

Introducing Cloud KMS

Overview

In this lab you'll learn how to use some advanced features of Google Cloud Security and Privacy APIs, including:

- Setting up a secure Cloud Storage bucket
- Managing keys and encrypted data using Key Management Service
- Viewing Cloud Storage audit logs

You'll take abridged data from the Enron Corpus, encrypt it, and load it into Cloud Storage.

What you'll learn

- How to encrypt data and manage encryption keys using Key Management Service (KMS).

What you'll use

- Cloud Storage.
- Cloud SDK

Create a Cloud Storage bucket

In order to store the data for this lab you need to create your own Cloud Storage bucket. Pick a name for your Cloud Storage bucket, such as `<YOUR_NAME>_enron_corpus`. For more information on naming buckets, see the Cloud Storage bucket [naming guidelines](#).

Run the following command in Cloud Shell to set a variable to your bucket name:

```
> BUCKET_NAME=<YOUR_NAME>_enron_corpus
```

Now create the bucket by running the following command:

```
> gsutil mb gs://{BUCKET_NAME}
```

Running this command should also help to verify that you've got the `gsutil` command line client set up correctly, authentication is working, and you have write access to the cloud project you're operating under.

After your bucket has been created, move on to the next step to download the Enron Corpus.

Check out the data

The [Enron Corpus](#) is a large database of over 600,000 emails generated by 158 employees of the Enron Corporation. This data has been copied to the Cloud Storage bucket `gs://enron_emails/`.



Download one of the source files locally so that you can see what it looks like by running:

```
> gsutil cp gs://enron_emails/allen-p/inbox/1. .
```

Now tail the downloaded file to verify the email text is there:

```
> tail 1.
```

You should receive the following output:

```
Attached is the Delta position for 1/18, 1/31, 6/20, 7/16, 9/24
<< File: west_delta_pos.xls >>
Let me know if you have any questions.
```

This should display the contents of a plaintext mail file. There are two types of files you'll be looking at: plaintext mail files and image files. If you're interested, use the same mechanism to check out what is in those other files.

Enable Cloud KMS

[Cloud KMS](#) is a cryptographic key management service on Google Cloud. Before using KMS you need to enable it in your project. In this lab you have been provisioned KMS should already have been enabled. You can make sure of this by using one of the `gcloud` CLI commands. Run the following in your Cloud Shell session:

```
> gcloud services enable cloudkms.googleapis.com
```

Note: KMS and other services can also be enabled on your project using the [Cloud Console UI](#) as well.

You shouldn't have received any output. Cloud KMS is now enabled in your project!

Create a Keyring and Cryptokey

In order to encrypt the 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 **qwiklab**. Run the following command in Cloud Shell to set environment variables:

```
> KEYRING_NAME=test CRYPTOKEY_NAME=qwiklab
```

Execute the `gcloud` command to create the KeyRing. For this lab you will be using a global location, but it could also be set to a specific region.

```
> gcloud kms keyrings create $KEYRING_NAME --location global
```

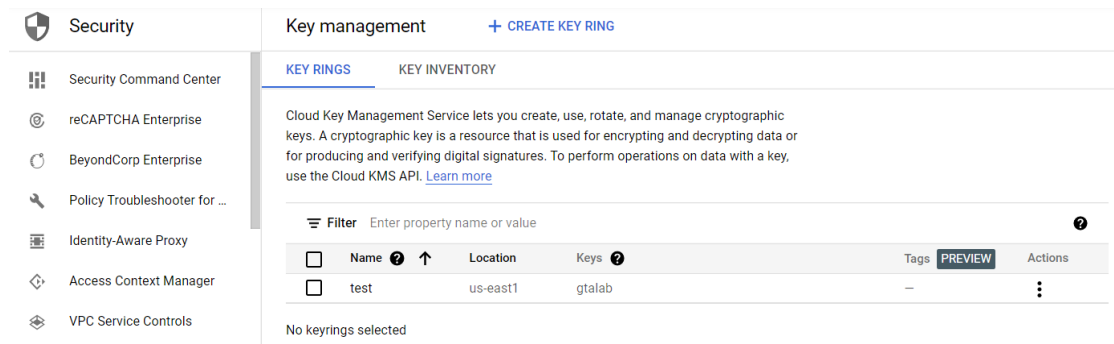
Next, using the new KeyRing, create a CryptoKey named **qwiklab**.

```
> gcloud kms keys create $CRYPTOKEY_NAME --location global \
    --keyring $KEYRING_NAME \
    --purpose encryption
```

Note: CryptoKeys and KeyRings cannot be deleted in Cloud KMS!

You shouldn't see any output. Just like that, you've created a KeyRing and CryptoKey!

Open the [Cryptographic Keys](#) through the Console by going to the **Navigation menu** > **Security** > **Key management**:



The Key Management web UI allows you to view and manage your CryptoKeys and KeyRings. You will use this UI later when you manage permissions.

Encrypt Your Data

Next, try to encrypt some data!

Take the contents of the email you looked at earlier and base64 encode it by running the following:

```
> PLAINTEXT=$(cat 1. | base64 -w0)
```

Note: Base-64 encoding allows binary data to be sent to the API as plaintext. This command works for images, videos, or any other kind of binary data.

Using the encrypt endpoint, you can send the base64-encoded text you want to encrypt to the specified key. Run the following:

```
curl -v
"https://cloudkms.googleapis.com/v1/projects/$DEVSHIELD_PROJECT_ID/locations/global/
keyRings/$KEYRING_NAME/cryptoKeys/$CRYPTOKEY_NAME:encrypt" \
-d '{"plaintext":"$PLAINTEXT"}' \
-H "Authorization:Bearer $(gcloud auth application-default print-access-token)" \
-H "Content-Type: application/json"
```

Note: The encrypt action will return a different result each time even when using the same text and key.

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

Now that your data is encrypted, you can save it to a file and upload it to your Cloud Storage bucket. To grab the encrypted text from the JSON response and save it to a file, use the command-line utility `jq`. The response from the previous call can be

pipelined into jq, which can parse out the ciphertext property to the file 1.encrypted.

Run the following:

```
curl -v
"https://cloudkms.googleapis.com/v1/projects/$DEVSHHELL_PROJECT_ID/locations/global/keyRings/$KEYRING_NAME/cryptoKeys/$CRYPTOKEY_NAME:encrypt" \
-d '{"plaintext":"'${PLAINTEXT}'"}' \
-H "Authorization:Bearer $(gcloud auth application-default
print-access-token)" \
-H "Content-Type:application/json" \
| jq .ciphertext -r > 1.encrypted
```

To verify the encrypted data can be decrypted, call the decrypt endpoint to verify the decrypted text matches the original email. The encrypted data has information on which CryptoKey version was used to encrypt it, so the specific version is never supplied to the decrypt endpoint. Run the following:

```
curl -v
"https://cloudkms.googleapis.com/v1/projects/$DEVSHHELL_PROJECT_ID/locations/global/keyRings/$KEYRING_NAME/cryptoKeys/$CRYPTOKEY_NAME:decrypt" \
-d '{"ciphertext":"'$(cat 1.encrypted)'"}' \
-H "Authorization:Bearer $(gcloud auth application-default
print-access-token)" \
-H "Content-Type:application/json" \
| jq .plaintext -r | base64 -d
```

Note: Usually decryption is performed at the application layer. For a walkthrough on how to encrypt and decrypt data in multiple programming languages, read the [Cloud KMS Quickstart](#). Now that you have verified the text has been encrypted successfully, upload the encrypted file to your Cloud Storage bucket.

```
> gsutil cp 1.encrypted gs://${BUCKET_NAME}
```

Configure IAM Permissions

In KMS, there are two major permissions to focus on. One permissions allows a user or service account to **manage KMS resources**, the other allows a user or service account to use keys to **encrypt and decrypt** data.

The permission to manage keys is `cloudkms.admin`, and allows anyone with the permission to create KeyRings and create, modify, disable, and destroy CryptoKeys.

The permission to encrypt and decrypt is

`cloudkms.cryptoKeyEncrypterDecrypter`, and is used to call the encrypt and decrypt API endpoints.

For this exercise, you will use the current authorized user to assign IAM permissions.

To get the current authorized user, run the command below:

```
USER_EMAIL=$(gcloud auth list --limit=1 2>/dev/null | grep '@' | awk '{print $2}')
```

Next, assign that user the ability to manage KMS resources. Run the following `gcloud` command to assign the IAM permission to manage the KeyRing you just created:

```
gcloud kms keyrings add-iam-policy-binding $KEYRING_NAME \
  --location global \
  --member user:$USER_EMAIL \
  --role roles/cloudkms.admin
```

Since CryptoKeys belong to KeyRings, and KeyRings belong to Projects, a user with a specific role or permission at a higher level in that hierarchy inherits the same permissions on the child resources. For example, a user who has the role of Owner on a Project is also an Owner on all the KeyRings and CryptoKeys in that project. Similarly, if a user is granted the `cloudkms.admin` role on a KeyRing, they have the associated permissions on the CryptoKeys in that KeyRing.

Without the `cloudkms.cryptoKeyEncrypterDecrypter` permission, the authorized user will not be able to use the keys to encrypt or decrypt data. Run the following `gcloud` command to assign the IAM permission to encrypt and decrypt data for any CryptoKey under the KeyRing you created:

```
gcloud kms keyrings add-iam-policy-binding $KEYRING_NAME \
  --location global \
  --member user:$USER_EMAIL \
  --role roles/cloudkms.cryptoKeyEncrypterDecrypter
```

Now you can view the assigned permissions in the Cryptographic Keys section of [Key Management](#).

Check the box by the name of the key ring (test), then click **Permissions** in the right column.

This will open up a menu where you can see the accounts and permissions for the key ring you just added.

Back up data on the Command Line

Now that you have an understanding of how to encrypt a single file, and have permission to do so, you can run a script to backup all files in a directory. For this example, copy all emails for **allen-p**, encrypt them, and upload them to a Cloud Storage bucket.

First, copy all emails for **allen-p** into your current working directory:










```
gsutil -m cp -r gs://enron_emails/allen-p .
```

Now copy and paste the following into Cloud Shell to back up and encrypt all the files in the **allen-p** directory to your Cloud Storage bucket:

```
MYDIR=allen-p
FILES=$(find $MYDIR -type f -not -name "*.encrypted")
for file in $FILES; do
  PLAINTEXT=$(cat $file | base64 -w0)
  curl -v
  "https://cloudkms.googleapis.com/v1/projects/$DEVSHIELD_PROJECT_ID/locations/global/keyRings
/$KEYRING_NAME/cryptoKeys/$CRYPTOKEY_NAME:encrypt" \
  -d '{"plaintext\":"$PLAINTEXT\"}' \
  -H "Authorization:Bearer $(gcloud auth application-default print-access-token)" \
  -H "Content-Type:application/json" \
  | jq .ciphertext -r > $file.encrypted
done
gsutil -m cp allen-p/inbox/*.encrypted gs://${BUCKET_NAME}/allen-p/inbox
```

This script loops over all the files in a given directory, encrypts them using the KMS API, and uploads them to Cloud Storage.

After the script completes, you can view the encrypted files when you click Storage from the Console's left menu. To find the files, go to **Navigation menu > Cloud Storage > Browser > YOUR_BUCKET > allen-p > inbox**. You should see something like this:

<input type="checkbox"/>	Name	Size	Type	Storage class
<input type="checkbox"/>	 1..encrypted	2.42 KB	application/octet-stream	Multi-Regional
<input type="checkbox"/>	 10..encrypted	5.85 KB	application/octet-stream	Multi-Regional
<input type="checkbox"/>	 11..encrypted	2.05 KB	application/octet-stream	Multi-Regional
<input type="checkbox"/>	 12..encrypted	2.15 KB	application/octet-stream	Multi-Regional
<input type="checkbox"/>	 14..encrypted	1.45 KB	application/octet-stream	Multi-Regional
<input type="checkbox"/>	 15..encrypted	6.04 KB	application/octet-stream	Multi-Regional
<input type="checkbox"/>	 17..encrypted	1.92 KB	application/octet-stream	Multi-Regional
<input type="checkbox"/>	 19..encrypted	1.45 KB	application/octet-stream	Multi-Regional
<input type="checkbox"/>	 20..encrypted	1.74 KB	application/octet-stream	Multi-Regional

View Cloud Audit Logs

Google Cloud Audit Logging consists of two log streams, Admin Activity and Data Access, which are generated by Google Cloud services to help you answer the question "who did what, where, and when?" within your Google Cloud projects.

To view the activity for any resource in KMS, return to the Cryptographic keys page (**Navigation menu > Security > Key management**), check the box next to your key ring, then click on the **Activity** tab in the right menu. This will take you to the Cloud Activity UI, where you should see the creation and all modifications made to the KeyRing.