* Manage keys and encrypted data using Cloud Key Management Storage (KMS)
* Create KeyRings and CryptoKeys
* Set a default encryption key for a storage bucket
* Encrypt an object with a Cloud KMS key
* Rotate encryption keys
* Manually perform server-side encryption with KMS keys

**Task 1: Configure required resources**

Create a Cloud Storage bucket

A bucket must have a globally unique name. For this lab you will use your GCP project ID as part of the bucket name to help ensure it will be unique. Your GCP project ID is automatically stored in a Cloudshell environment variable named DEVSHELL\_PROJECT\_ID .

1. On the Google Cloud Platform menu, click **Activate Google Cloud Shell** ( )to open Cloud Shell. If prompted, click **Start Cloud Shell**.
2. Run the following command to create the bucket:

gsutil mb -l us gs://$DEVSHELL\_PROJECT\_ID-kms

1. Run the following commands to create a few sample files that will be uploaded to the bucket:

echo "This is sample file 1" > file1.txt

echo "This is sample file 2" > file2.txt

echo "This is sample file 3" > file3.txt

1. Run the following command to copy **file1.txt** to the bucket:

gsutil cp file1.txt gs://$DEVSHELL\_PROJECT\_ID-kms

Enable Cloud KMS

Before using KMS you need to enable it in your project. In the Qwiklab GCP Project you have been provisioned, KMS should already have been enabled. Just to be safe, you will issue the command to enable it anyway.

Run the following command in Cloud shell to enable Cloud KMS:

gcloud services enable cloudkms.googleapis.com

Click *Check my progress* to verify the objective.

Create a Cloud Storage bucket

Check my progress

**Task 2. Using Cloud KMS**

Create a Keyring and Cryptokey

In order to encrypt data, you need to create a KeyRing and a CryptoKey. KeyRings are useful for grouping keys. Keys can be grouped by environment (like test, staging, and prod) or by some other conceptual grouping. For this lab, your KeyRing will be called **test** and your CryptoKey will be called **labkey**.

1. In Cloud Shell, run the following commands to create variables to hold the KeyRing name and CryptoKey name:

KEYRING\_NAME=lab-keyring

CRYPTOKEY\_1\_NAME=labkey-1

CRYPTOKEY\_2\_NAME=labkey-2

1. Execute the following command to create the KeyRing. For this lab, the location will be set to us, but it could also be a specific region.

gcloud kms keyrings create $KEYRING\_NAME --location us

1. Next, using the new KeyRing, create a CryptoKey named **labkey-1**.

gcloud kms keys create $CRYPTOKEY\_1\_NAME --location us \

--keyring $KEYRING\_NAME --purpose encryption

1. Create another CryptoKey named **labkey-2**.

gcloud kms keys create $CRYPTOKEY\_2\_NAME --location us \

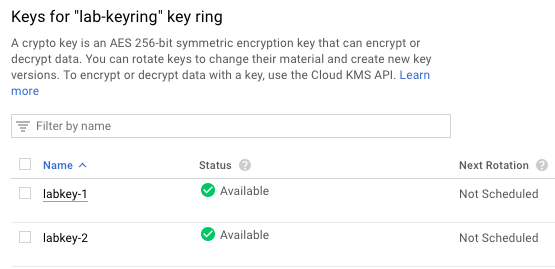
--keyring $KEYRING\_NAME --purpose encryption

You can view the KeyRing and keys in the GCP console.

1. In the GCP console, go to **Navigation menu** > **Security** > **Cryptographic keys**.

You will see the KeyRing named **lab-keyring**.

1. Click on the KeyRing named **lab-keyring** to view the encryption keys named **labkey-1** and **labkey-2**.



Click *Check my progress* to verify the objective.

Create a Keyring and Cryptokey

Check my progress

**Task 3: Adding a default key for a bucket**

Viewing the current default key for a bucket

Run the following command to view the default encryption key for your bucket:

gsutil kms encryption gs://$DEVSHELL\_PROJECT\_ID-kms

The bucket should not currently have a default encryption key. This means all data in the bucket will be encrypted by Google managed encryption keys.

Assigning Cloud KMS keys to a service account

Run the following commands to give your Cloud Storage service account permission to use both of your Cloud KMS keys:

gsutil kms authorize -p $DEVSHELL\_PROJECT\_ID -k \

projects/$DEVSHELL\_PROJECT\_ID/locations/us/keyRings\

/$KEYRING\_NAME/cryptoKeys/$CRYPTOKEY\_1\_NAME

gsutil kms authorize -p $DEVSHELL\_PROJECT\_ID -k \

projects/$DEVSHELL\_PROJECT\_ID/locations/us/keyRings\

/$KEYRING\_NAME/cryptoKeys/$CRYPTOKEY\_2\_NAME

The general syntax of the above commands is given below:

gsutil kms authorize -p [PROJECT\_STORING\_OBJECTS] -k [KEY\_RESOURCE]

Where [KEY\_RESOURCE] is in the format:

projects/[PROJECT\_STORING\_KEYS]/locations/[LOCATION]/keyRings/[KEY\_RING\_NAME]/cryptoKeys/[KEY\_NAME]

Setting the default key for a bucket

A Cloud KMS key can be set as the default key when objects are written to a bucket. When setting the default key, the key resource must be specified in the same format as the previous command: projects/[PROJECT\_STORING\_KEYS]/locations/[LOCATION]/keyRings/ [KEY\_RING\_NAME]/cryptoKeys/[KEY\_NAME]

1. Run the following command to set the default key for your bucket to the first key you generated:

gsutil kms encryption -k \

projects/$DEVSHELL\_PROJECT\_ID/locations/us/keyRings\

/$KEYRING\_NAME/cryptoKeys/$CRYPTOKEY\_1\_NAME \

gs://$DEVSHELL\_PROJECT\_ID-kms

1. Run the following command to view the default key for the bucket to verify the last command was successful:

gsutil kms encryption gs://$DEVSHELL\_PROJECT\_ID-kms

The default encryption key for the bucket should now be your first encryption key.

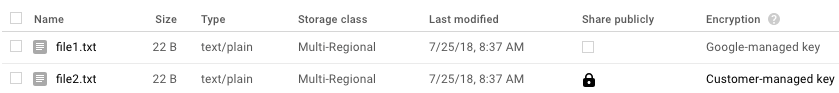
**Important**: Objects that were written to a bucket prior to adding or changing the default key remain encrypted with the previous encryption method.

1. Run the following command to copy **file2.txt** to the bucket:

gsutil cp file2.txt gs://$DEVSHELL\_PROJECT\_ID-kms

Viewing the files in the bucket

1. In the GCP console, go to **Navigation menu** > **Storage** > **Browser** and click on your bucket for this lab.
2. You will see **file 1** was encrypted with a Google-managed key and **file 2** was encrypted with a Customer-managed key.



Click *Check my progress* to verify the objective.

Add a default key for the bucket

Check my progress

**Task 4: Encrypting individual objects with a Cloud KMS key**

Encrypting an object with a specific key

You can also encrypt an individual object with a Cloud KMS key. This is useful if you want to use a different key from the default key set on the bucket, or if you don't have a default key set on the bucket. This can be done by passing the key to use in each gsutil command by using the -o flag: -o "GSUtil:encryption\_key=[KEY\_RESOURCE]"

1. Run the following command to copy **file3.txt** to the bucket, encrypting it with your second encryption key:

gsutil -o \

"GSUtil:encryption\_key=projects/$DEVSHELL\_PROJECT\_ID/locations/us/keyRings\

/$KEYRING\_NAME/cryptoKeys/$CRYPTOKEY\_2\_NAME" \

cp file3.txt gs://$DEVSHELL\_PROJECT\_ID-kms

1. In the GCP console, refresh the **Bucket details** screen and you will see **file3.txt** is also encrypted with a Customer-managed key.

Identifying the key used to encrypt an object

Notice that the GCP console does not indicate which key was used to encrypt the files. Currently it is not possible to identify the key used in the console. This can be done with gsutil.

1. Run the following command to display details about an object (the **-L** option causes **gsutil ls** to display all file details):

gsutil ls -L gs://$DEVSHELL\_PROJECT\_ID-kms/file3.txt

1. In the information returned, locate the KMS key: line. This displays the encryption key being used by that file.
2. Run the previous command again for **file1.txt** and **file2.txt**.

Click *Check my progress* to verify the objective.

Encrypt individual objects with a Cloud KMS key

Check my progress

**Task 5: Key Rotation**

In Cloud KMS, a key rotation is triggered by generating a new version of a key, and marking that version as the primary version. Each key has a designated primary version at any point in time, which Cloud KMS uses to encrypt data. After rotating a key, its previous key versions (which no longer are primary) are neither disabled or destroyed, and remain available for decrypting data.

Automatically rotating keys

By providing a rotation schedule, Cloud KMS will automatically rotate your keys for you. A key's rotation schedule can be set using the gcloud command-line tool or via the Google Cloud Platform Console.

1. In the GCP console, go to **Navigation menu** > **Security** > **Cryptographic keys** and click on the KeyRing named **lab-keyring** to view your encryption keys named **labkey-1** and **labkey-2**.
2. Click on the key named **labkey-1** to view all versions. Currently you only have one version.
3. Click the **EDIT ROTATION PERIOD** button.
4. Set the **rotation period** drop down to **30 days**. Notice that the rotation period can also be set to a Custom period that allows you to specify any desired period.
5. Click **SAVE**.

The console now displays the next rotation date for this key.

Manually rotating keys

Manually rotating keys can also be done with the gcloud command-line tool or via the Google Cloud Platform Console.

1. In the GCP console, go back to the KeyRing named **lab-keyring** and click on the key named **labkey-2** to view all versions.
2. Click the **ROTATE** button and then click **ROTATE**.

You now have two versions of this key, version 2 is the primary one.

**Note**: Using the key rotation commands above, key rotation does NOT re-encrypt already encrypted data with the newly generated key version. If you suspect unauthorized use of a key, you should re-encrypt the data protected by that key and then disable or schedule destruction of the prior key version.

Destroying old keys

**Caution**: If you destroy a key that encrypts existing objects, you will be unable to recover that data, but you will continue to be charged for storage of your objects until you delete them.

In this part, you will not actually destroy a key, but you will investigate the process for doing so.

1. From the **labkey-2** versions screen, click the three vertical dots on the far right of the line for version 1 of the key and select **Destroy**.
2. Read the message in the **Schedule key version 1 for destruction** and click **CANCEL** when done.

You have successfully used KMS keys to encrypt data in Google Cloud Storage.

Click *Check my progress* to verify the objective.

Key Rotation

Check my progress

**Bonus: Encrypting data with the REST API**

The Google Cloud-KMS service also provides a REST API to perform encryption and decryption. The content to be encrypted is specified as part of a JSON document in the REST request, and this content must be encoded using Base64 encoding. This JSON document has the following form:

{"plaintext":"Base64 encoded data to encrypt"}

In this bonus section to the lab, you will manually invoke the REST api using curl commands to demonstrate the capability of the API.

1. This section assumes you still have the Cloud Shell session open and the following environment variables are defined:

KEYRING\_NAME CRYPTOKEY\_1\_NAME CRYPTOKEY\_2\_NAME

If these variables are no longer defined, go back to earlier in the lab and run the commands to create these variables.

1. Run the following command to encode some sample text as Base64 and store it in a variable named **PLAIN\_TEXT**:

PLAIN\_TEXT=$(echo -n "Some text to be encrypted" | base64)

1. Echo the PLAIN\_TEXT variable to verify the text was encoded:

echo $PLAIN\_TEXT

You should see the base64-encoded text.

1. Use the REST API to encrypt the encoded text by calling the encrypt method of your key. Supply the base64-encoded content in the plaintext field of the JSON for your request.

curl \

"https://cloudkms.googleapis.com/v1/projects/$DEVSHELL\_PROJECT\_ID/locations/us/keyRings/$KEYRING\_NAME/cryptoKeys/$CRYPTOKEY\_1\_NAME:encrypt" \

-d "{\"plaintext\":\"$PLAIN\_TEXT\"}" \

-H "Authorization:Bearer $(gcloud auth application-default \

print-access-token)" \

-H "Content-Type: application/json"

The response will be a JSON payload containing the encrypted text in the ciphertext field.

The encrypted text can easily be extracted from the JSON response, and saved to a file by using the command-line utility **jq**. The response from the previous call can be piped into **jq**, which can parse out the **ciphertext** property and save to **data1.encrypted**.

1. Run the following command that repeats the encryption, but this time parses out the **ciphertext** property and saves it to the **data1.encrypted** file.

curl \

"https://cloudkms.googleapis.com/v1/projects/$DEVSHELL\_PROJECT\_ID/locations/us/keyRings/$KEYRING\_NAME/cryptoKeys/$CRYPTOKEY\_1\_NAME:encrypt" \

-d "{\"plaintext\":\"$PLAIN\_TEXT\"}" \

-H "Authorization:Bearer $(gcloud auth application-default \

print-access-token)" \

-H "Content-Type: application/json" \

| jq .ciphertext -r > data1.encrypted

1. View the contents of the **data1.encrypted** file with the following command:

more data1.encrypted

The encrypted text can be decrypted by calling the decrypt method of your key. You must use the same key that was used to encrypt the content.

1. Run the following command to decrypt the contents in the **data1.encrypted** file and save it into the file named **data1.decrypted**.

curl -v \

"https://cloudkms.googleapis.com/v1/projects/$DEVSHELL\_PROJECT\_ID/locations/us/keyRings/$KEYRING\_NAME/cryptoKeys/$CRYPTOKEY\_1\_NAME:decrypt" \

-d "{\"ciphertext\":\"$(cat data1.encrypted)\"}" \

-H "Authorization:Bearer $(gcloud auth application-default \

print-access-token)" \

-H "Content-Type:application/json" \

| jq .plaintext -r | base64 -d > data1.decrypted

1. View the contents of the **data1.decrypted** file with the following command:

more data1.decrypted

You have successfully used KMS keys.