Using Customer Managed Encryption Keys for Encrypting GCP Disks and Databases (Preview)

Date published: 2021-11-17
Date modified: 2021-11-17

Legal Notice

Contents

Introduction

By default, a Google-managed encryption key is used to encrypt disks and databases in CDP clusters, but you can optionally configure CDP to use a customer-managed encryption key (CMEK) instead.

When a CMEK is provided during environment registration, all the disks (Data Lake, FreeIPA, and Data Hub) and databases are encrypted using that key.

To set up a CMEK, perform the following tasks:

1. Review the CMEK requirements.
2. Create a key ring and an encryption key.
3. Assign the required permissions to the encryption key.
4. Install Beta CDP CLI.
5. Create a CDP environment passing the encryption key.

This document guides you through all the required steps performed using the GCP console, Google Cloud Shell, and Beta CDP CLI.

Related links
Customer-managed encryption keys (CMEK)

CMEK requirements

The CMEK needs to meet the following requirements:

- CMEK needs to be in the same region as the environment.
- The key should have the following permissions for the compute and cloud sql service agents:
    - `Cloud KMS CryptoKey Encrypter/Decrypter`

The instructions below show you how to create a CMEK that meets these requirements.

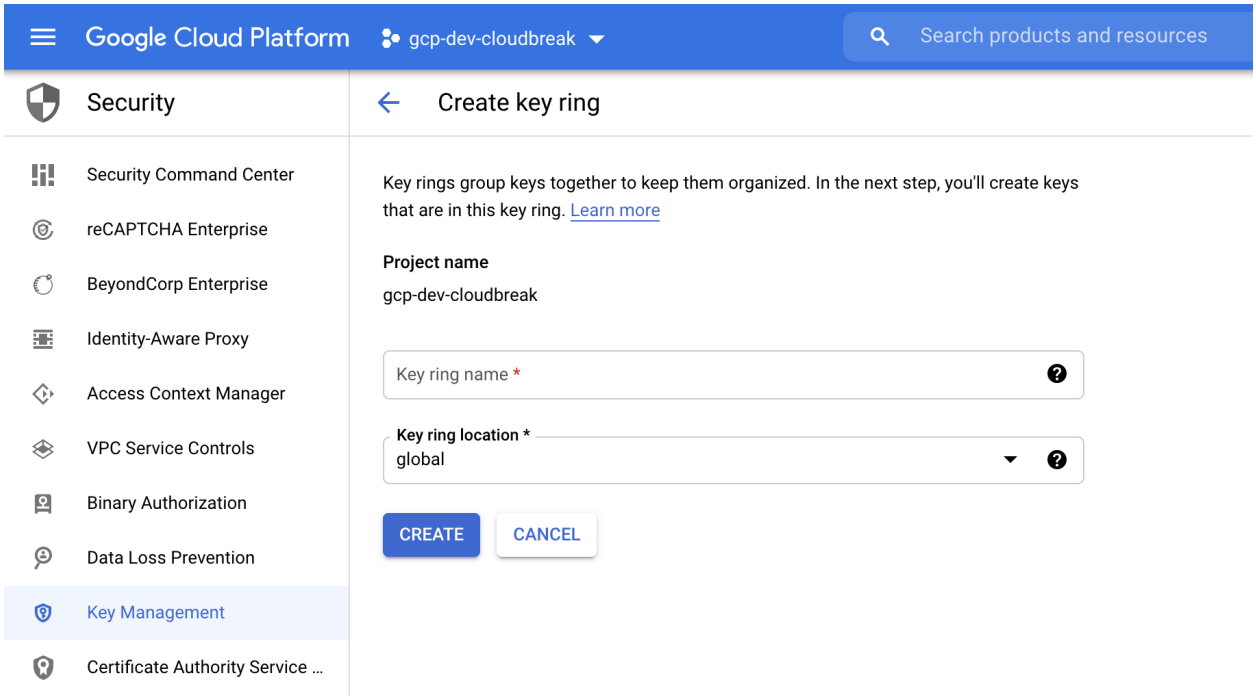
Create key ring and encryption key

Use the following instructions to create a key ring and an encryption key.

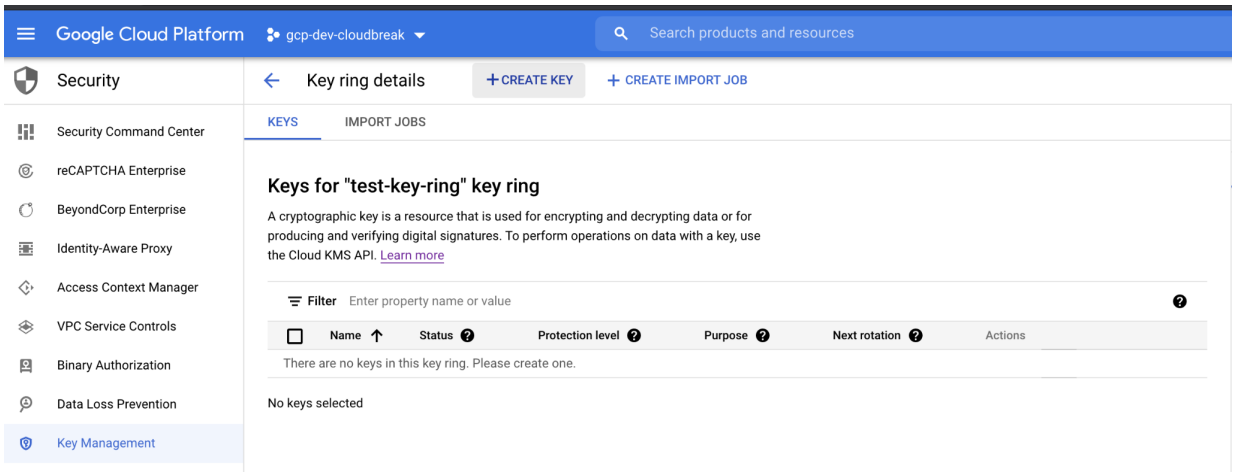
Note: Key rotation and storage are Google-managed.

Steps

1. In the GCP console, navigate to **Security > Key Management.**
2. Create a key ring or use any existing one. Ensure that the key ring location and the location of the resources you create for the CDP environment are the same.

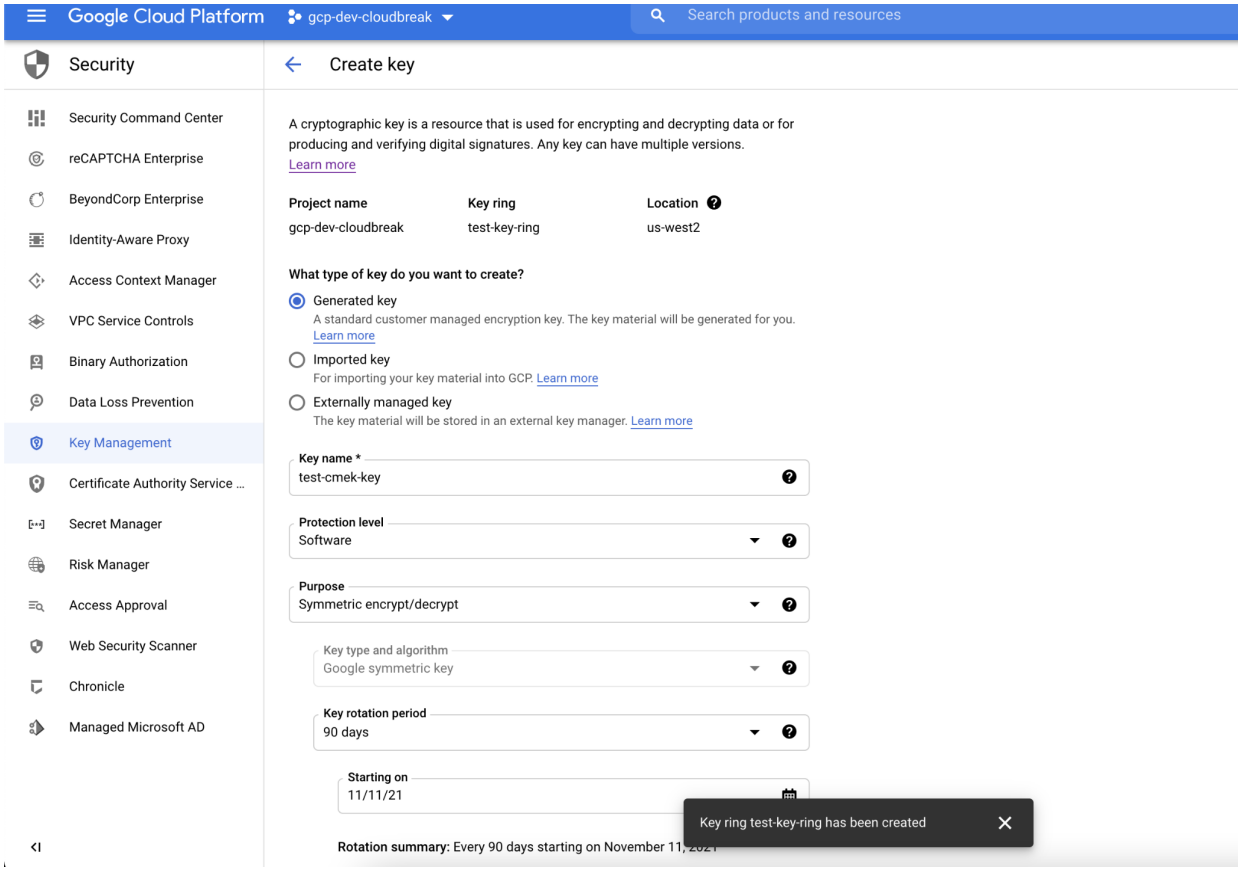


3. Navigate to the key ring that you have previously created.
4. Click on **+Create Key to c**reate a new key inside the key ring.



5. Under **What type of key do you want to create?**, select **Generated key**.
6. Under **Key name**, enter the name for your key.
7. From the **Protection level** dropdown, select **Software**.
8. From the **Purpose** dropdown, select **Symmetric encrypt/decrypt**.
9. Use the default values for **Rotation period** and **Starting on**.

10. Click **Create**.



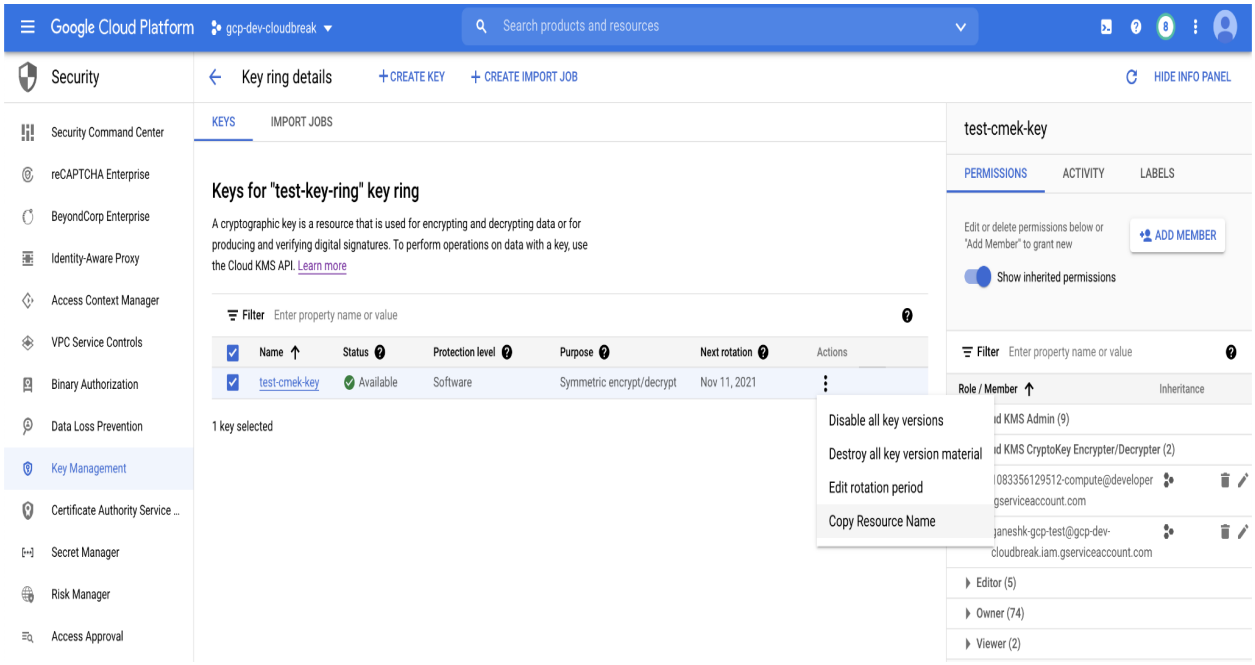
Assign the required permissions to the encryption key

Once the key has been created, you need to assign the required permissions to it. The following commands can be used to set it up using Google Cloud Shell.

Prerequisites

Make sure that you have the following available:

- The project number of the Google project number where the compute and SQL resources are created (PROJECT_NUMBER).
- Copy the key ring resource name (KEYRING_RESOURCE_NAME) and the key resource name (KEY_RESOURCE_NAME) from the Google cloud console. You can copy the key ring resource name from the dropdown after clicking three vertical dots next to the key ring. You can copy the key resource name in a similar manner.



Steps

1. If the cloud sql service agent does not exist in the project, create it using:

```
gcloud beta services identity create --service=sqladmin.googleapis.com --project=<project_name>
```

2. Assign the IAM policy to encrypt and decrypt KMS keys. Replace the variables in caps with the values obtained earlier:

```
gcloud kms keys add-iam-policy-binding KEY_RESOURCE_NAME \
 --location=GCP_REGION \
 --keyring=KEYRING_RESOURCE_NAME \
 --member=serviceAccount:service-PROJECT_NUMBER@gcp-sa-cloud-sql.iam.gserviceaccount.com \
 --role=roles/cloudkms.cryptoKeyEncrypterDecrypter
```

3. Assign the IAM policy to the compute service agent. Replace the variables in caps with the values obtained earlier:

```
gcloud kms keys add-iam-policy-binding KEY_RESOURCE_NAME \
 --location=GCP_REGION \
 --keyring=KEYRING_RESOURCE_NAME \
 --member=serviceAccount:service-PROJECT_NUMBER@compute-system.iam.gserviceaccount.com \
 --role=roles/cloudkms.cryptoKeyEncrypterDecrypter
```

Install Beta CDP CLI

The steps below can only be performed via CDP Beta CLI. They are not supported via the standard CDP CLI.

Prerequisites

Do not install both the regular and beta CLIs in the same Python environment, as they use the same entry points and will therefore conflict. Either use a separate virtual environment or uninstall the cdpcli first before installing cdpcli-beta.

Steps

1. Install Python if needed, as described in CDP CLI documentation for your platform. Do not install CDP CLI. Instead, proceed to step 2.
2. Run the following to install Beta CDP CLI:
   pip3 install cdpcli-beta
3. Configure access keys as described in Generating an API access key.
4. If you have previously installed Beta CDP CLI, you can update it to the latest version using:

   ```
   pip3 install cdpcli-beta --upgrade --user
   ```

Create a CDP environment with an encryption key

Create an environment passing the `--encryption-key` parameter as shows in this example:

```
cdp environments create-gcp-environment \
--no-use-public-ip \
--environment-name <ENVIRONMENT_NAME> \
--credential-name <EXISTING_CREDENTIAL-NAME>\
--region <REGION>\
--security-access securityGroupIdForKnox=<SG_NAME1>,defaultSecurityGroupId=<SG_NAME2> \
--public-key <PUBLIC_SSH_KEY>\
--log-storage storageLocationBase=<LOGS_STORAGE_LOCATION> \
--existing-network-params
networkName=<NETWORK>,subnetNames=<SUBNET>,sharedProjectId=<PROJECT_ID>\
--workload-analytics \
--encryption-key <PATH_TO_THE_ENCRYPTION_KEY>
```

Note: If the `--encryption-key` parameter is not provided the GCP resources are not encrypted using CMEK, falling back to the default behaviour of Google managed encryption.

Next, create a Data Lake and mappings using the usual commands. Once the environment is running, Data Hubs can be created using the usual steps.