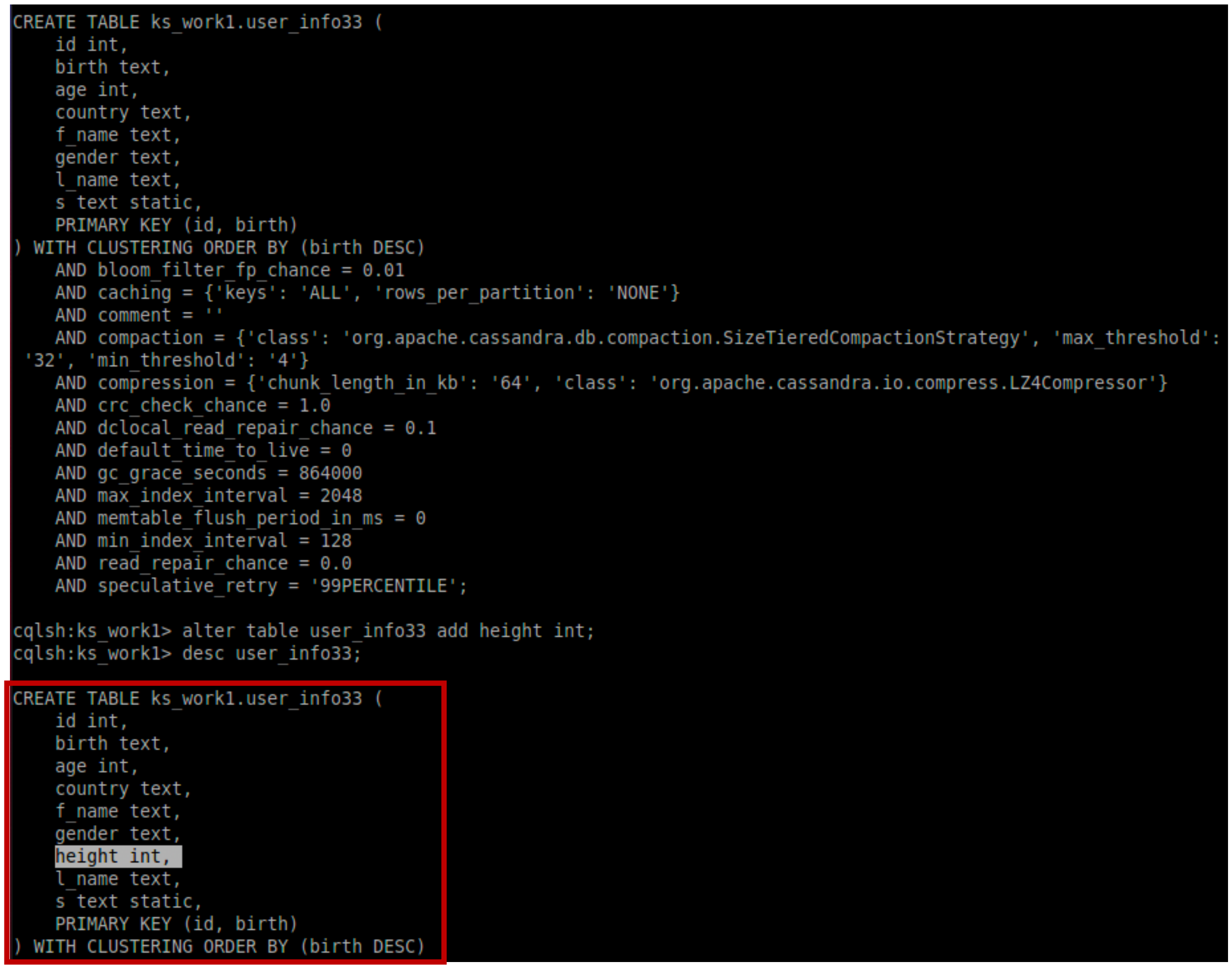


```
cqlsh> use ks_work1;
cqlsh:ks_work1> desc tables;

user_info2  user_info  user_info33  user_info3

cqlsh:ks_work1> desc user_info;

CREATE TABLE ks_work1.user_info (
    id uuid PRIMARY KEY,
    birth int,
    country text,
    f_name text,
    gender text,
    l_name text
) WITH bloom_filter_fp_chance = 0.01
```

3.3 Add the following column to the user_info33 table. And verify the user_info33 table.

3.3.1 Column name : height, type int

```
cqlsh:ks_work1> alter table user_info33 add height int;
cqlsh:ks_work1> desc user_info33;
```



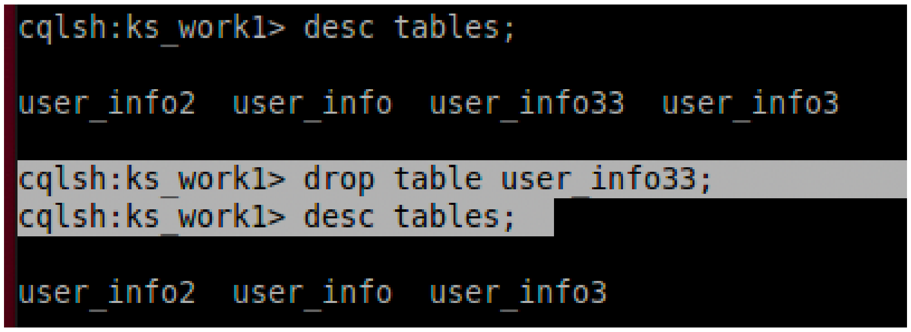
```
CREATE TABLE ks_work1.user_info33 (
    id int,
    birth text,
    age int,
    country text,
    f_name text,
    gender text,
    l_name text,
    s text static,
    PRIMARY KEY (id, birth)
) WITH CLUSTERING ORDER BY (birth DESC)
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold':
 '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';

cqlsh:ks_work1> alter table user_info33 add height int;
cqlsh:ks_work1> desc user_info33;

CREATE TABLE ks_work1.user_info33 (
    id int,
    birth text,
    age int,
    country text,
    f_name text,
    gender text,
    height int,
    l_name text,
    s text static,
    PRIMARY KEY (id, birth)
) WITH CLUSTERING ORDER BY (birth DESC)
```

3.4 Delete the user_info33 table you just modified.

```
cqlsh:ks_work1> drop table user_info33;
```

3.5 List all tables again to confirm that table user_inf33 was dropped successfully.

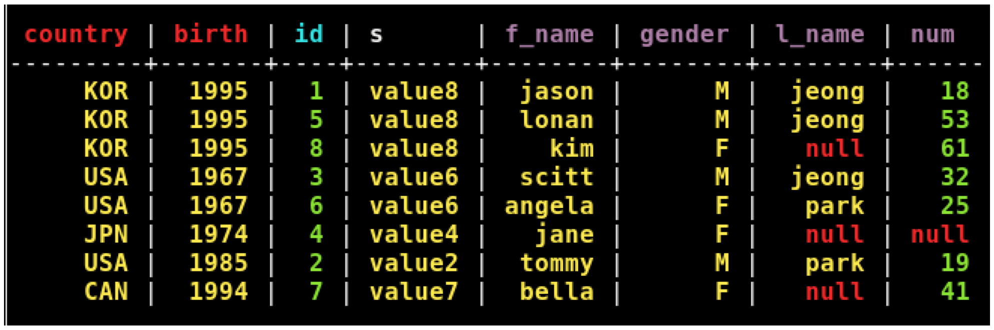
```
cqlsh:ks_work1> desc tables;
```



```
cqlsh:ks_work1> desc tables;

user_info2  user_info  user_info33  user_info3

cqlsh:ks_work1> drop table user_info33;
cqlsh:ks_work1> desc tables;

user_info2  user_info  user_info3
```

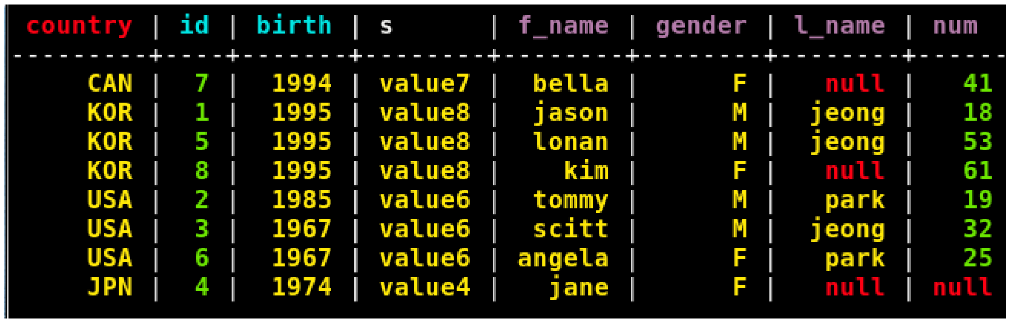
3.6 Enter data into the user_info2 and user_info3 tables, respectively, as shown in the following figure.

3.6.1 User_info2



```
 country | birth | id | s      | f_name | gender | l_name | num
---------+-------+----+--------+--------+--------+--------+------
     KOR |  1995 |  1 | value8 |  jason |      M |  jeong |   18
     KOR |  1995 |  5 | value8 |  lonan |      M |  jeong |   53
     KOR |  1995 |  8 | value8 |    kim |      F |   null |   61
     USA |  1967 |  3 | value6 |  scitt |      M |  jeong |   32
     USA |  1967 |  6 | value6 | angela |      F |   park |   25
     JPN |  1974 |  4 | value4 |   jane |      F |   null | null
     USA |  1985 |  2 | value2 |  tommy |      M |   park |   19
     CAN |  1994 |  7 | value7 |  bella |      F |   null |   41
```

3.6.2 User_info3



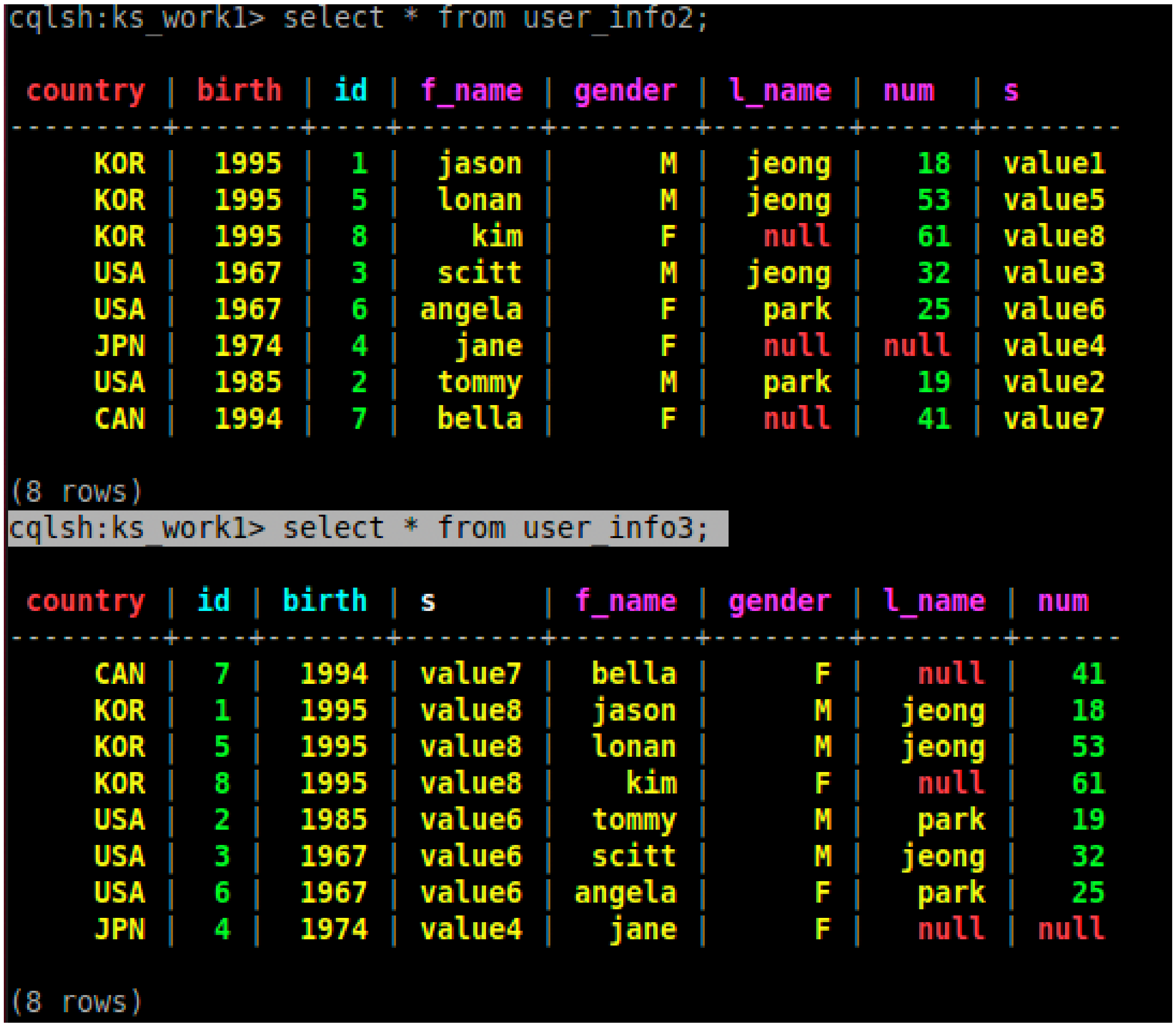
```
 country | id | birth | s      | f_name | gender | l_name | num
---------+----+-------+--------+--------+--------+--------+------
     CAN |  7 |  1994 | value7 |  bella |      F |   null |   41
     KOR |  1 |  1995 | value8 |  jason |      M |  jeong |   18
     KOR |  5 |  1995 | value8 |  lonan |      M |  jeong |   53
     KOR |  8 |  1995 | value8 |    kim |      F |   null |   61
     USA |  2 |  1985 | value6 |  tommy |      M |   park |   19
     USA |  3 |  1967 | value6 |  scitt |      M |  jeong |   32
     USA |  6 |  1967 | value6 | angela |      F |   park |   25
     JPN |  4 |  1974 | value4 |   jane |      F |   null | null
```

4. Data Accessing using DML

4.1 Select statement that outputs Info2, info3 data and limit to 1 row per partition.

```
cqlsh:ks_work1> select * from user_info2;
cqlsh:ks_work1> select * from user_info3;
```

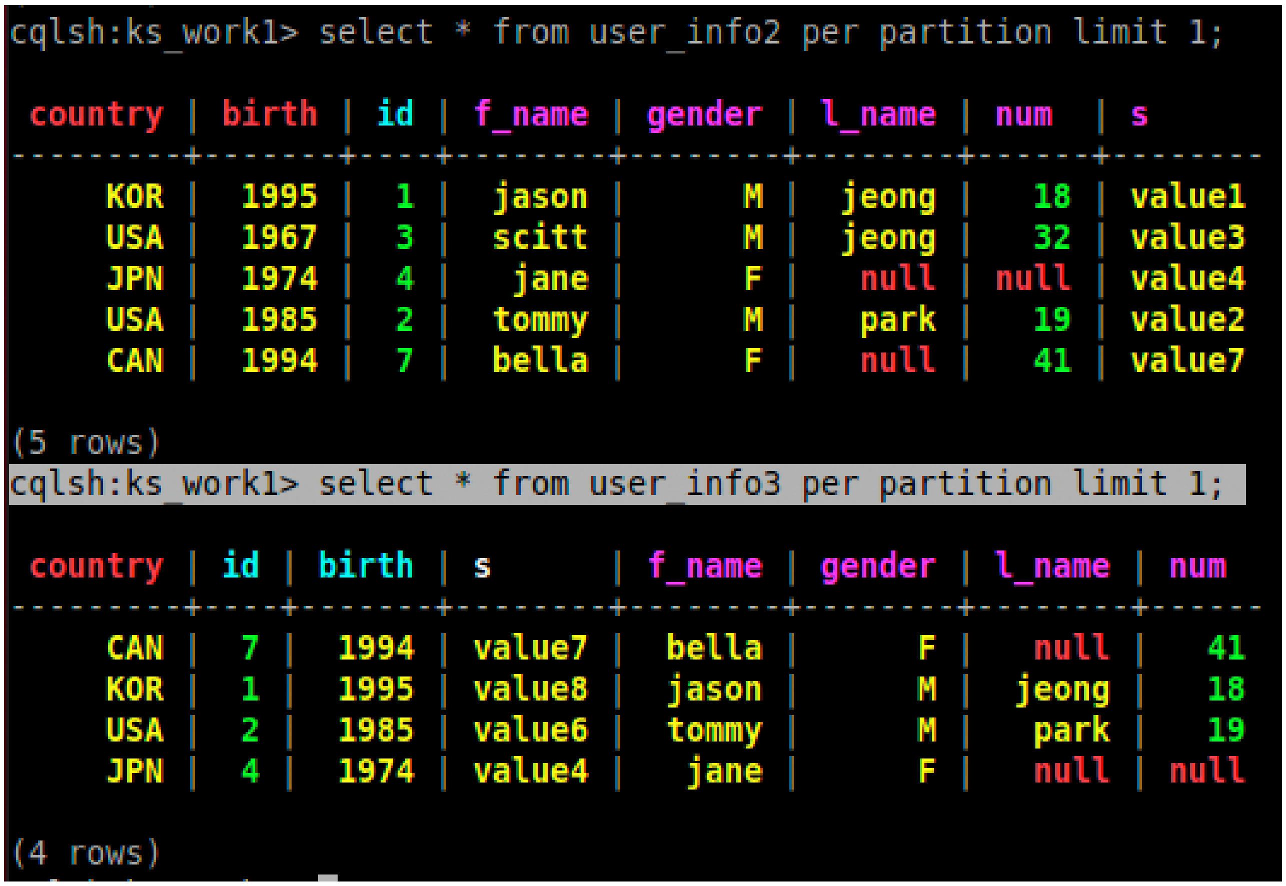
4.1.1 Verify the records for user_info2 and user_info3;



```
cqlsh:ks_work1> select * from user_info2;

 country | birth | id | f_name | gender | l_name | num  | s
---------+-------+----+--------+--------+--------+------+--------
     KOR |  1995 |  1 |  jason |      M |  jeong |   18 | value1
     KOR |  1995 |  5 |  lonan |      M |  jeong |   53 | value5
     KOR |  1995 |  8 |    kim |      F |   null |   61 | value8
     USA |  1967 |  3 |  scitt |      M |  jeong |   32 | value3
     USA |  1967 |  6 | angela |      F |   park |   25 | value6
     JPN |  1974 |  4 |   jane |      F |   null | null | value4
     USA |  1985 |  2 |  tommy |      M |   park |   19 | value2
     CAN |  1994 |  7 |  bella |      F |   null |   41 | value7

(8 rows)
cqlsh:ks_work1> select * from user_info3;

 country | id | birth | s      | f_name | gender | l_name | num
---------+----+-------+--------+--------+--------+--------+------
     CAN |  7 |  1994 | value7 |  bella |      F |   null |   41
     KOR |  1 |  1995 | value8 |  jason |      M |  jeong |   18
     KOR |  5 |  1995 | value8 |  lonan |      M |  jeong |   53
     KOR |  8 |  1995 | value8 |    kim |      F |   null |   61
     USA |  2 |  1985 | value6 |  tommy |      M |   park |   19
     USA |  3 |  1967 | value6 |  scitt |      M |  jeong |   32
     USA |  6 |  1967 | value6 | angela |      F |   park |   25
     JPN |  4 |  1974 | value4 |   jane |      F |   null | null

(8 rows)
```

```
cqlsh:ks_work1> select * from user_info2 per partition limit 1;
cqlsh:ks_work1> select * from user_info3 per partition limit 1;
```

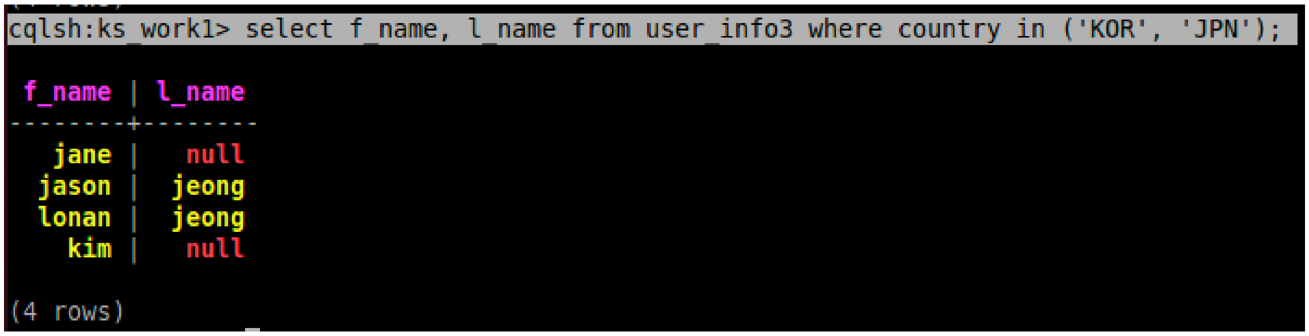
4.1.2 Verify the records for user_info2 and user_info3 with 1 row per partition.



```
cqlsh:ks_work1> select * from user_info2 per partition limit 1;

 country | birth | id | f_name | gender | l_name | num  | s
---------+-------+----+--------+--------+--------+------+--------
     KOR |  1995 |  1 |  jason |      M |  jeong |   18 | value1
     USA |  1967 |  3 |  scitt |      M |  jeong |   32 | value3
     JPN |  1974 |  4 |   jane |      F |   null | null | value4
     USA |  1985 |  2 |  tommy |      M |   park |   19 | value2
     CAN |  1994 |  7 |  bella |      F |   null |   41 | value7

(5 rows)
cqlsh:ks_work1> select * from user_info3 per partition limit 1;

 country | id | birth | s      | f_name | gender | l_name | num
---------+----+-------+--------+--------+--------+--------+------
     CAN |  7 |  1994 | value7 |  bella |      F |   null |   41
     KOR |  1 |  1995 | value8 |  jason |      M |  jeong |   18
     USA |  2 |  1985 | value6 |  tommy |      M |   park |   19
     JPN |  4 |  1974 | value4 |   jane |      F |   null | null

(4 rows)
```

4.2 List of persons included in KOR/JPN.

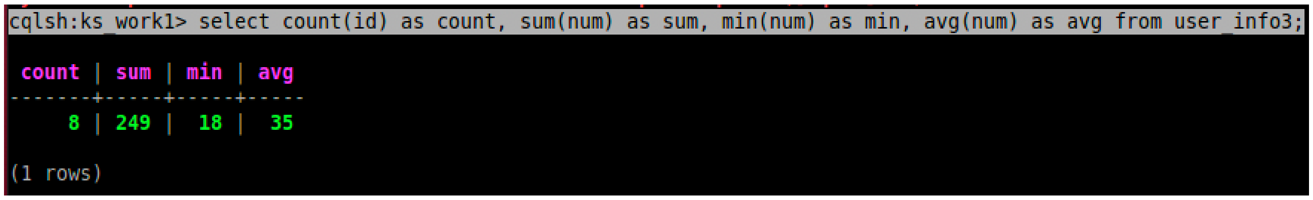
```
cqlsh:ks_work1> select f_name, l_name from user_info3 where country in ('KOR',
'JPN');
```



```
cqlsh:ks_work1> select f_name, l_name from user_info3 where country in ('KOR', 'JPN');

 f_name | l_name
--------+--------
   jane |   null
  jason |  jeong
  lonan |  jeong
    kim |   null

(4 rows)
```

4.3 Find the total number of records in the user_info3 table and the sum, minimum, and average values of num.

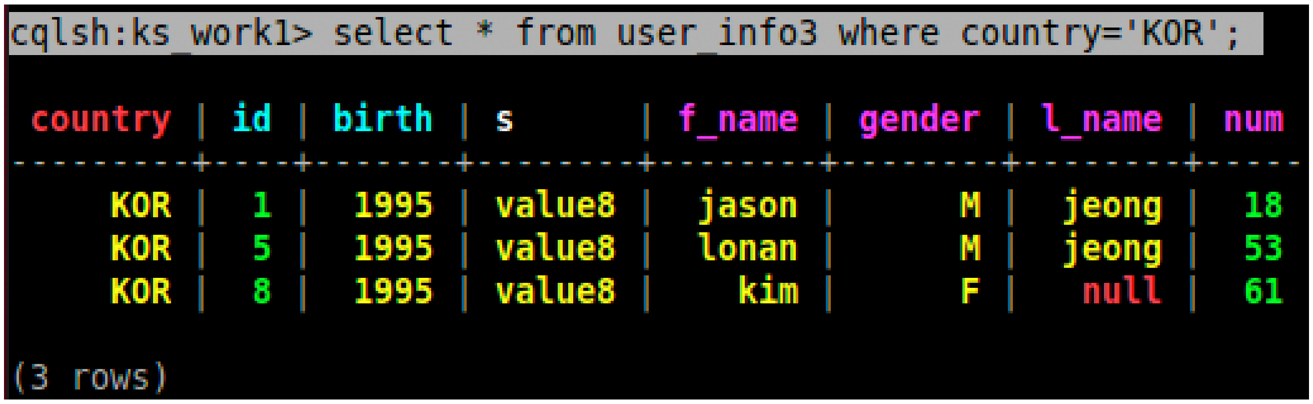
```
cqlsh:ks_work1> select count(id) as count, sum(num) as sum, min(num) as min,
avg(num) as avg from user_info3;
```



```
cqlsh:ks_work1> select count(id) as count, sum(num) as sum, min(num) as min, avg(num) as avg from user_info3;

 count | sum | min | avg
-------+-----+-----+-----
     8 | 249 |  18 |  35

(1 rows)
```

4.4 Get a record of the country being KOR.

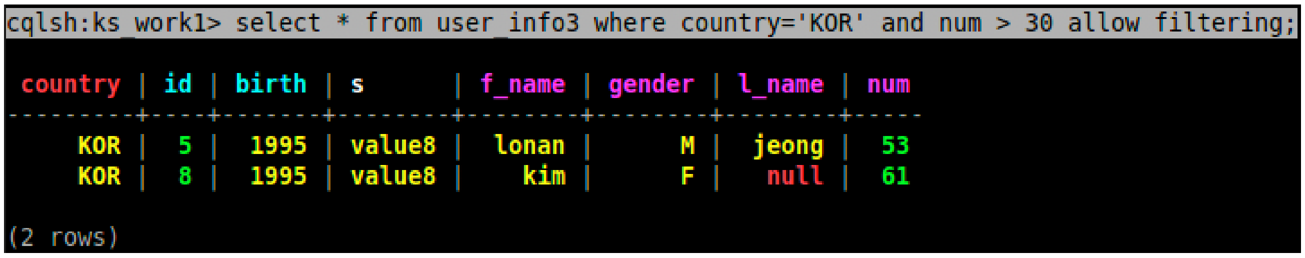
```
cqlsh:ks_work1> select * from user_info3 where country='KOR';
```



```
cqlsh:ks_work1> select * from user_info3 where country='KOR';

 country | id | birth | s      | f_name | gender | l_name | num
---------+----+-------+--------+--------+--------+--------+-----
     KOR |  1 |  1995 | value8 |  jason |      M |  jeong |  18
     KOR |  5 |  1995 | value8 |  lonan |      M |  jeong |  53
     KOR |  8 |  1995 | value8 |    kim |      F |   null |  61

(3 rows)
```

4.5 Find records where country Is KOR and number is greater than 30

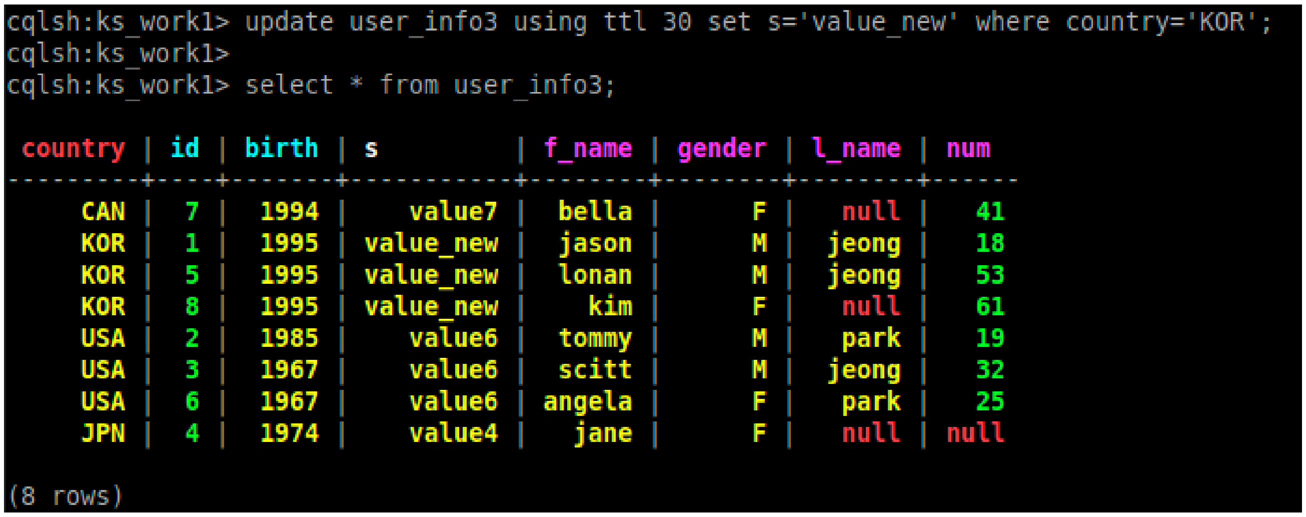
```
cqlsh:ks_work1> select * from user_info3 where country='KOR' and num > 30 allow
filtering;
```



```
cqlsh:ks_work1> select * from user_info3 where country='KOR' and num > 30 allow filtering;

 country | id | birth | s      | f_name | gender | l_name | num
---------+----+-------+--------+--------+--------+--------+-----
     KOR |  5 |  1995 | value8 |  lonan |      M |  jeong |  53
     KOR |  8 |  1995 | value8 |    kim |      F |   null |  61

(2 rows)
```

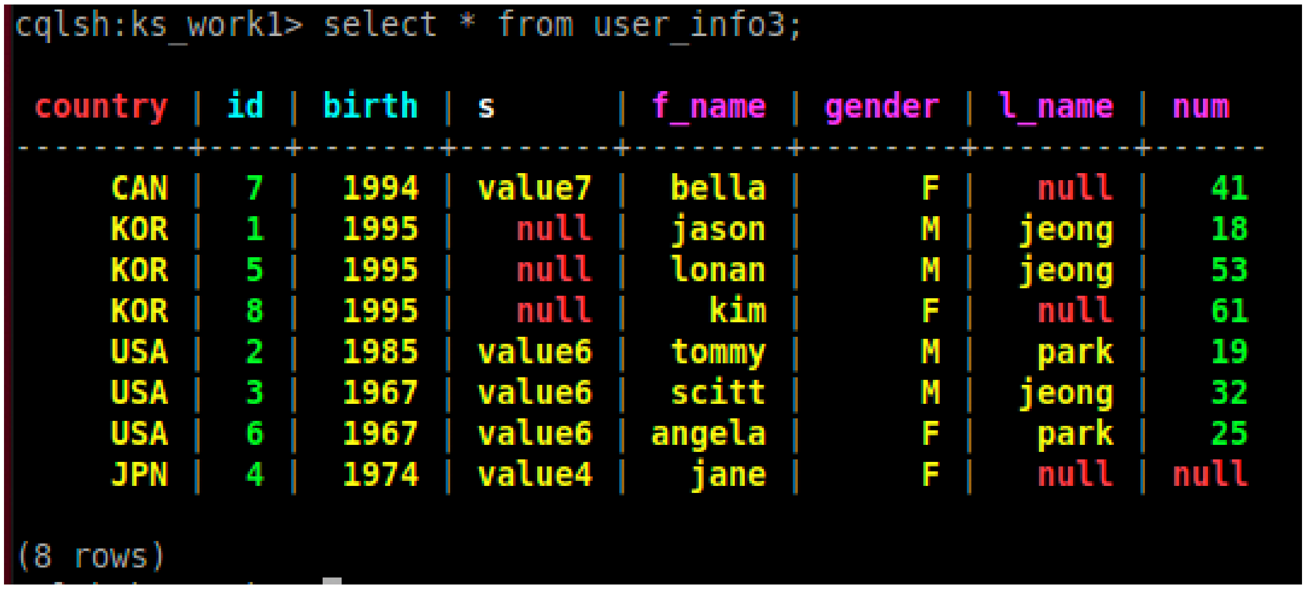
4.6 Modify the value of s in KOR to ‘value8’, and make it null after 30 seconds.

```
cqlsh:ks_work1> update user_info3 using ttl 30 set s='value_new' where
country='KOR';
```



```
cqlsh:ks_work1> update user_info3 using ttl 30 set s='value_new' where country='KOR';
cqlsh:ks_work1>
cqlsh:ks_work1> select * from user_info3;

 country | id | birth | s         | f_name | gender | l_name | num
---------+----+-------+-----------+--------+--------+--------+------
     CAN |  7 |  1994 |    value7 |  bella |      F |   null |   41
     KOR |  1 |  1995 | value_new |  jason |      M |  jeong |   18
     KOR |  5 |  1995 | value_new |  lonan |      M |  jeong |   53
     KOR |  8 |  1995 | value_new |    kim |      F |   null |   61
     USA |  2 |  1985 |    value6 |  tommy |      M |   park |   19
     USA |  3 |  1967 |    value6 |  scitt |      M |  jeong |   32
     USA |  6 |  1967 |    value6 | angela |      F |   park |   25
     JPN |  4 |  1974 |    value4 |   jane |      F |   null | null

(8 rows)
```

4.6.1 After 30 seconds, the values in 3 rows are deleted and become null.



```
cqlsh:ks_work1> select * from user_info3;

 country | id | birth | s      | f_name | gender | l_name | num
---------+----+-------+--------+--------+--------+--------+------
     CAN |  7 |  1994 | value7 |  bella |      F |   null |   41
     KOR |  1 |  1995 |   null |  jason |      M |  jeong |   18
     KOR |  5 |  1995 |   null |  lonan |      M |  jeong |   53
     KOR |  8 |  1995 |   null |    kim |      F |   null |   61
     USA |  2 |  1985 | value6 |  tommy |      M |   park |   19
     USA |  3 |  1967 | value6 |  scitt |      M |  jeong |   32
     USA |  6 |  1967 | value6 | angela |      F |   park |   25
     JPN |  4 |  1974 | value4 |   jane |      F |   null | null

(8 rows)
```

4.7 Insert the value of s in KOR to 'value8 and update ' and update the value of the row with id 4 with the following condition.

f_name:ann, l_name:kim, num:85

```
cqlsh:ks_work1> insert into user_info3 (id, f_name, l_name, num, country, birth)
values(4,'ann','kim', 85, 'JPN', '1974');
```

Note: insert into user_info3 (id, f_name, l_name, num) values(4,'ann','kim',85);

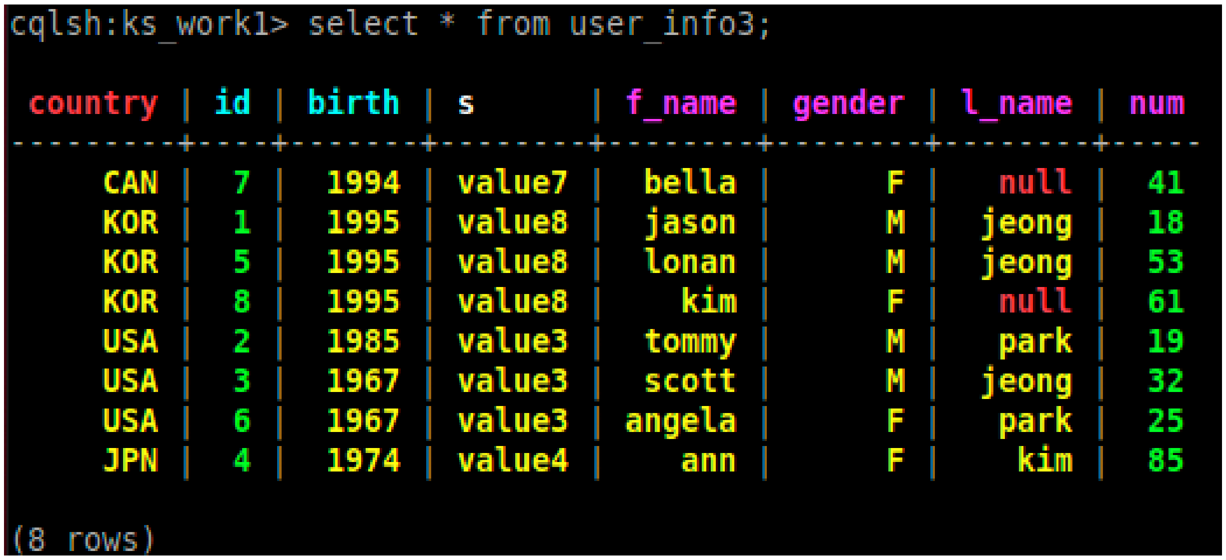
insert into user_info3 (id, f_name, l_name, num, country) values(4,'kim',85, 'JPN');

Execute the above 2 commands and consider the reason for the error.

4.8 Change the value of f_name in the record with id 3 to "scott".

```
cqlsh:ks_work1> insert into user_info3 (id, l_name, f_name, country, birth, gender, s,
num) values (3,'jeong','scott','USA','1967', 'M', 'value3', 32) if not exists;
```

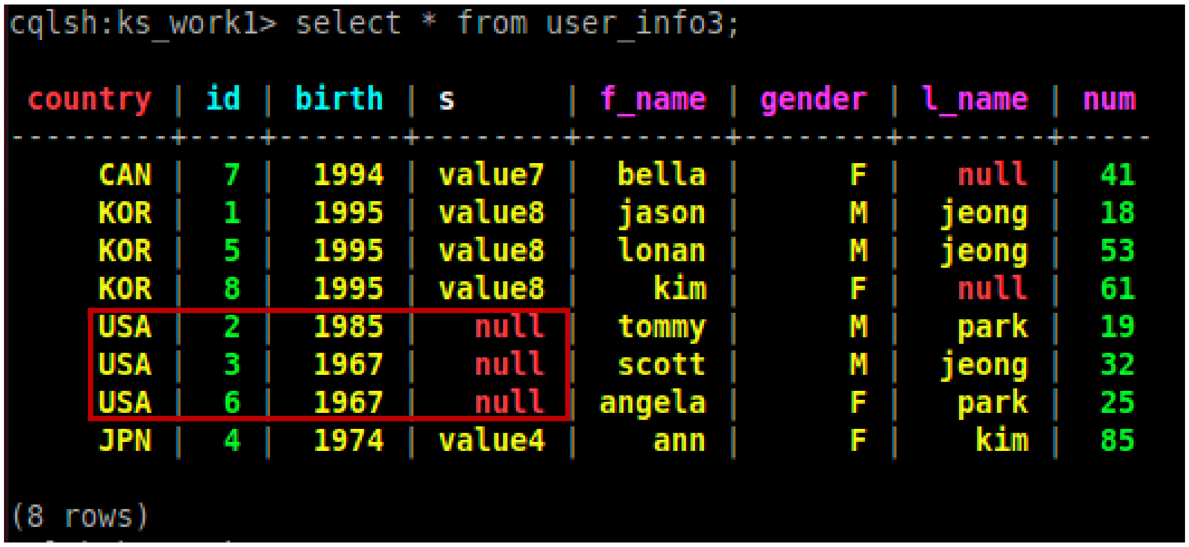
4.9 Change Scitt to scott in the 3rd row of the user_info3 table.



```
cqlsh:ks_work1> select * from user_info3;

 country | id | birth | s      | f_name | gender | l_name | num
---------+----+-------+--------+--------+--------+--------+-----
     CAN |  7 |  1994 | value7 |  bella |      F |   null |  41
     KOR |  1 |  1995 | value8 |  jason |      M |  jeong |  18
     KOR |  5 |  1995 | value8 |  lonan |      M |  jeong |  53
     KOR |  8 |  1995 | value8 |    kim |      F |   null |  61
     USA |  2 |  1985 | value3 |  tommy |      M |   park |  19
     USA |  3 |  1967 | value3 |  scott |      M |  jeong |  32
     USA |  6 |  1967 | value3 | angela |      F |   park |  25
     JPN |  4 |  1974 | value4 |    ann |      F |    kim |  85

(8 rows)
```

4.10 Delete the s column where the country is USA from the user_info3 table.

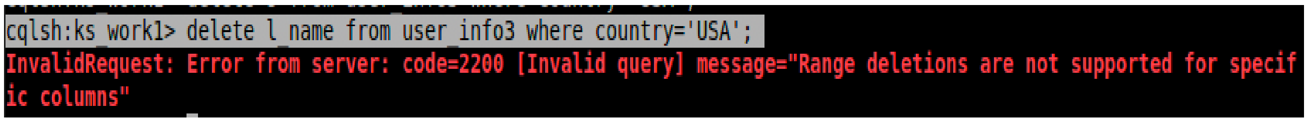
```
cqlsh:ks_work1> delete s from user_info3 where country='USA';
cqlsh:ks_work1> select * from user_info3;
```



```
cqlsh:ks_work1> select * from user_info3;

 country | id | birth | s      | f_name | gender | l_name | num
---------+----+-------+--------+--------+--------+--------+-----
     CAN |  7 |  1994 | value7 |  bella |      F |   null |  41
     KOR |  1 |  1995 | value8 |  jason |      M |  jeong |  18
     KOR |  5 |  1995 | value8 |  lonan |      M |  jeong |  53
     KOR |  8 |  1995 | value8 |    kim |      F |   null |  61
     USA |  2 |  1985 |   null |  tommy |      M |   park |  19
     USA |  3 |  1967 |   null |  scott |      M |  jeong |  32
     USA |  6 |  1967 |   null | angela |      F |   park |  25
     JPN |  4 |  1974 | value4 |    ann |      F |    kim |  85

(8 rows)
```

4.11 Delete the l_name column under the same conditions with 4.10

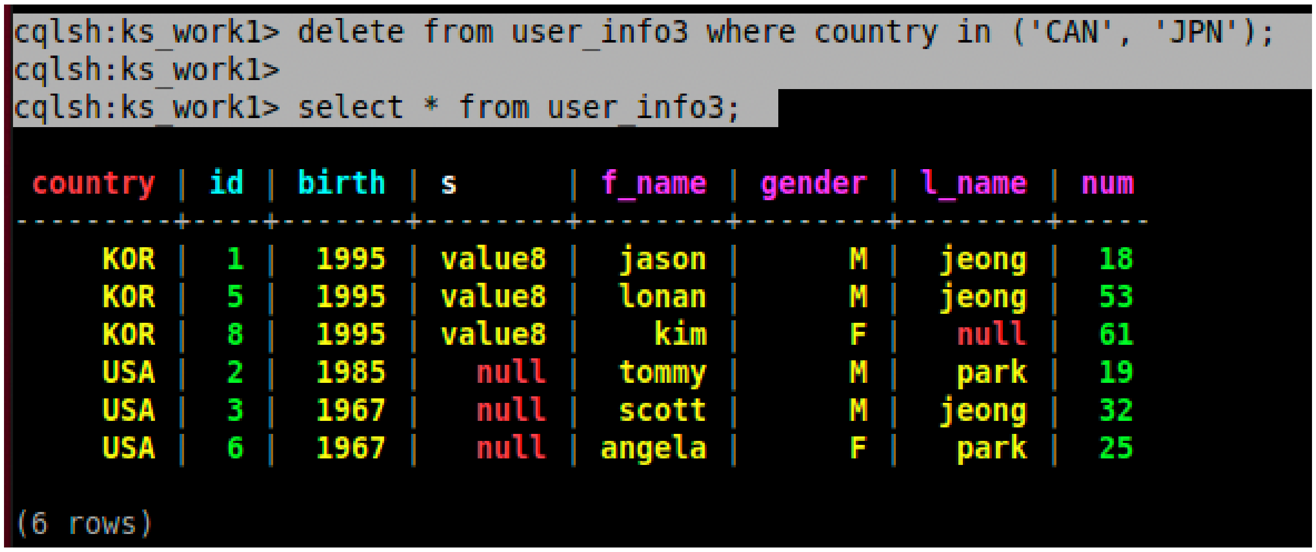
```
cqlsh:ks_work1> delete l_name from user_info3 where country='USA';
```



```
cqlsh:ks_work1> delete l_name from user_info3 where country='USA';
InvalidRequest: Error from server: code=2200 [Invalid query] message="Range deletions are not supported for specif
ic columns"
```

4.12 Delete the row where the country is CAN or JPN from the user_info3 table.››

```
cqlsh:ks_work1> delete from user_info3 where country in ('CAN', 'JPN');
cqlsh:ks_work1> select * from user_info3;
```



```
cqlsh:ks_work1> delete from user_info3 where country in ('CAN', 'JPN');
cqlsh:ks_work1>
cqlsh:ks_work1> select * from user_info3;

 country | id | birth | s      | f_name | gender | l_name | num
---------+----+-------+--------+--------+--------+--------+-----
     KOR |  1 |  1995 | value8 |  jason |      M |  jeong |  18
     KOR |  5 |  1995 | value8 |  lonan |      M |  jeong |  53
     KOR |  8 |  1995 | value8 |    kim |      F |   null |  61
     USA |  2 |  1985 |   null |  tommy |      M |   park |  19
     USA |  3 |  1967 |   null |  scott |      M |  jeong |  32
     USA |  6 |  1967 |   null | angela |      F |   park |  25

(6 rows)
```

Lab 7: Working with Cassandra

1. Write a command for Table creation for “country”

   1.1 Create a table with the following conditions:

Column Name	Column Type	Key
id	uuid	Primary Key
name	text	Partition Key
capital	text	Partition Key
population	int	

   1.2 Input the following values in the created country.

id	Name	Capital	Population
1	USA	Washington D.C.	333,098,437
2	Germany	Berlin	84,073,352
3	France	Paris	65,426,179
4	Korea	Seoul	51,305,186

2. Working with Queries.

   2.1 Query the **country** table for the following information:

      2.1.1 Population of Korea

      2.1.2 Population and capital of all entries in the **country** table

      2.1.3 Names of countries in alphabetical order

      2.1.4 Country, capital and population, ordered by population, descending

      2.1.5 Same as above but ordered ascending

      2.1.6 Countries with a population less than 100,000,000.

      2.1.7 Countries with a population between 50 and 100 million

   2.2 Create queries on the **country** table

      2.2.1 Name the country with the smallest population

      2.2.2 Name the country with the second largest population

      2.2.3 List the first row in the countries table

      2.2.4 How many countries are listed in the **country** table?

END OF LAB