1. Utilize CSEK to encrypt files in cloud storage
2. Rotate encryption keys

**Task 1: Configure required resources**

Create an IAM service account

**Note:** In this lab, you will launch a VM in Compute Engine and perform most of the work on this VM. A service account will first be created to provide the VM with the required permissions to perform the lab.

1. In the GCP Console, on the **Navigation menu > IAM & admin > Service accounts**.
2. Click **CREATE SERVICE ACCOUNT**.
3. Specify the **Service account name** as **cseklab**.
4. Click **Create**.
5. Specify the **Role** as **Storage > Storage Admin**.
6. Click **Continue**.
7. Click **Done**.

Create a Compute Engine VM

1. In the GCP Console, go to **Navigation menu > Compute Engine > VM instances**. Click **Create**.
2. Specify the following, and leave the remaining settings as their defaults:

|  |  |
| --- | --- |
| **Property** | **Value** |
| Name | **cseklab-vm** |
| Region | **us-central1** |
| Zone | **us-central1-a** |
| Machine type | **n1-standard-1** |
| Service account | The **cseklab** service account just created |

1. Click **Create**.
2. Once the VM launches, click the **SSH** button to connect to the VM.

Create a Cloud Storage bucket

**Note:** 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 can be copied from the **CONNECTION DETAILS** pane in Qwiklabs.

1. From the **SSH** terminal connected to the **cseklab-vm**, run the following command to create an environment variable to store the name of your bucket:

export BUCKET\_NAME=[PUT GCP\_PROJECT\_ID HERE]-csek

1. Enter the following command to create the bucket:

gsutil mb -l us gs://$BUCKET\_NAME

Click *Check my progress* to verify the objective.

Configure required resources

Check my progress

Download a sample file using CURL and make two copies

1. Use the following command to download a sample file (this sample file is a publicly available Hadoop documentation HTML file). This file will be copied into the lab's storage bucket:

curl http://hadoop.apache.org/docs/current/\

hadoop-project-dist/hadoop-common/\

ClusterSetup.html > setup.html

1. Make two copies of the file:

cp setup.html setup2.html

cp setup.html setup3.html

**Task 2. Configuring customer-supplied encryption keys**

Generate a CSEK key

**Important**. When using customer-supplied encryption keys, it is up to you to generate and manage your encryption keys. You must provide Google Cloud Storage a key that is a 256-bit string encoded in [RFC 4648 standard base64](https://tools.ietf.org/html/rfc4648#section-4). For this lab, you will generate a key with a random number.

1. In the **cseklab-vm** SSH terminal, run the following command to create a key:

python3 -c 'import base64; import os; print(base64.encodebytes(os.urandom(32)))'

Result (do not copy; this is example output):

b'tmxElCaabWvJqR7uXEWQF39DhWTcDvChzuCmpHe6sb0=\n'

1. Copy the value of the generated key excluding b' and \n' from the command output. Key should be in form of tmxElCaabWvJqR7uXEWQF39DhWTcDvChzuCmpHe6sb0=.

Modify the .boto file

**Note:** The encryption controls are contained in a **gsutil** configuration file named **.boto**.

1. Run the following command in the SSH terminal to verify the **.boto** file exists:

ls -al

1. If you do not see a **.boto** file run the following commands to generate and list it:

gsutil config -n

ls -al

1. To edit the **.boto** file, run the following command:

nano .boto

1. Within the **.boto** file, locate the line with "**#encryption\_key=**". To search in Nano, press **Ctrl+W** and type #encrypt.
2. Uncomment the encryption\_key line by removing the **#** character, and paste the key you generated earlier at the end of the line.

Example (do not copy):

Before:

# encryption\_key=

After:

encryption\_key=tmxElCaabWvJqR7uXEWQF39DhWTcDvChzuCmpHe6sb0=

1. Press **Ctrl+O, ENTER** to save the boto file, and then press **Ctrl+X** to exit nano.

Upload files (encrypted) and verify in the GCP Console

1. Run the following commands to upload two files:

gsutil cp setup.html gs://$BUCKET\_NAME

gsutil cp setup2.html gs://$BUCKET\_NAME

1. Return to the GCP Console and view the storage bucket contents with: **Navigation menu > Storage**, then click on the bucket.

Both the **setup.html** and **setup2.html** files show that they are customer-encrypted.

Click *Check my progress* to verify the objective.

Configuring customer-supplied encryption keys

Check my progress

Delete a local file, copy from cloud storage, and verify encryption

1. Delete the local **setup.html** file, run the following command:

rm setup.html

1. To copy the file back from the bucket, run the following command:

gsutil cp gs://$BUCKET\_NAME/setup.html ./

1. View the file to see whether they made it back with the following command:

cat setup.html

**Task 3: Rotate CSEK keys**

To rotate CSEKs, you can change your encryption\_key configuration value to a

decryption\_key configuration value and then use a new value for the

encryption\_key. Then you can use the rewrite command to rotate keys in the

cloud without downloading and re-uploading the data.

Generate another CSEK key and add to the boto file

1. In the SSH terminal, run the following command to generate a new key:

python3 -c 'import base64; import os; print(base64.encodebytes(os.urandom(32)))'

1. Copy the value of the generated key excluding b' and \n' from the command output. Key should be in form of tmxElCaabWvJqR7uXEWQF39DhWTcDvChzuCmpHe6sb0=.
2. To open the boto file, run the following command:

nano .boto

1. Locate the current **encryption\_key** line and comment it out by adding the # character to the beginning of the line.
2. Add a new line with **encryption\_key=** and paste the new key value.

Result (do not copy; this is example output):

Before:

encryption\_key=2dFWQGnKhjOcz4h0CudPdVHLG2g+OoxP8FQOIKKTzsg=

After:

# encryption\_key=2dFWQGnKhjOcz4h0CudPdVHLG2g+OoxP8FQOIKKTzsg=

encryption\_key=HbFK4I8CaStcvKKIx6aNpdTse0kTsfZNUjFpM+YUEjY==

1. Uncomment the **decryption\_key1=** line by removing the # character.
2. Copy the value of the original **encryption\_key** from the line that was commented out, and paste it for the value of the **decryption\_key1** line.

Result (do not copy; this is example output):

Before:

# encryption\_key=2dFWQGnKhjOcz4h0CudPdVHLG2g+OoxP8FQOIKKTzsg=

encryption\_key=HbFK4I8CaStcvKKIx6aNpdTse0kTsfZNUjFpM+YUEjY==

# decryption\_key1=

After:

# encryption\_key=2dFWQGnKhjOcz4h0CudPdVHLG2g+OoxP8FQOIKKTzsg=

encryption\_key=HbFK4I8CaStcvKKIx6aNpdTse0kTsfZNUjFpM+YUEjY==

decryption\_key1=2dFWQGnKhjOcz4h0CudPdVHLG2g+OoxP8FQOIKKTzsg=

The original encryption\_key line that is commented out can also be completed deleted from the file.

1. Press **Ctrl+O, ENTER** to save the **.boto** file, and then press **Ctrl+X** to exit nano.

Encrypt a file with the new key and decrypt a file with the old key

1. Upload a new file to the bucket. This file will be encrypted with the new key:

gsutil cp setup3.html gs://$BUCKET\_NAME

At this point, **setup.html** and **setup2.html** are encrypted with the original key and **setup3.html** is encrypted with the new key.

1. Delete the local **setup2.html** and **setup3.html** files with the following commands:

rm setup2.html

rm setup3.html

1. To copy the files back from the storage bucket, run the following commands:

gsutil cp gs://$BUCKET\_NAME/setup2.html ./

gsutil cp gs://$BUCKET\_NAME/setup3.html ./

1. View the encrypted file to see whether they made it back with the following commands:

cat setup2.html

cat setup3.html

This lab demonstrates how new keys can be generated for new data, but note that files encrypted with the older keys can still be decrypted.

Rewrite the key for file 1 and comment out the old decrypt key

Rewriting an encrypted file causes the file to be decrypted it using the decryption\_key1 that you previously set, and encrypts the file with the new encryption\_key.

1. Run the following command to rewrite setup.html

gsutil rewrite -k gs://$BUCKET\_NAME/setup.html

At this point, **setup.html** has been rewritten with the new encryption key and **setup3.html** is encrypted with the new key as well. The **setup2.html** file is still encrypted with the original key so that you can see what happens if you don't rotate the keys properly.

1. Open the boto file with the following command:

nano .boto

1. Comment out the current **decryption\_key1** line by adding the # character back in.
2. Press **Ctrl+O, ENTER** to save the **.boto** file, and then press **Ctrl+X** to exit nano.
3. Delete all three local files with the following command:

rm setup\*.html

1. Download **setup.html** and **setup3.html** (both encrypted with the new key) with following commands:

gsutil cp gs://$BUCKET\_NAME/setup.html ./

gsutil cp gs://$BUCKET\_NAME/setup3.html ./

1. View the encrypted files to see whether they made it back through the process using the following commands:

cat setup.html

cat setup3.html

1. Try to download **setup2.html** (encrypted with the original key) using the following command:

gsutil cp gs://$BUCKET\_NAME/setup2.html ./

What happened? **setup2.html** was not rewritten with the new key, so it can no longer be decrypted, and the copy failed.

You have successfully rotated the CSEK keys.