# Bucket To Bucket Encryption

Installation & Configuration Guide

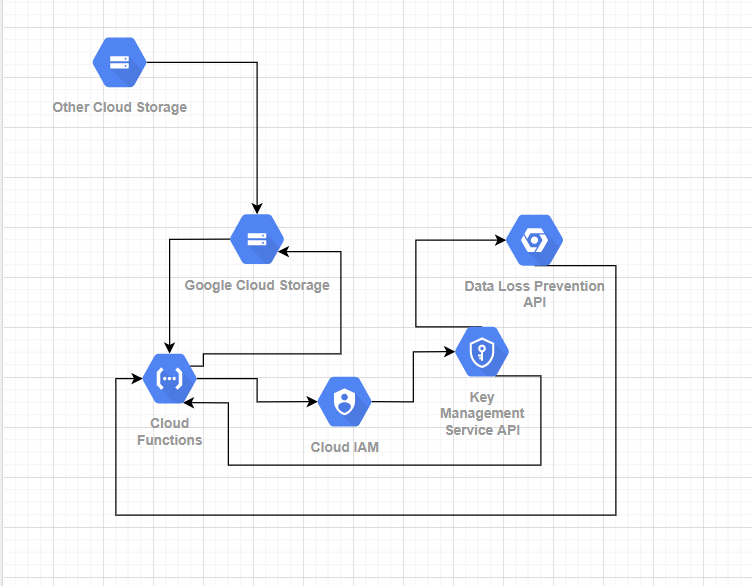
# Ver. 01.00

Date: 20 October 2020

**Introduction**

Bucket to Bucket Encryption is a Google Cloud utility function that encrypts data as it is passed between buckets. The code is placed inside of a Google Cloud Function so that whenever anything is put into the specific bucket the Cloud Function applies to, that data is encrypted or deidentified with whatever method the user has selected. Bucket to Bucket Encryption works with Google Cloud Key Management Service (KMS), Google Cloud Build, and Data Loss Prevention (DLP) APIs.

The below diagram shows how the Google Cloud function works. The user transfers files from another Google Cloud Platform project, or another Cloud Service Provider, to Google Cloud Storage. This triggers the Google Cloud Function, which ensures the user has access to the DLP and KMS APIs. The data is then deidentified or encrypted based on user input and returned to Google Cloud Storage.



**Concepts**

**Format Preserving Encryption (FPE):** Format Preserving Encryption, or FPE, is data encryption of plaintext that results in ciphertext of the same length as the plaintext. Using a wrapped AES key, data can be encrypted with FPE through the Google Cloud DLP API.

**Key Management Service (KMS) API:** The Google Cloud Key Management Service, or KMS, API deals with cryptographic security keys and key rings. It can create new symmetric or asymmetric keys, supports the use of on-premises or cloud-based Hardware Security Modules (HSMs) for key security, allows the import of customer managed keys, and integrates with other Google Cloud API like such as the Data Loss Prevention API.

**Data Loss Prevention (DLP) API:** Google Cloud’s Data Loss Prevention, or DLP, API is used for the protection of sensitive data. Data can be de-identified through the DLP API with several different methods. FPE, replacement, masking, and date shifting are all examples of data de-identification available with the DLP API. This API also allows for the inspection and identification of data that is sensitive in a document, and integrates with other API very easily.

**Key Wrapping:** To de-identify data with the DLP API, an AES key must first be imported from the local machine of the user. When this key is imported to Google Cloud, the AES key is wrapped by a wrapping key to ensure the key has not been compromised. When a key is wrapped, this means another key, which is in Google Cloud, encrypts the key being imported. This ensures the security of the key in transit, and allows the user to manage the key before it is sent to the Cloud.

**Character Masking Deidentification:** Character masking is a type of deidentification where the specified infotypes are masked by a character supplied by the user. The amount of letters/numbers to mask are also supplied by the user. Character masking is NOT reversible.

Example Input Email: [example@email.com](mailto:example@email.com); mask 5 letters with \*

Output: \*\*\*\*\*le@email.com

**Replacement Deidentification:** Replacement is a form of deidentification where the user supplies a replacement phrase to replace the selected infotype with.

Example Input Credit Card: 2389 2340 5025 2202; replace with [creditcard#]

Output: [creditcard#]

**Redaction Deidentification:** Redaction is a form of deidentification where the user specifies an infotype to be completely omitted from a table.

Example Input Sentence: My email address is [example@example.com](mailto:example@example.com).

Output: My email address is .

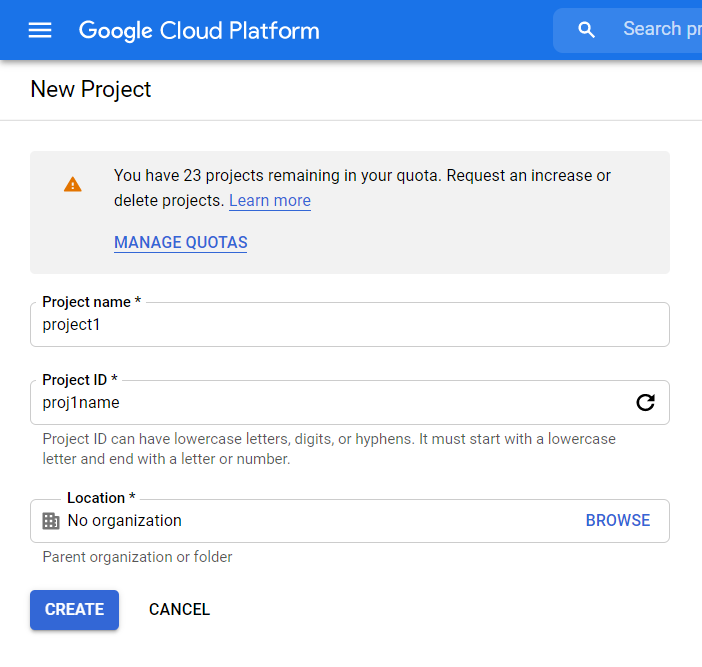
**Infotypes:** An infotype is a type of sensitive data, such as a phone number or credit card number, which is identified by a key phrase, like PHONE\_NUMBER or CREDIT\_CARD\_NUMBER.

**Installation**

**Before you Begin:** Ensure you have a Google Cloud account created

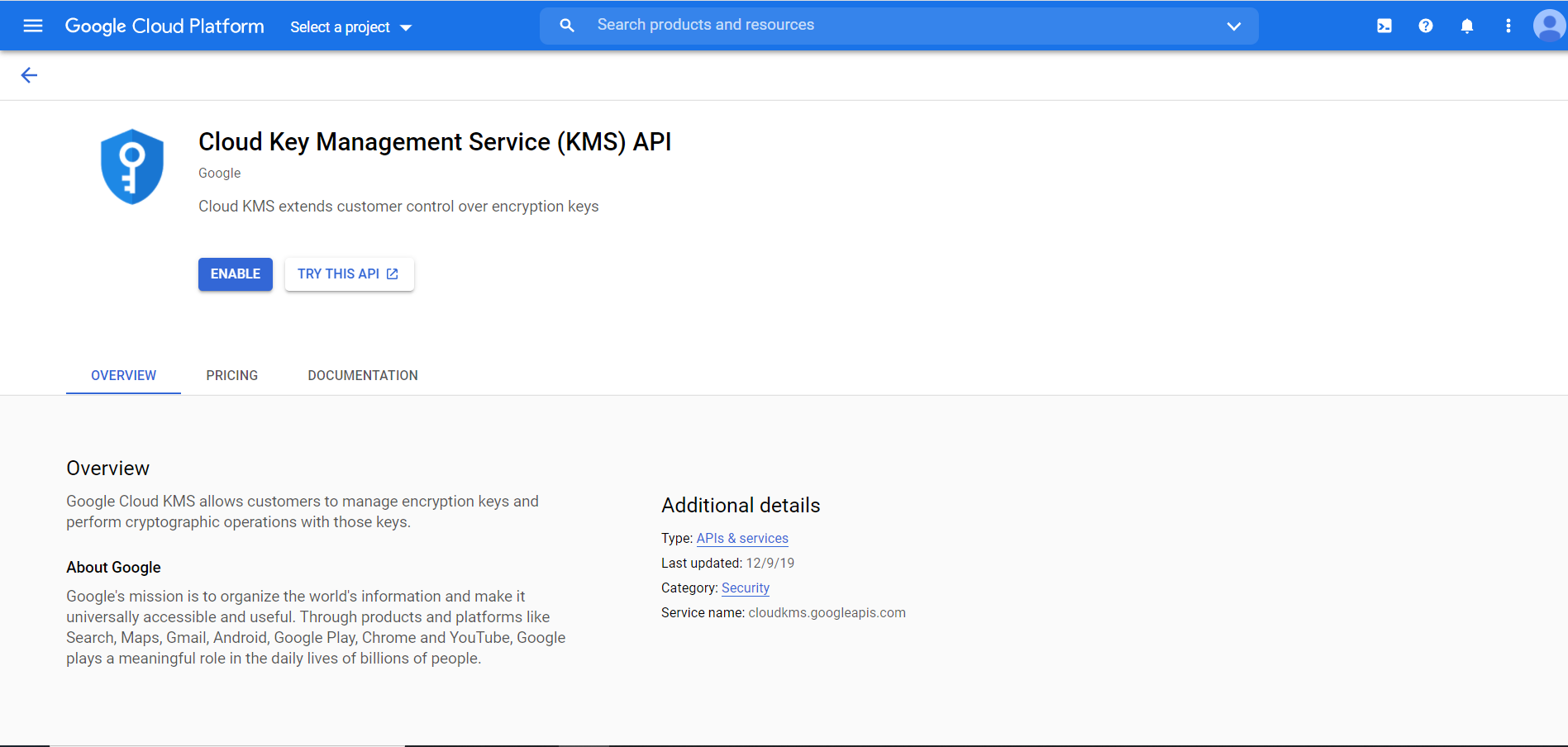
**Setting up your Google Cloud Account:**

1. Creating a Google Cloud Project
   1. Login into your Google Cloud account
   2. Click the “Select a Project” button at the top of the main page
   3. Select “New Project” from the new menu
   4. Name your project

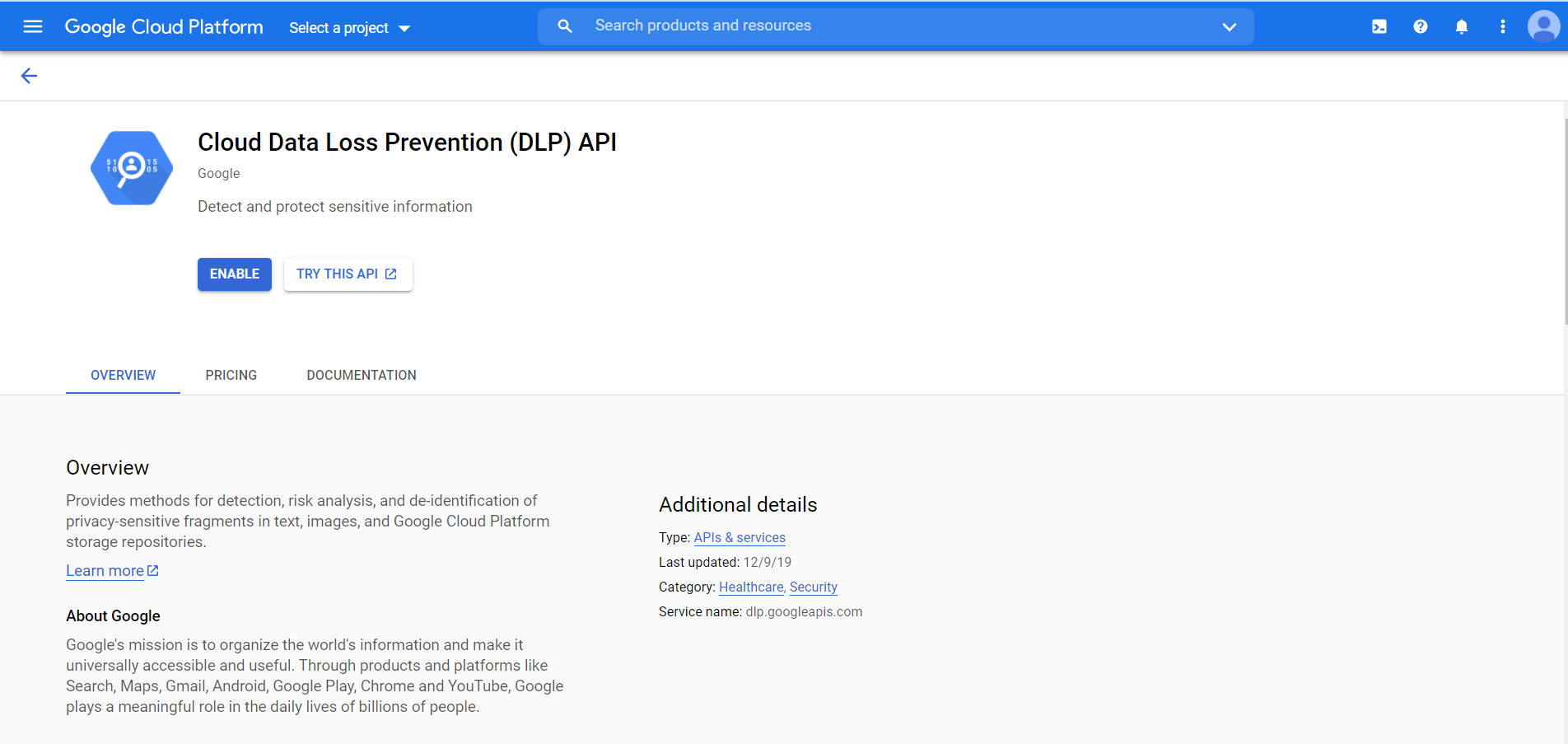


* 1. Optional: Change the project id to something easier to remember via the edit button under project name
  2. If applicable: Select your organization name

1. Enabling the Google Cloud KMS API
   1. Click [this link](https://console.cloud.google.com/marketplace/product/google/cloudkms.googleapis.com?q=search&referrer=search&authuser=3&project=fp4567) to get to the page pictured below
   2. Ensure your project is selected, and then click enable



1. Enabling the Google Cloud DLP API
   1. Click [this link](https://console.cloud.google.com/marketplace/product/google/dlp.googleapis.com?q=search&referrer=search&project=fp4567) to get to the page pictured below
   2. Ensure your project is selected, then click enable

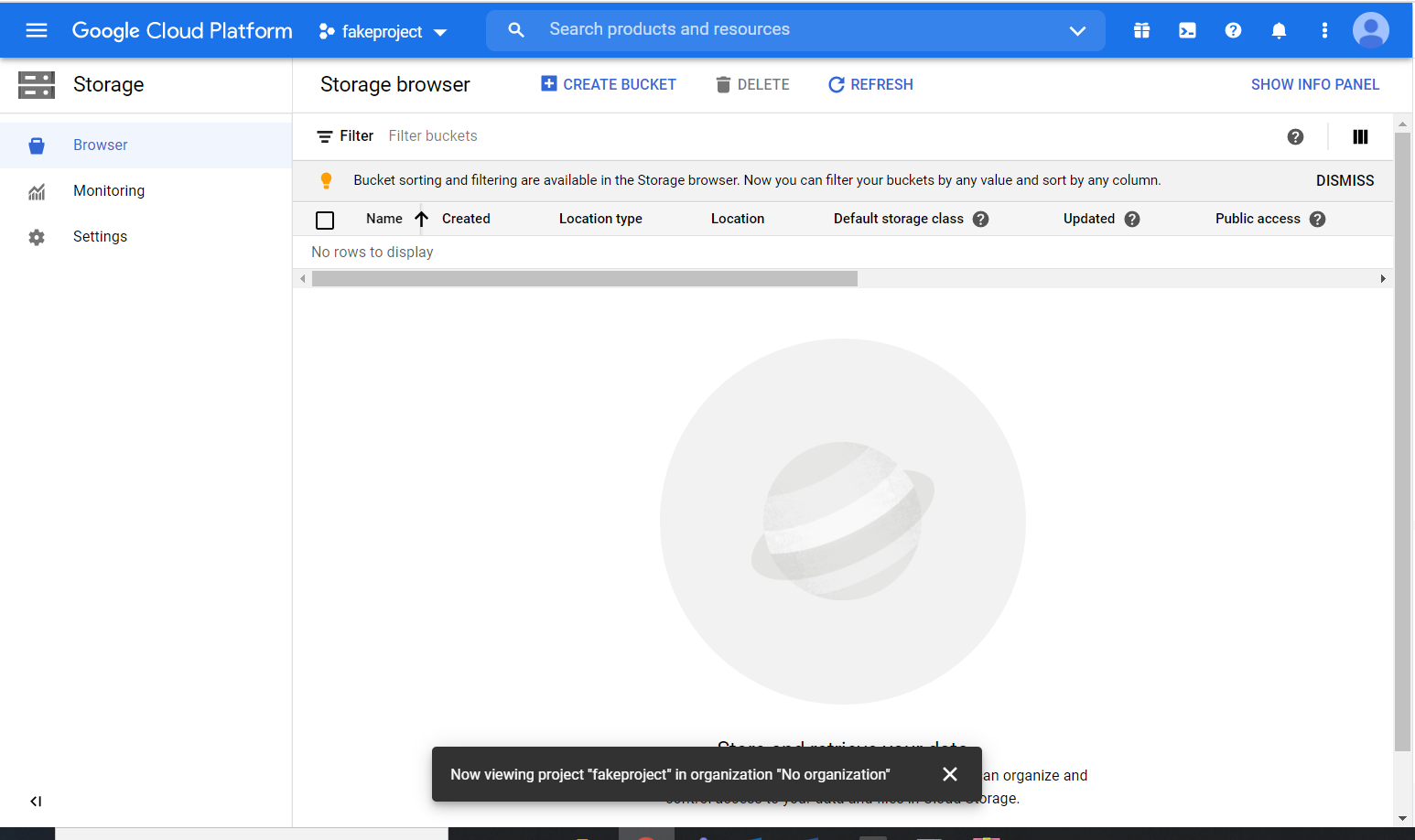


1. Creating a bucket that triggers the Google Cloud Function
   1. Click the image below on your project’s homepage, and select “Storage” from the list

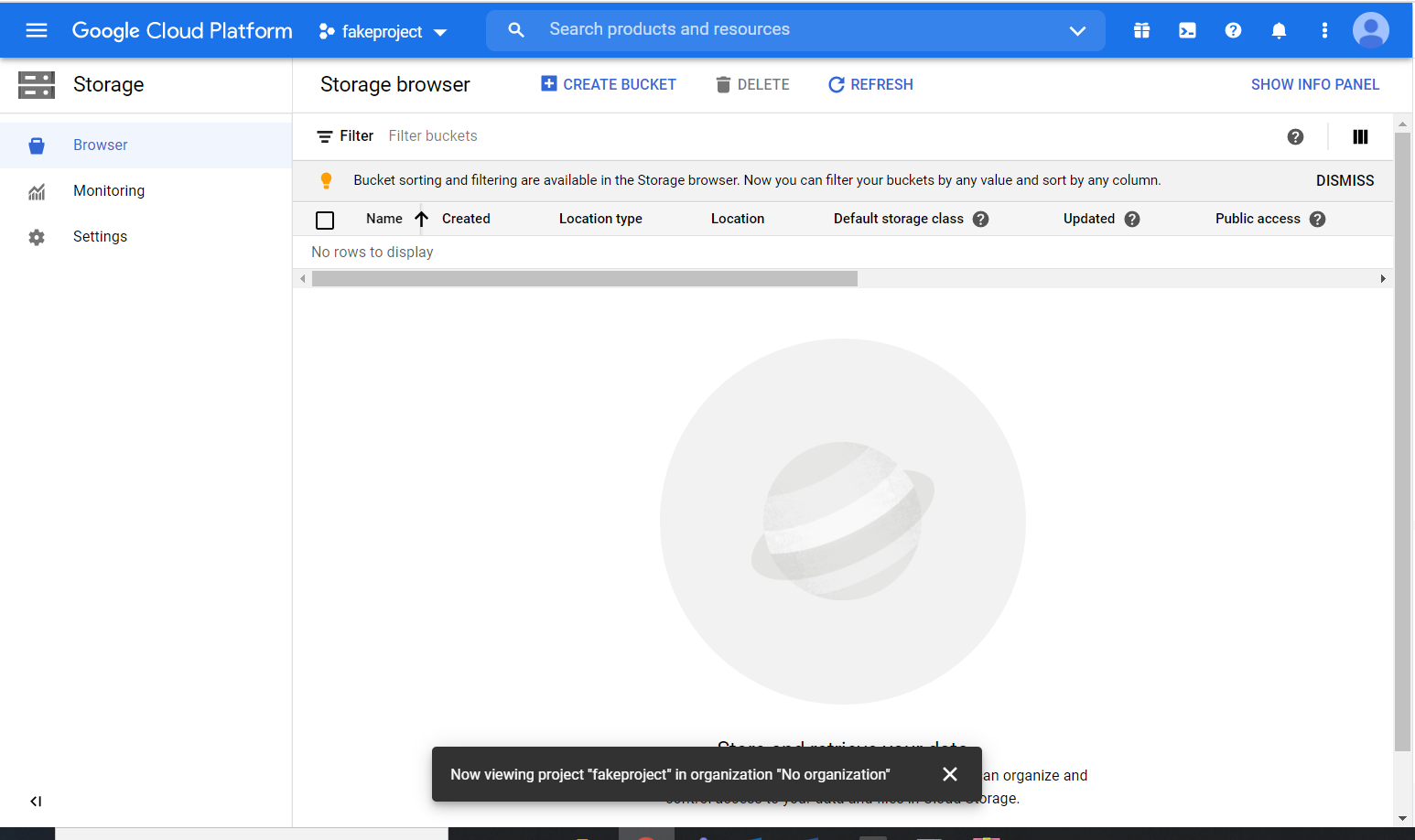


* 1. Select “Create bucket”
  2. Give your bucket a unique name and select “Continue”
  3. Under “Location Type” select Region
  4. Select your current region from the “Location” drop down menu and select “Continue”
  5. Leave “Standard” selected and select “Continue”
  6. Leave “Fine-grained” selected and click “Continue”
  7. Select Google Cloud managed key and click “Create”

1. Creating a bucket to deploy the Google Cloud Function from
   1. Create a bucket exactly the same as step 5, with a different name.



1. Create a “protected” bucket that stores the files once deidentified/encrypted
   1. Create a bucket exactly the same as step 5, with a different name.



1. Creating your wrapping key ring and wrapping key (for FPE only)
   1. Click the image below on your project’s homepage, and select “Security” from the list

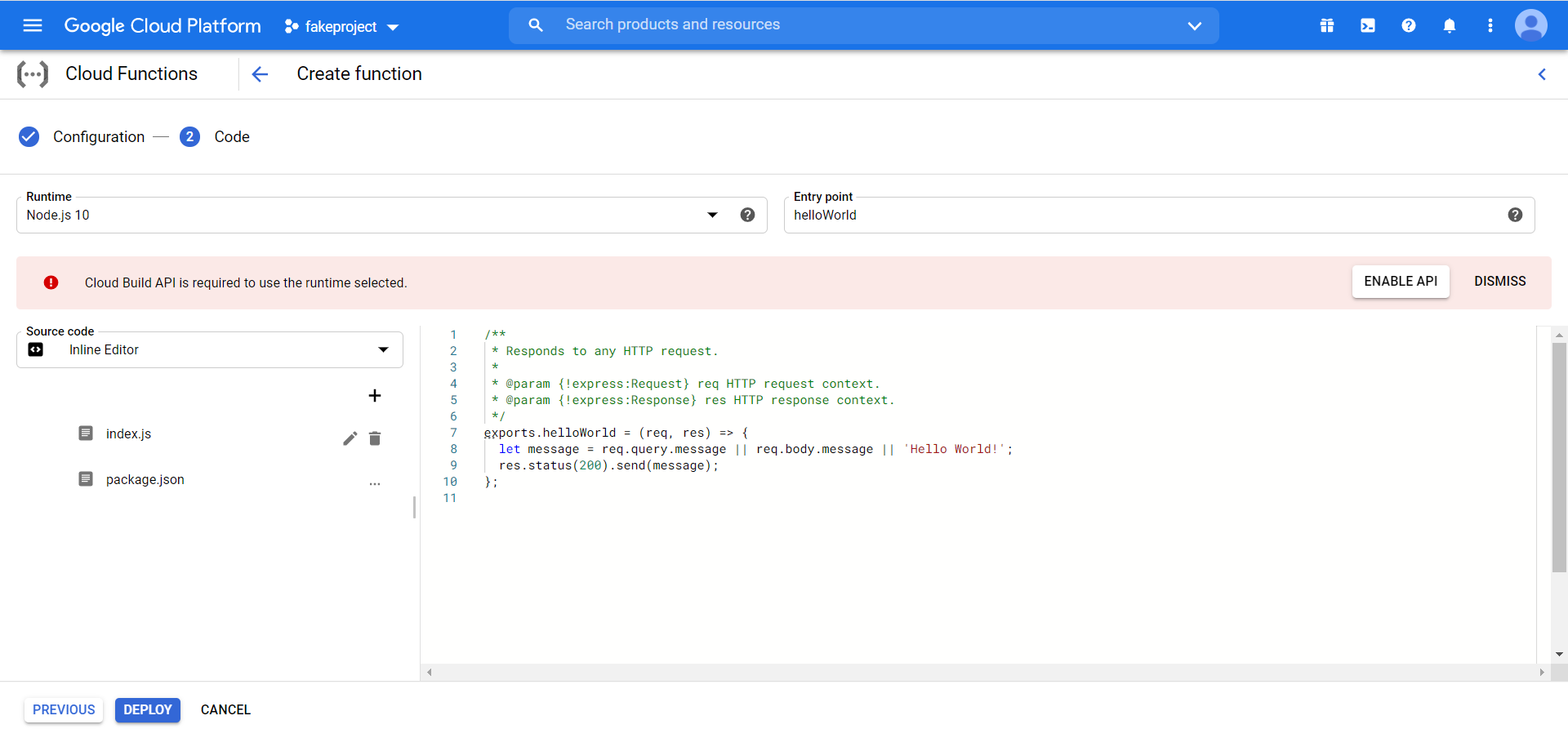


* 1. Select “Cryptographic Keys” on the menu on the left
  2. Select “Create Key Ring”
  3. Choose a key ring name, and leave the location as “global”
  4. Leave “Generated Key” selected, and give your key a name
  5. Leave everything else as default, and create the key

1. Creating the Google Function
   1. Click the image below on your project’s homepage, and select “Cloud Function” from the list



* 1. Select “Create Function”
  2. Give your function a name
  3. Select the region you are in
  4. Under “Trigger Type” select Cloud Storage
  5. Under “Event Type” select Finalize/Create
  6. Select the bucket you wish this Google Cloud Function to be triggered by
  7. Click the “Save” button
  8. Click “Next”
  9. Click “Enable API” when the below page appears



* 1. Select “Java 11” under “Runtime”
  2. Enter “com.mycompany.EncryptionToBucket” as the entry point
  3. Under source code, select zip upload.
  4. Select the downloaded zip file (my-app.zip) and select the staging bucket you created to stage the function.
  5. Once the function has a green check mark at the front, move onto the next steps

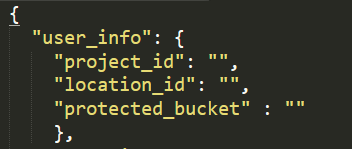
1. Setting IAM Roles
   1. Click the image below on your project’s homepage, and select “IAM & Admin” from the list



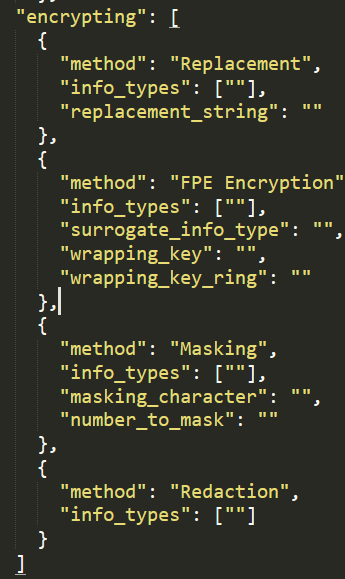
* 1. Select the pencil on the IAM member with the name “APP Engine default service account”
  2. Click add another role and select the DLP Administrator role
  3. Add another role for Cloud KMS Encrypter/Decrypter
  4. Do the steps d-f again, but for the role with the name “Google Cloud Functions Service Agent”

**Utilizing the Google Cloud Function:**

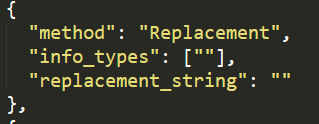
1. Using the cloudFuncConfig.json file
   1. Open the cloudFuncConfig.json file
   2. Enter the appropriate information into the “user\_info” section of your cloudFuncConfig.json file, pictured below



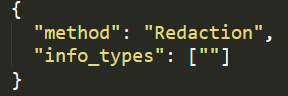
* 1. Under “encrypting”, choose the methods you wish to run on your .csv file. All the different options you can perform are pictured below. A list of usable Info Types can be found [here](https://cloud.google.com/dlp/docs/infotypes-reference).



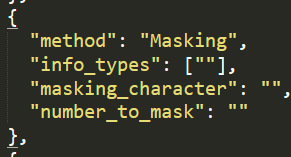
* + 1. Replacement requires the info types to be deidentified and the string to replace the info types with.



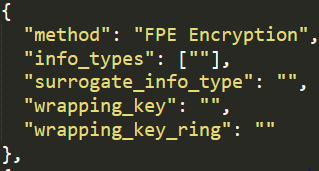
* + 1. Redaction requires the info types to be deidentified.



* + 1. Character masking requires the info types to be deidentified, the masking character to use, and the number of characters to mask.



* + 1. FPE encryption requires the info types to be deidentified, the wrapping key name, the wrapping key ring name, and the surrogate info type to replace the selected info types with.



* 1. DELETE ANY METHODS FROM THE JSON FILE YOU ARE NOT USING. An example can be seen in the image below.



* 1. Now upload cloudFuncConfig.json to the trigger bucket.

1. Testing the Google Cloud function
   1. After completing all of the above steps, upload your .csv file to the trigger bucket
   2. Go back to bucket selection and open the bucket you created to store your encrypted/deidentified files, after the google cloud function runs, and download the newly created “<your-original-filename>\_protected.csv” file to see the Google Cloud Function worked.

**Limitations**

Google Cloud currently has some limitations on the deidentification and encryption methods it uses. As of the current time, Deterministic Encryption is not supported in Java 11, which is the language this Google Cloud Function is written in. Dateshifting deidentification and FPE Decryption will be implemented soon.