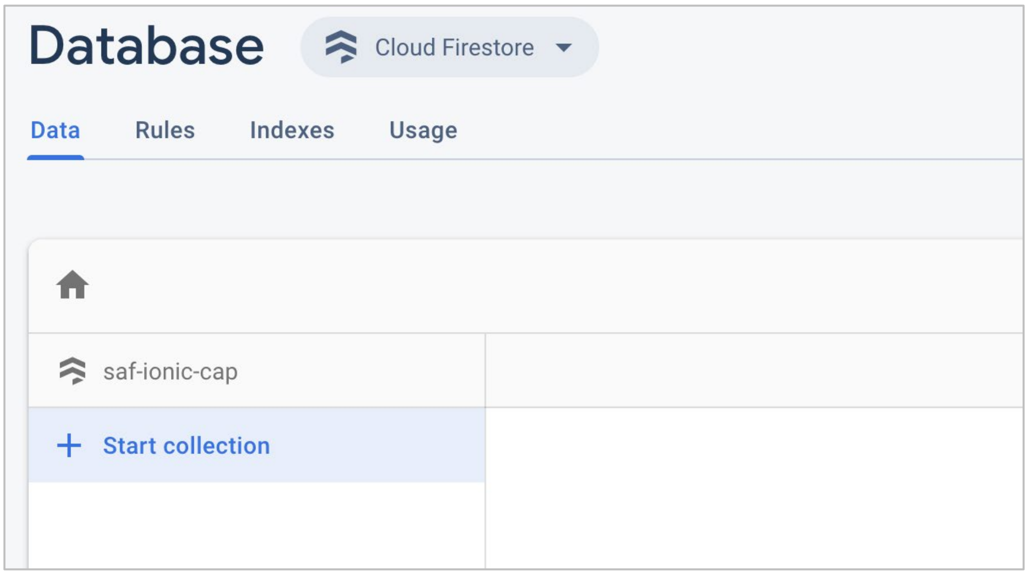
# Practical 9: Database

**Objectives:**

Enable Firebase Storage and Firestore



**Tasks:**

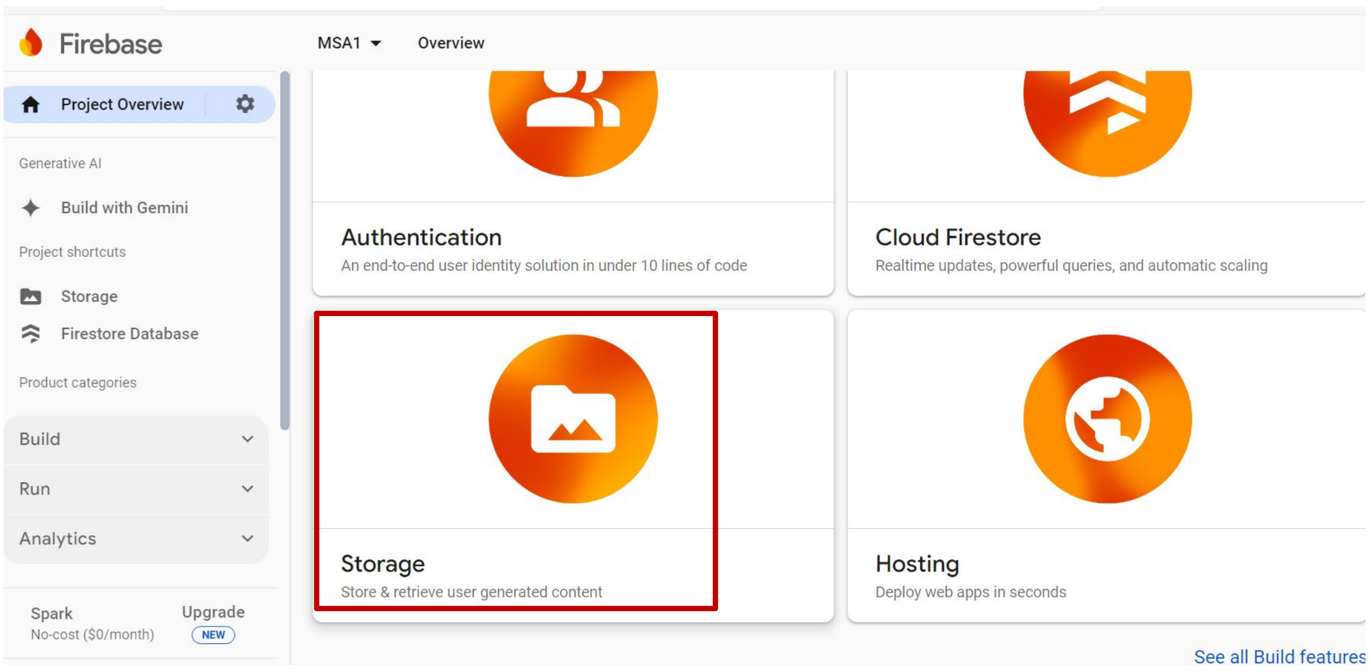
1. Create Firebase Storage
2. Create Firebase Database

# 1 Create Firebase Storage

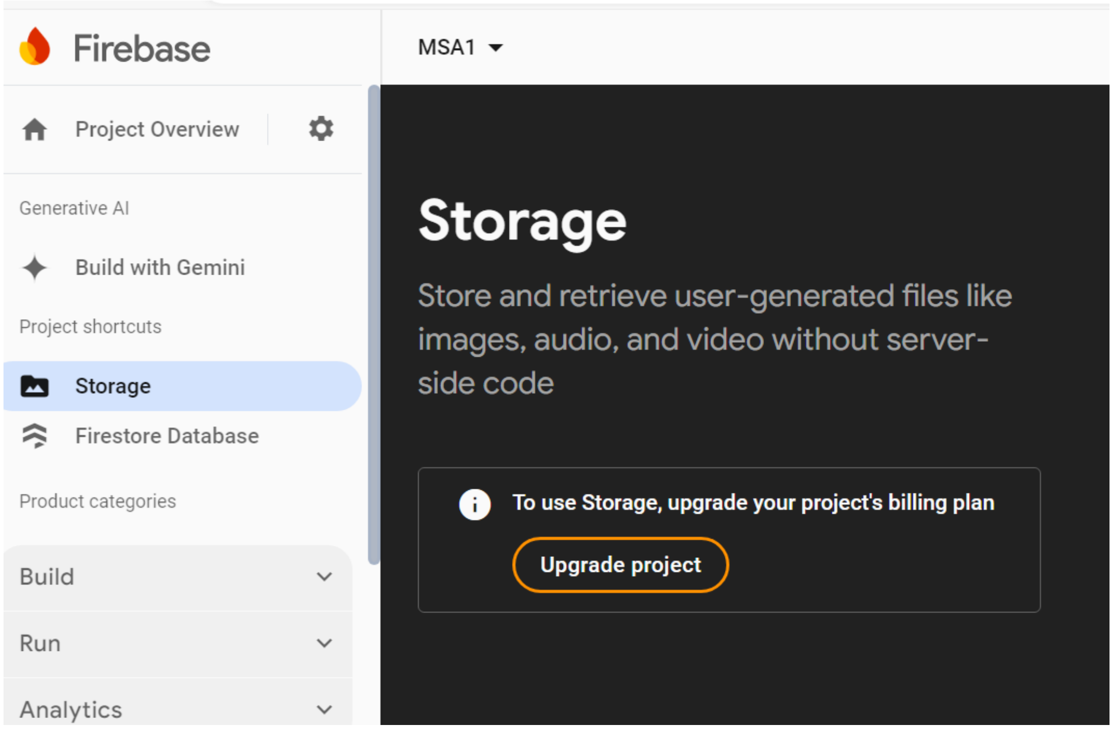
## 1.1 Enable Firebase Storage

1. Go to https://firebase.google.com.
2. **Sign in** using your Google Account.
3. Select **Go to console**.
4. Select the *Skippy Q* Project.

   Select **Storage**.



5. Select **Upgrade project**.



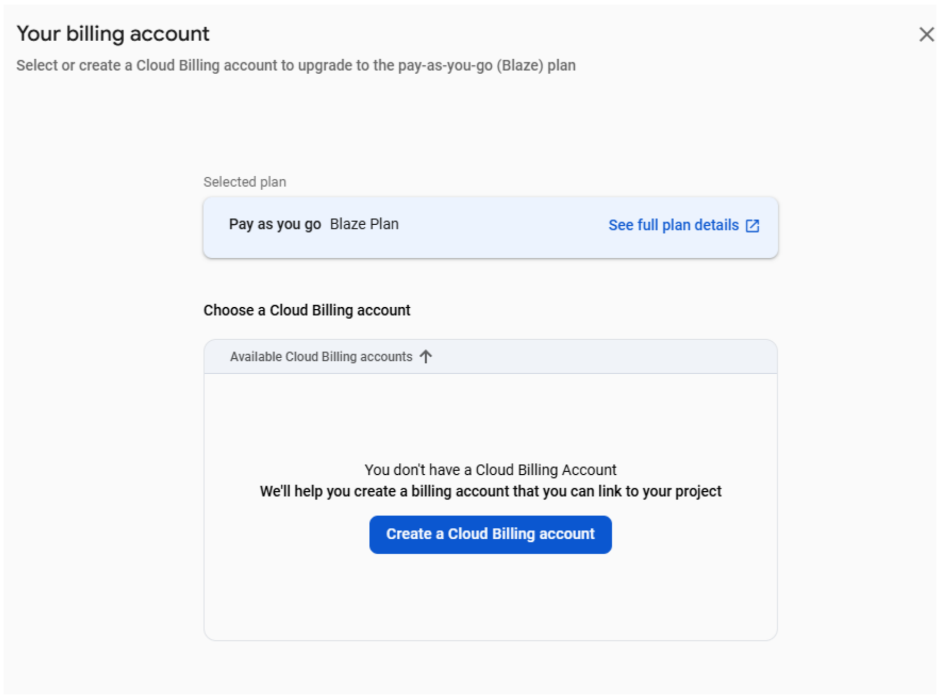
Storage

Store and retrieve user-generated files like images, audio, and video without server-side code

To use Storage, upgrade your project's billing plan

Upgrade project

6. Select **Create a Cloud Billing account.**



Your billing account

Select or create a Cloud Billing account to upgrade to the pay-as-you-go (Blaze) plan

Selected plan

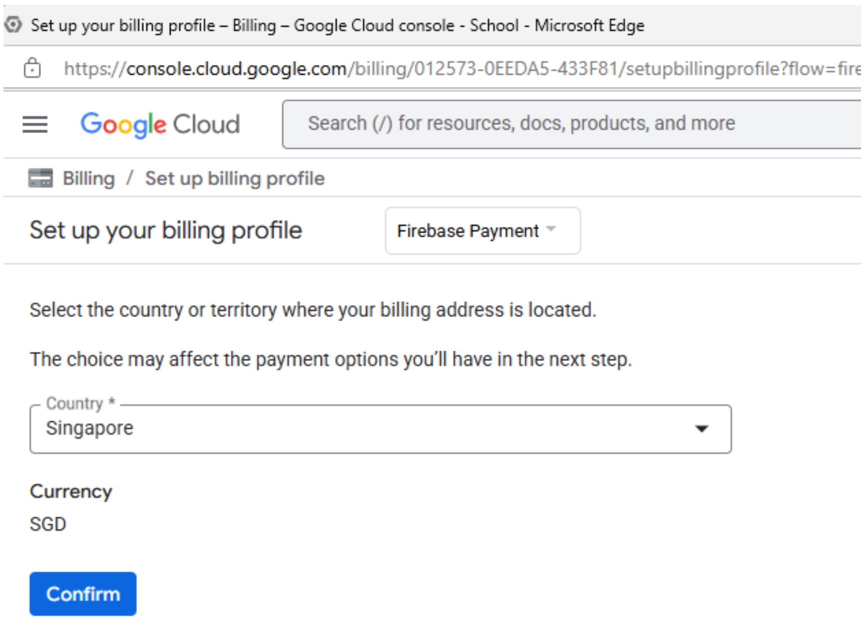
Pay as you go Blaze Plan See full plan details

Choose a Cloud Billing account

Available Cloud Billing accounts

You don't have a Cloud Billing Account
We'll help you create a billing account that you can link to your project

Create a Cloud Billing account



Set up your billing profile – Billing – Google Cloud console - School - Microsoft Edge

https://console.cloud.google.com/billing/012573-0EEDA5-433F81/setupbillingprofile?flow=fire

Google Cloud

Search (/) for resources, docs, products, and more

Billing / Set up billing profile

Set up your billing profile

Firebase Payment

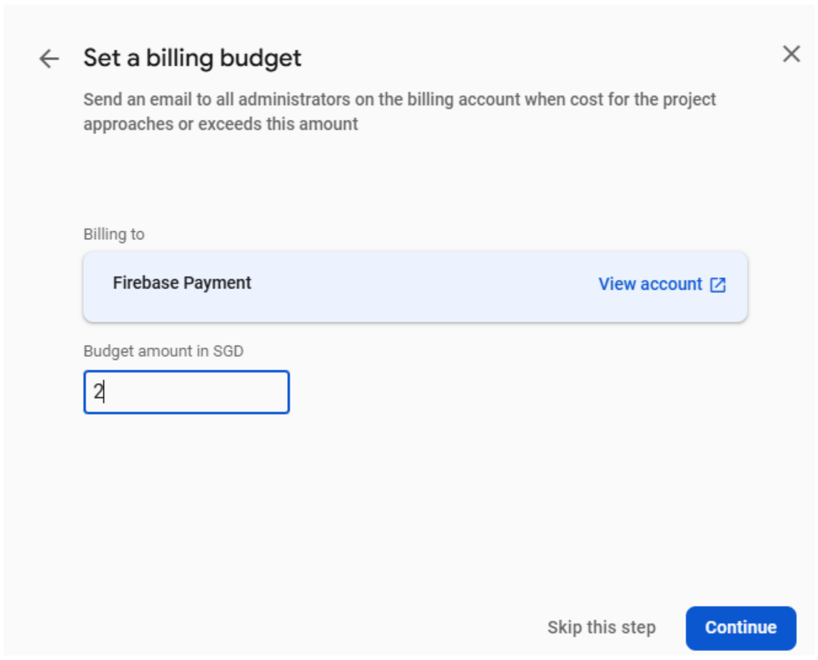
Select the country or territory where your billing address is located.

The choice may affect the payment options you'll have in the next step.

Country *
Singapore

Currency
SGD

Confirm



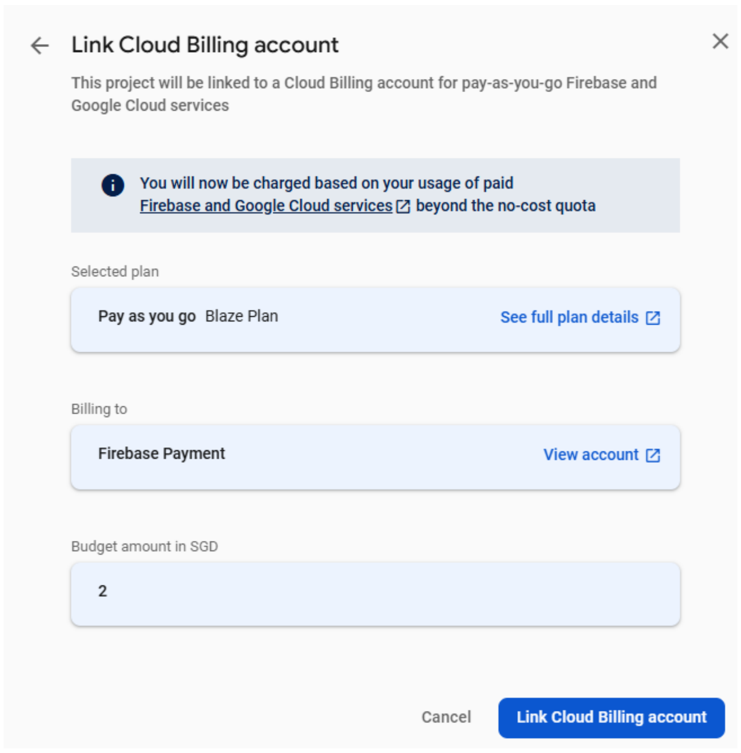
Set a billing budget

Send an email to all administrators on the billing account when cost for the project approaches or exceeds this amount

Billing to
Firebase Payment
View account

Budget amount in SGD
2

Skip this step
Continue


Link Cloud Billing account

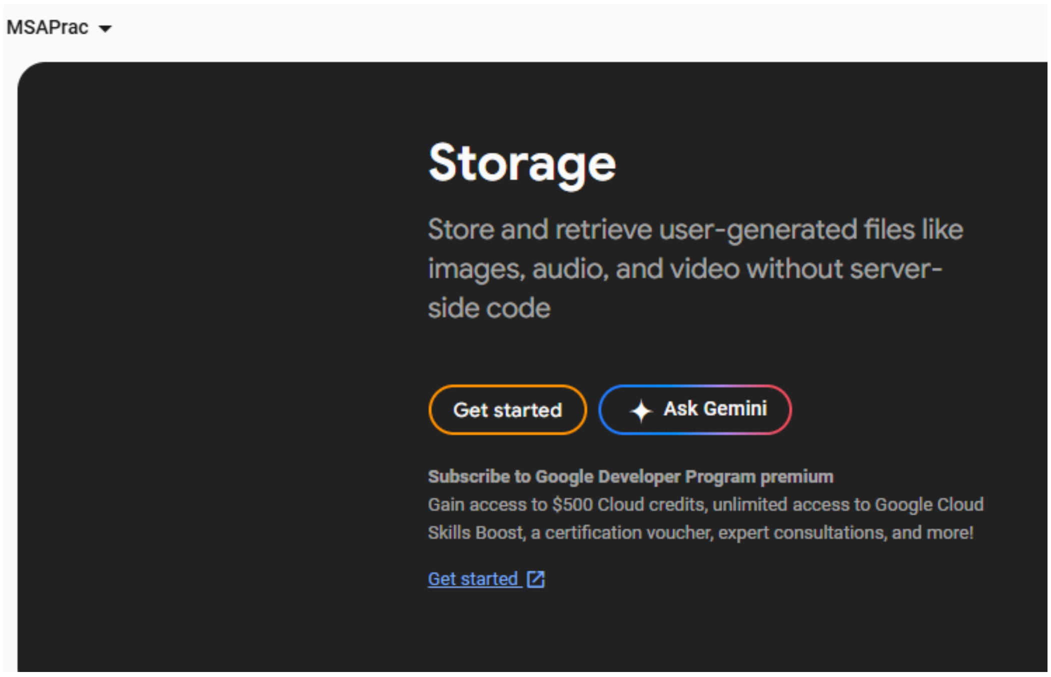
This project will be linked to a Cloud Billing account for pay-as-you-go Firebase and Google Cloud services

You will now be charged based on your usage of paid Firebase and Google Cloud services beyond the no-cost quota

Selected plan: Pay as you go Blaze Plan — See full plan details

Billing to: Firebase Payment — View account

Budget amount in SGD: 2


MSAPrac

Storage

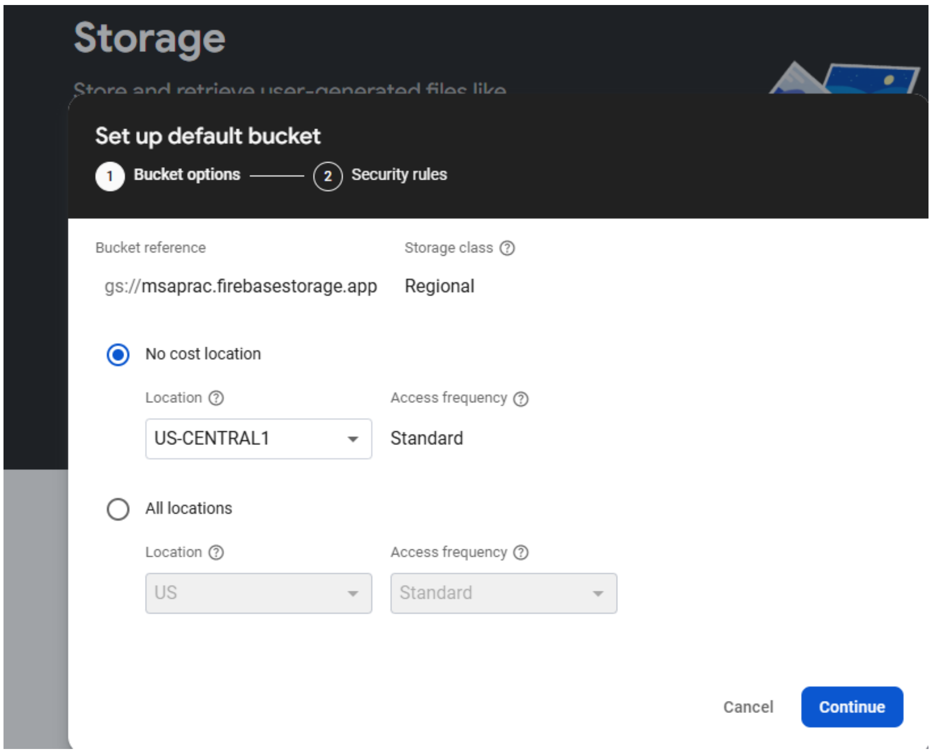
Store and retrieve user-generated files like images, audio, and video without server-side code

Subscribe to Google Developer Program premium

Gain access to $500 Cloud credits, unlimited access to Google Cloud Skills Boost, a certification voucher, expert consultations, and more!

7. **Get Started** on Storage

8. Select default *us-central1*. Select **Continue**.



Storage

Store and retrieve user-generated files like

Set up default bucket

1 Bucket options —— 2 Security rules

Bucket reference: gs://msaprac.firebasestorage.app

Storage class: Regional

[✓ No cost location]

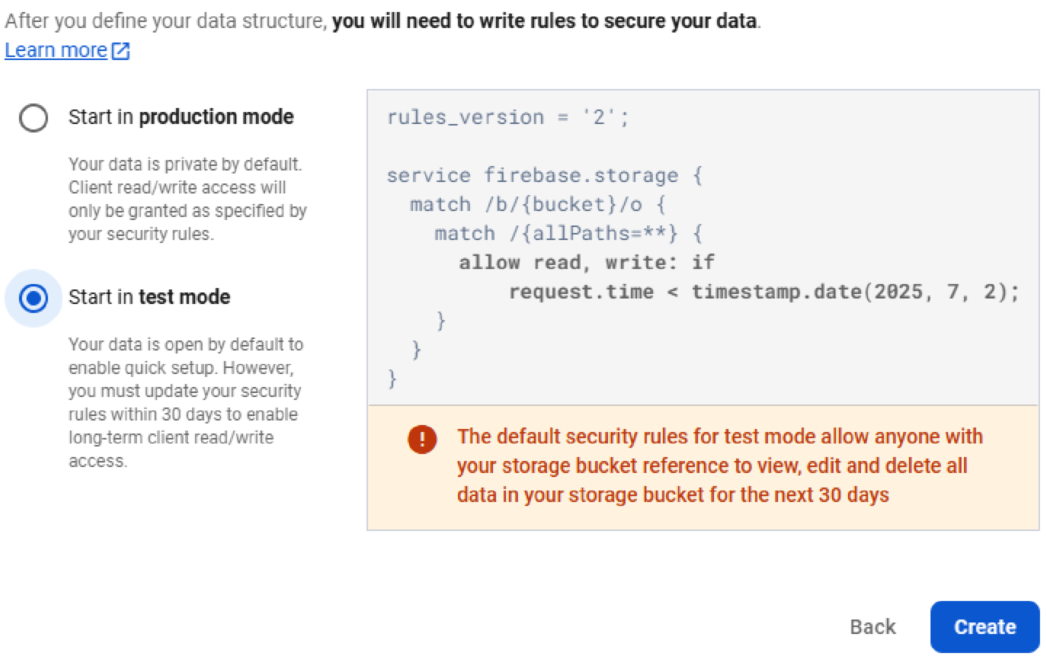
Location: US-CENTRAL1

Access frequency: Standard

All locations

Location: US

Access frequency: Standard



After you define your data structure, **you will need to write rules to secure your data**.
Learn more

Start in **production mode**

Your data is private by default. Client read/write access will only be granted as specified by your security rules.

[✓ Start in **test mode**]

Your data is open by default to enable quick setup. However, you must update your security rules within 30 days to enable long-term client read/write access.

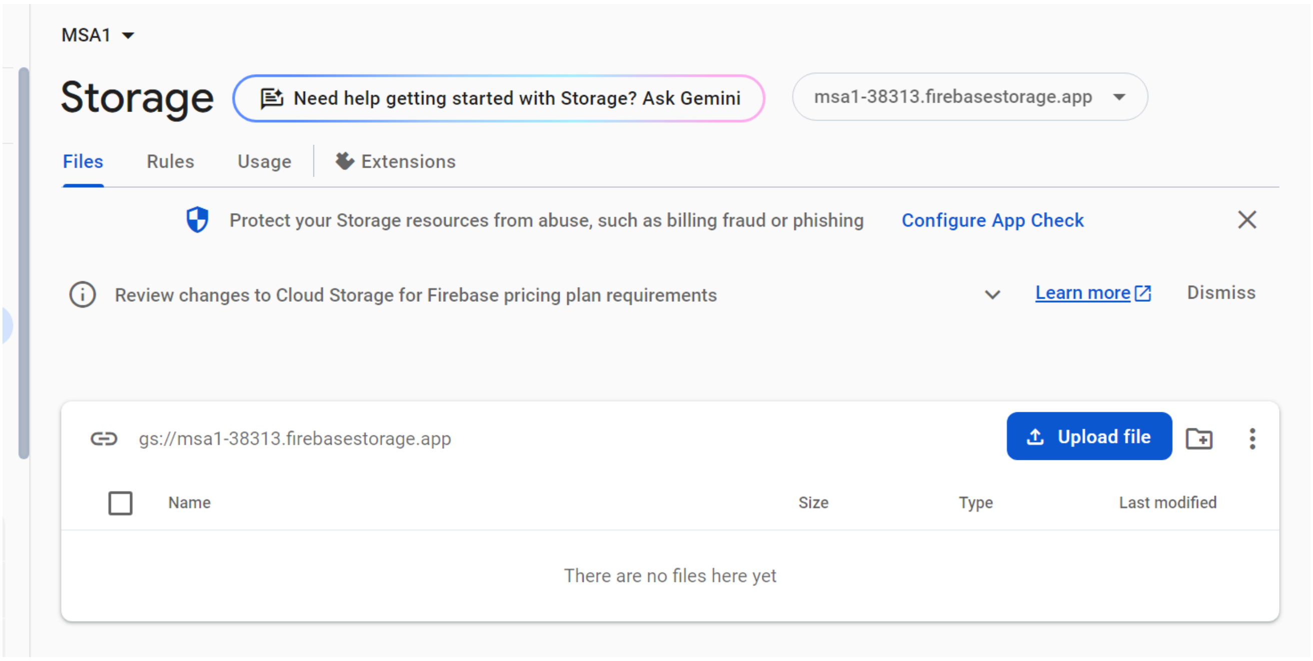
```
rules_version = '2';

service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      allow read, write: if
          request.time < timestamp.date(2025, 7, 2);
    }
  }
}
```

The default security rules for test mode allow anyone with your storage bucket reference to view, edit and delete all data in your storage bucket for the next 30 days

## 1.2 Add Images to Firebase Cloud Storage

You may download the image files required from **Politemall** or search for your own images online.

1. In Firebase console, select **Storage** and then the **Files** tab.
   Select **Upload file**.



MSA1

Storage

Need help getting started with Storage? Ask Gemini

msa1-38313.firebasestorage.app

Files Rules Usage Extensions

Protect your Storage resources from abuse, such as billing fraud or phishing Configure App Check

Review changes to Cloud Storage for Firebase pricing plan requirements Learn more Dismiss

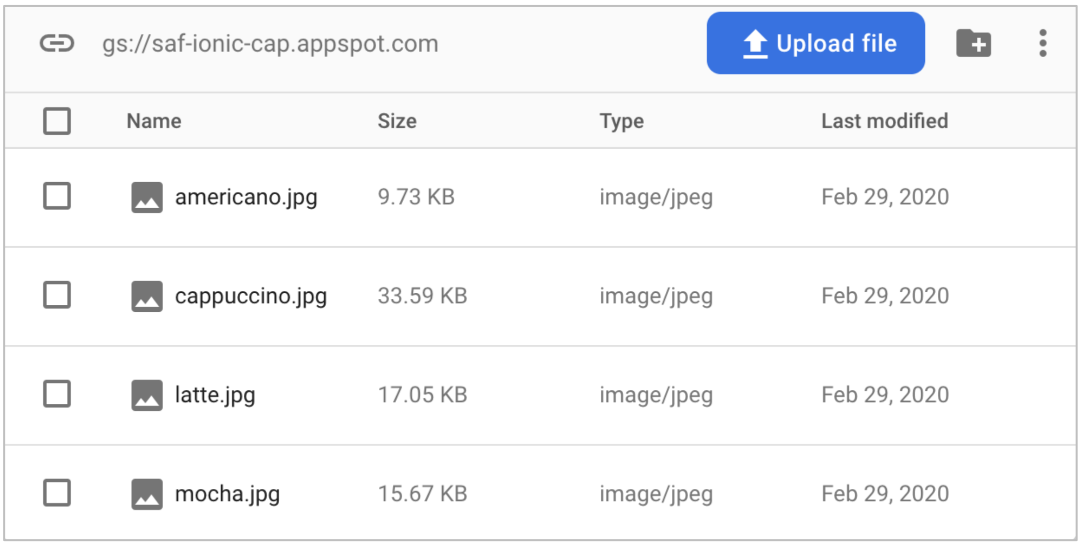
gs://msa1-38313.firebasestorage.app

Upload file

| Name | Size | Type | Last modified |
|---|---|---|---|

There are no files here yet

2. Add more files to the **Cloud Storage**.



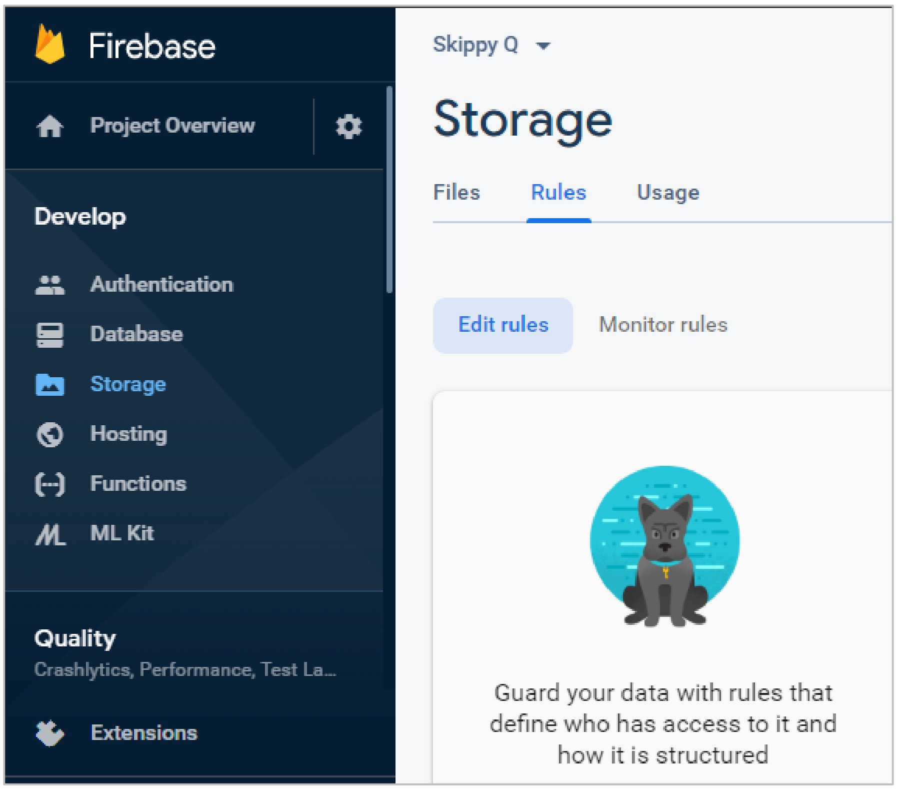
gs://saf-ionic-cap.appspot.com

Upload file

| Name | Size | Type | Last modified |
|---|---|---|---|
| americano.jpg | 9.73 KB | image/jpeg | Feb 29, 2020 |
| cappuccino.jpg | 33.59 KB | image/jpeg | Feb 29, 2020 |
| latte.jpg | 17.05 KB | image/jpeg | Feb 29, 2020 |
| mocha.jpg | 15.67 KB | image/jpeg | Feb 29, 2020 |

## 1.3 Configure Security Rules

1. In Firebase console, select **Storage** and then the **Rules** tab.

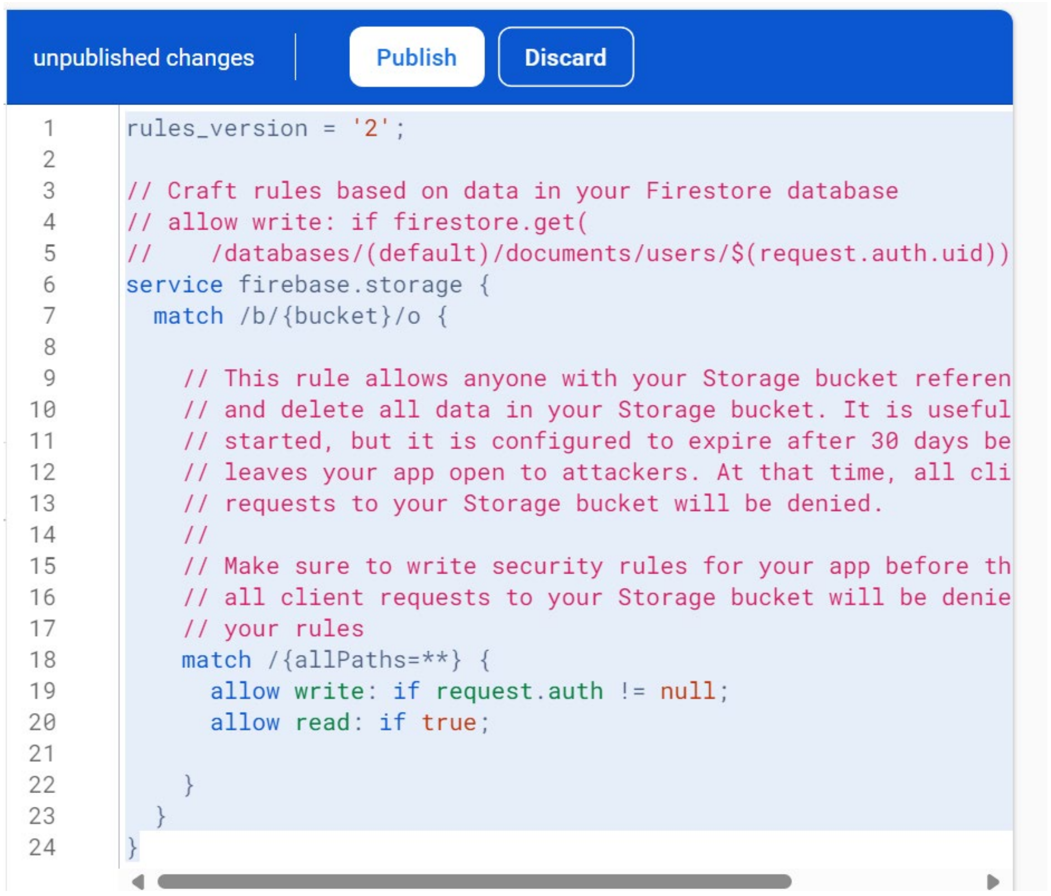


Guard your data with rules that define who has access to it and how it is structured

2. Type the following to allow anyone to read the Storage, but only authenticated users to write.

```
rules_version = '2';

// Craft rules based on data in your Firestore database
// allow write: if firestore.get(
//    /databases/(default)/documents/users/$(request.auth.uid)).data.isAdmin;
service firebase.storage {
  match /b/{bucket}/o {

    // This rule allows anyone with your Storage bucket reference to view,
edit,
    // and delete all data in your Storage bucket. It is useful for getting
    // started, but it is configured to expire after 30 days because it
    // leaves your app open to attackers. At that time, all client
    // requests to your Storage bucket will be denied.
    //
    // Make sure to write security rules for your app before that time, or else
    // all client requests to your Storage bucket will be denied until you
Update
    // your rules
    match /{allPaths=**} {
      allow write: if request.auth != null;
      allow read: if true;

    }
  }
}
```

Click **Publish**.

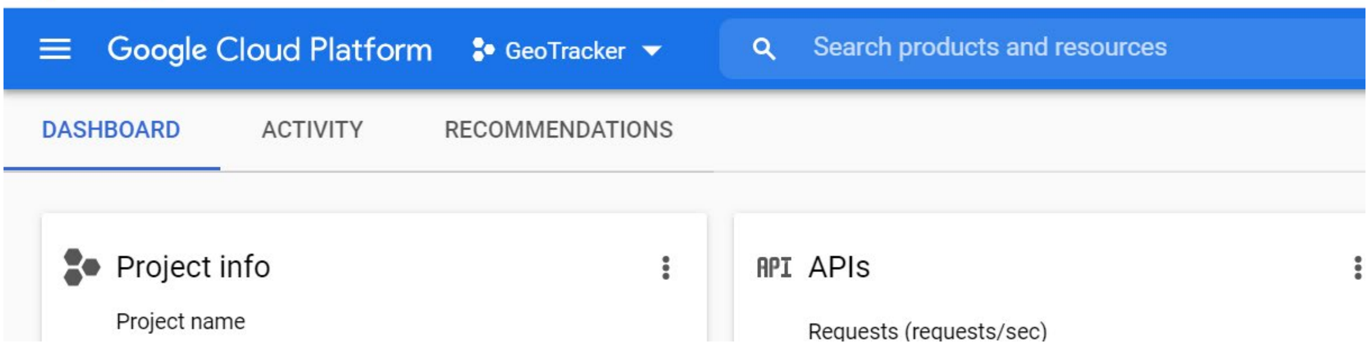


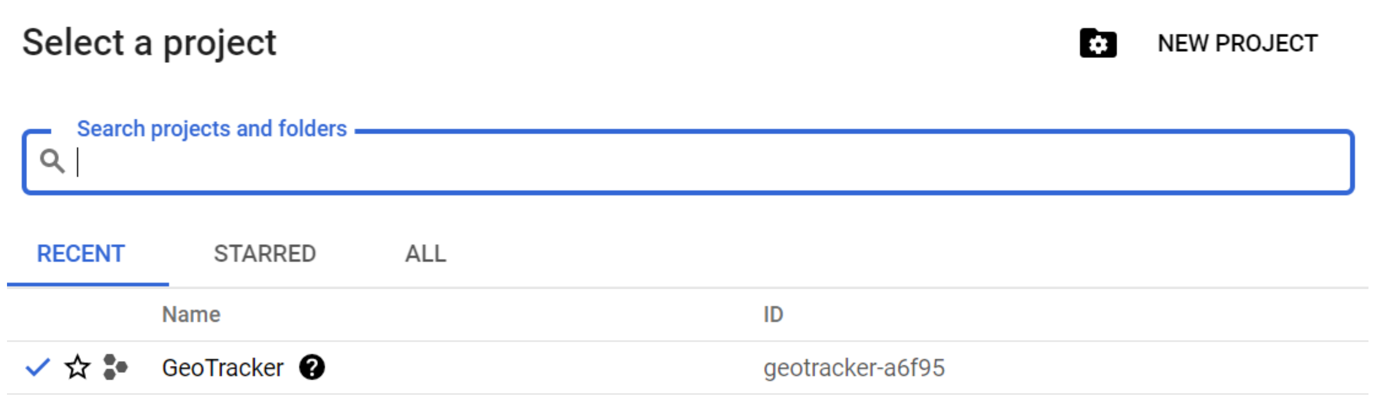
```
rules_version = '2';

// Craft rules based on data in your Firestore database
// allow write: if firestore.get(
//    /databases/(default)/documents/users/$(request.auth.uid))
service firebase.storage {
  match /b/{bucket}/o {

    // This rule allows anyone with your Storage bucket referen
    // and delete all data in your Storage bucket. It is useful
    // started, but it is configured to expire after 30 days be
    // leaves your app open to attackers. At that time, all cli
    // requests to your Storage bucket will be denied.
    //
    // Make sure to write security rules for your app before th
    // all client requests to your Storage bucket will be denie
    // your rules
    match /{allPaths=**} {
      allow write: if request.auth != null;
      allow read: if true;

    }
  }
}
```

## 1.4 *(Optional)* Enable Cross Origin Resource Sharing (CORS)

In order to use Firebase Storage in Ionic mobile app, you might need to enable CORS in your Firebase project.

1. Go to **Google Cloud Console** https://console.cloud.google.com/
2. Click on the drop down





Select a project

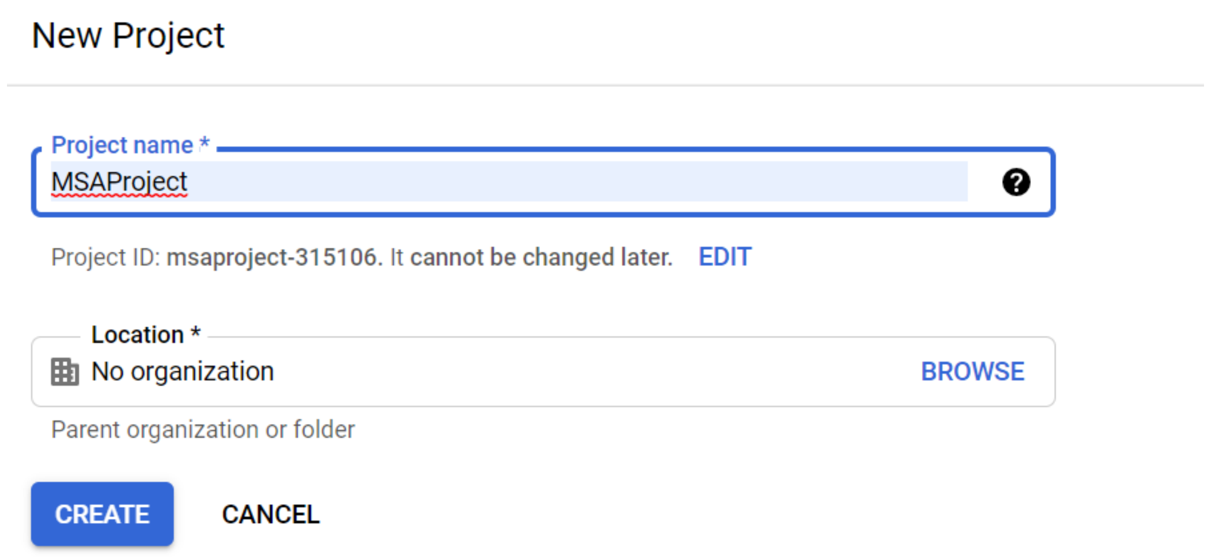
NEW PROJECT

Search projects and folders

RECENT STARRED ALL

| | Name | ID |
|---|---|---|
| ✓ | GeoTracker | geotracker-a6f95 |

3. Select New Project, Type project name and click Create



New Project

Project name * MSAProject

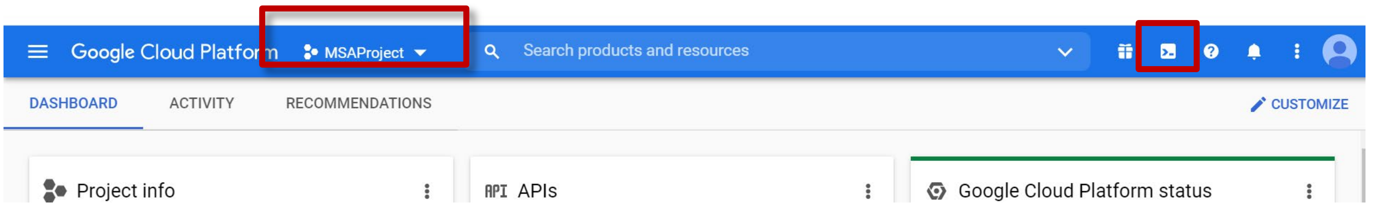
Project ID: msaproject-315106. It cannot be changed later. EDIT

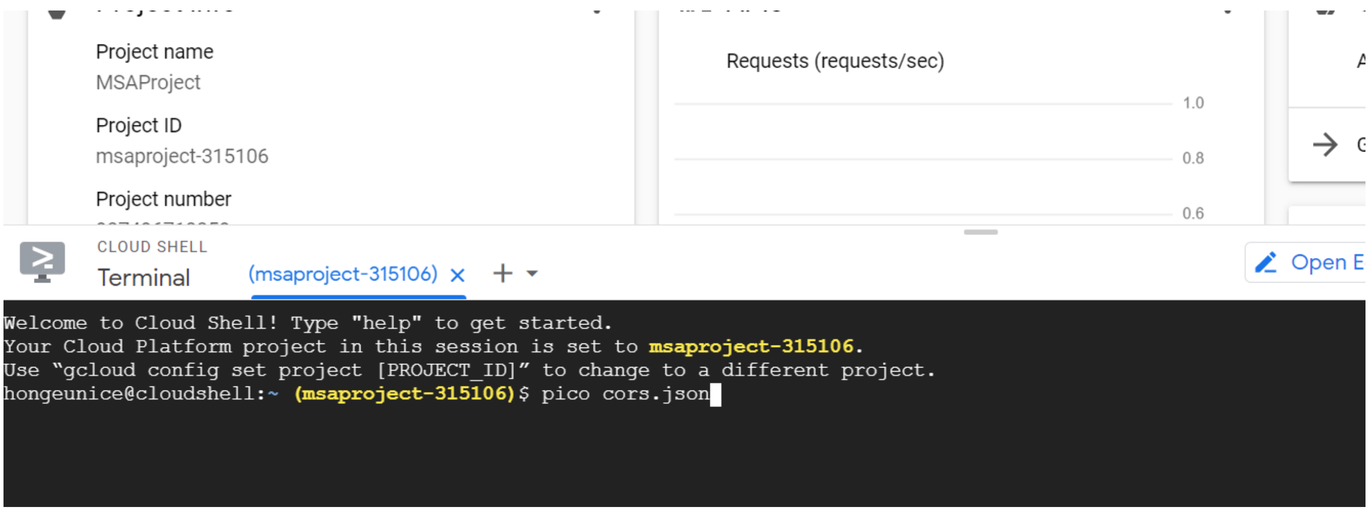
Location * No organization BROWSE

Parent organization or folder

CREATE CANCEL

4. Select the **MSAProject** project. Click on the [icon] icon to activate **Google Cloud Shell**.





```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to msaproject-315106.
Use “gcloud config set project [PROJECT_ID]” to change to a different project.
hongeunice@cloudshell:~ (msaproject-315106)$ pico cors.json
```

5. Type the following to use the **pico editor** to write `cors.json`.

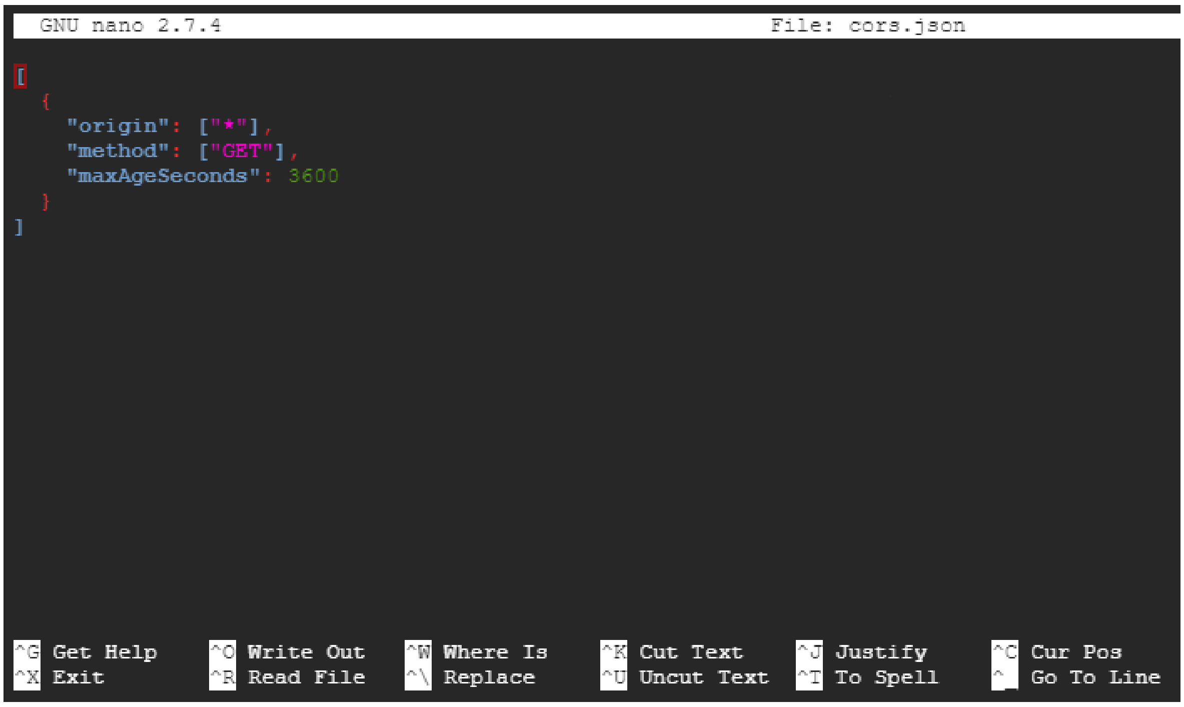
```
pico cors.json
```

6. In the **pico editor**, type the following.

cors.json

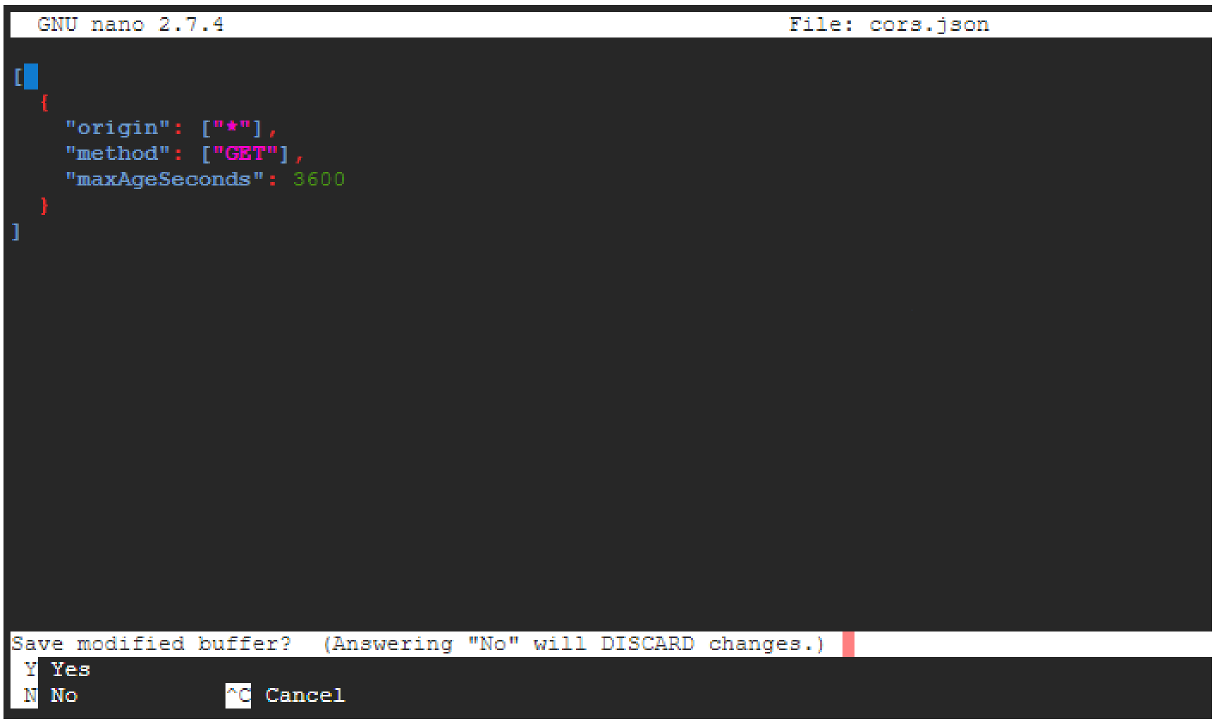
```
[
   {
        "origin": ["*"],
        "method": ["GET"],
        "maxAgeSeconds": 3600
    }
]
```

7. Using the keyboard, press `Ctrl X` to **Exit**.



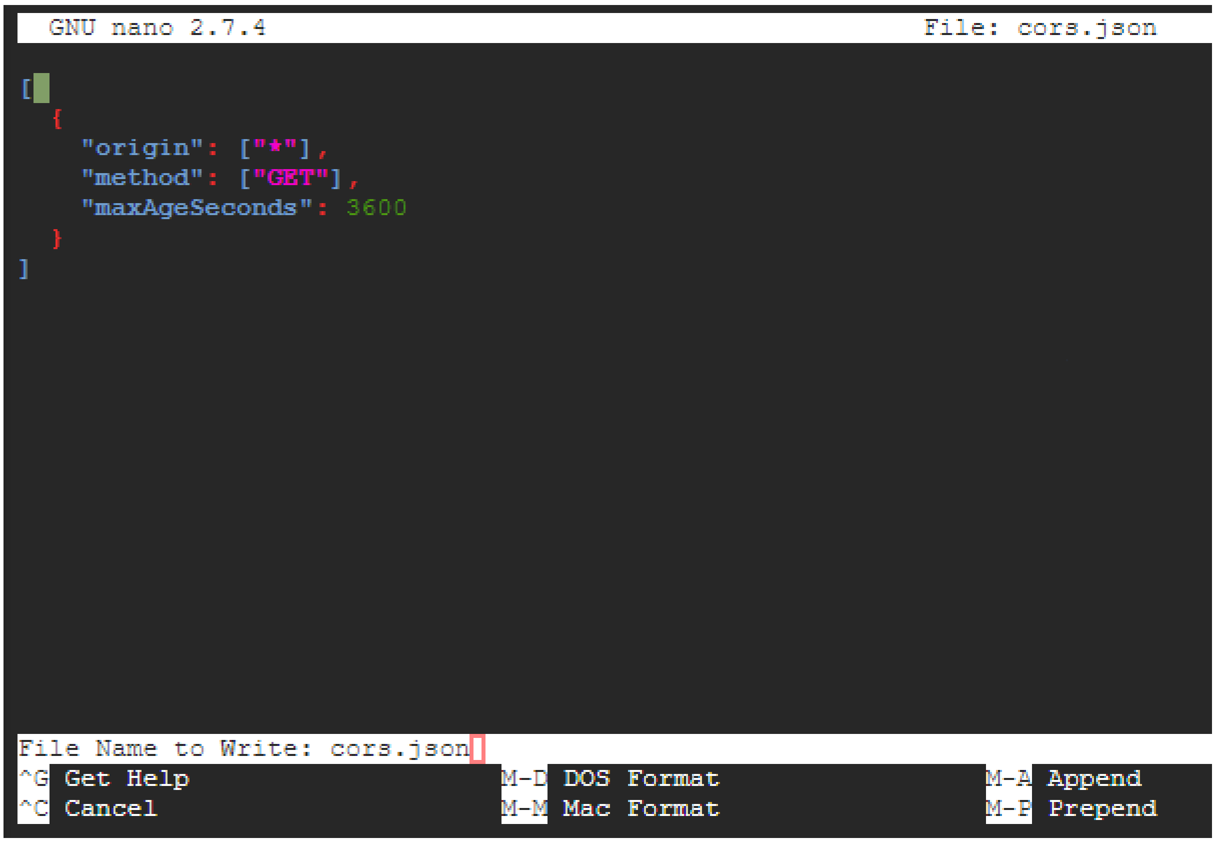
```
  GNU nano 2.7.4                                        File: cors.json

[
  {
    "origin": ["*"],
    "method": ["GET"],
    "maxAgeSeconds": 3600
  }
]

^G Get Help     ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify      ^C Cur Pos
^X Exit         ^R Read File    ^\ Replace      ^U Uncut Text   ^T To Spell     ^_ Go To Line
```

8. Type `Y` to save if you make changes.



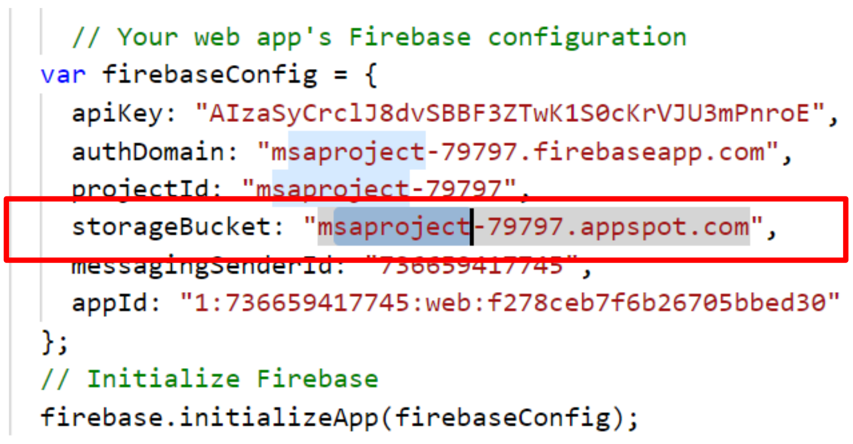
```
  GNU nano 2.7.4                                        File: cors.json

[
  {
    "origin": ["*"],
    "method": ["GET"],
    "maxAgeSeconds": 3600
  }
]

Save modified buffer?  (Answering "No" will DISCARD changes.)
 Y Yes
 N No           ^C Cancel
```

9. Press **Enter** to keep the file name as `cors.json`.



```
  GNU nano 2.7.4                                                File: cors.json

[
  {
    "origin": ["*"],
    "method": ["GET"],
    "maxAgeSeconds": 3600
  }
]

File Name to Write: cors.json
^G Get Help                         M-D DOS Format                      M-A Append
^C Cancel                           M-M Mac Format                      M-P Prepend
```

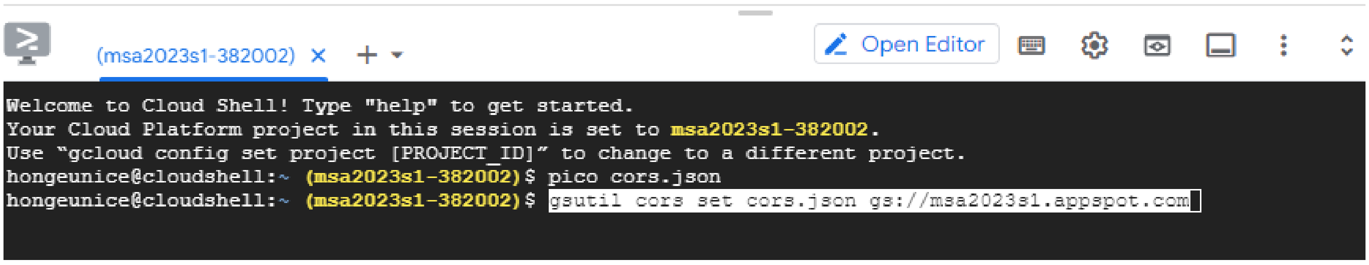
**10.** Refer to your previous firebase config



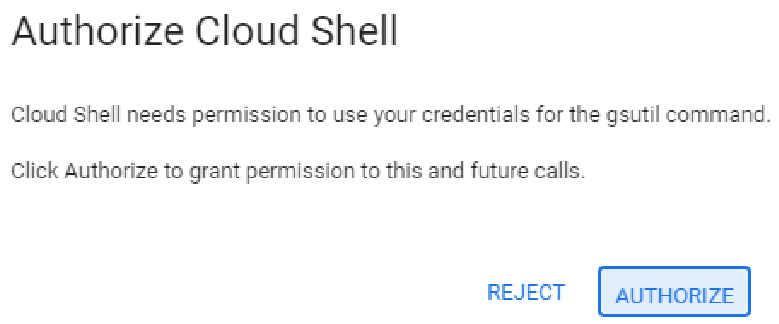
```
  // Your web app's Firebase configuration
var firebaseConfig = {
  apiKey: "AIzaSyCrclJ8dvSBBF3ZTwK1S0cKrVJU3mPnroE",
  authDomain: "msaproject-79797.firebaseapp.com",
  projectId: "msaproject-79797",
  storageBucket: "msaproject-79797.appspot.com",
  messagingSenderId: "736659417745",
  appId: "1:736659417745:web:f278ceb7f6b26705bbed30"
};
// Initialize Firebase
firebase.initializeApp(firebaseConfig);
```

**11.** In **Google Cloud Shell**, type the following by changing *gs://msaproject-79797.appspot.com* to your **Firebase Storage URL**.(Please change to reference your own id)

```
gsutil cors set cors.json gs://msaproject-79797.appspot.com
```



```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to msa2023s1-382002.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
hongeunice@cloudshell:~ (msa2023s1-382002)$ pico cors.json
hongeunice@cloudshell:~ (msa2023s1-382002)$ gsutil cors set cors.json gs://msa2023s1.appspot.com
```



## Authorize Cloud Shell

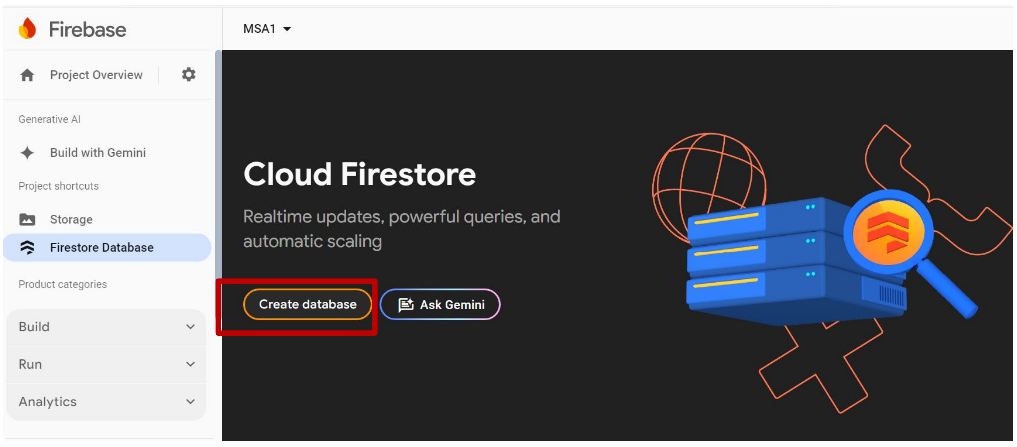
Cloud Shell needs permission to use your credentials for the gsutil command.

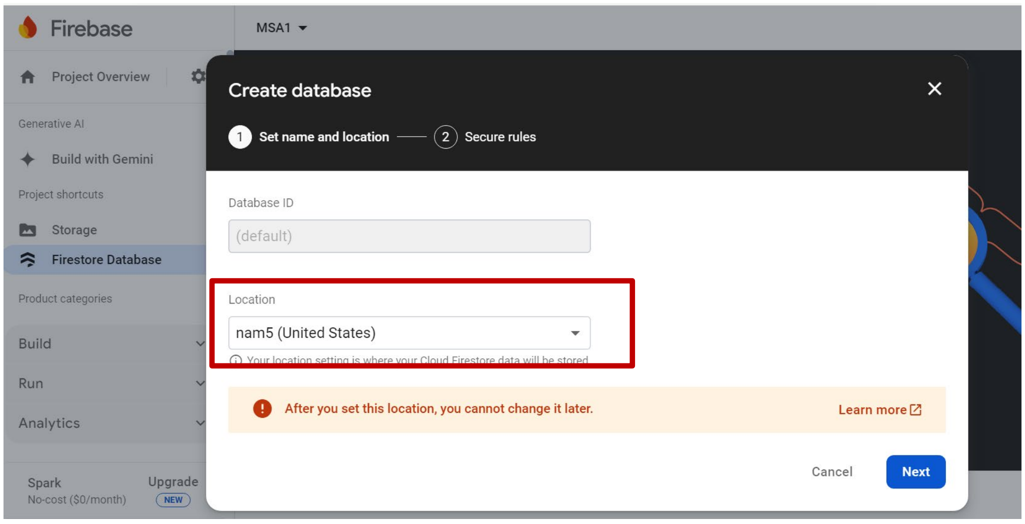
Click Authorize to grant permission to this and future calls.

# 2 Create Firebase Database

## 2.1 Enable Firebase Cloud Firestore

1. Go to https://firebase.google.com.
2. **Sign in** using your Google Account.
3. Select **Go to console**.
4. Select **Firestore Database**.





Create database

1 Set name and location —— 2 Secure rules

Database ID

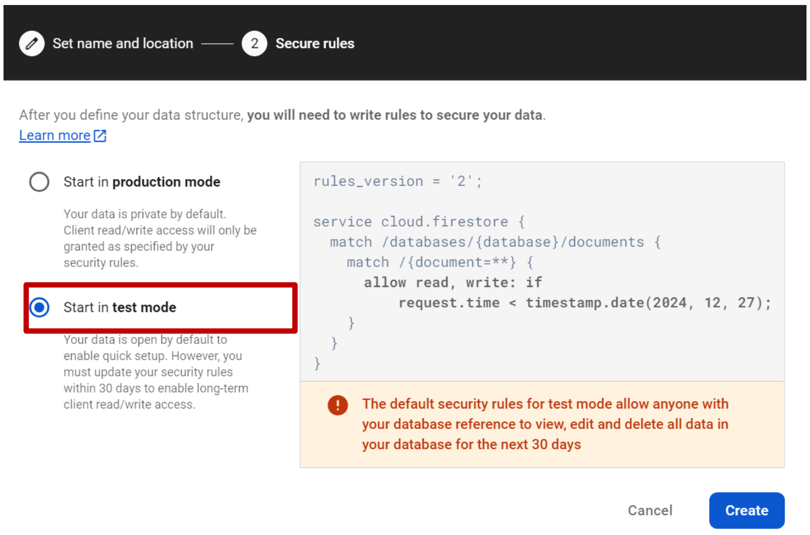
(default)

Location

nam5 (United States)

Your location setting is where your Cloud Firestore data will be stored

After you set this location, you cannot change it later.


Set name and location —— 2 Secure rules

After you define your data structure, **you will need to write rules to secure your data**.

Start in **production mode**

Your data is private by default. Client read/write access will only be granted as specified by your security rules.

Start in **test mode**

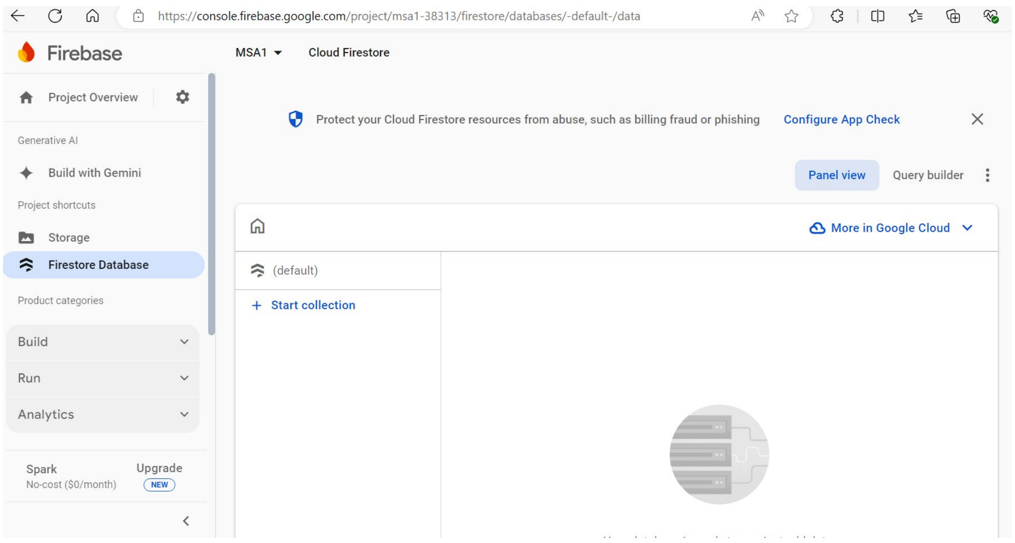
Your data is open by default to enable quick setup. However, you must update your security rules within 30 days to enable long-term client read/write access.

```
rules_version = '2';

service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if
          request.time < timestamp.date(2024, 12, 27);
    }
  }
}
```

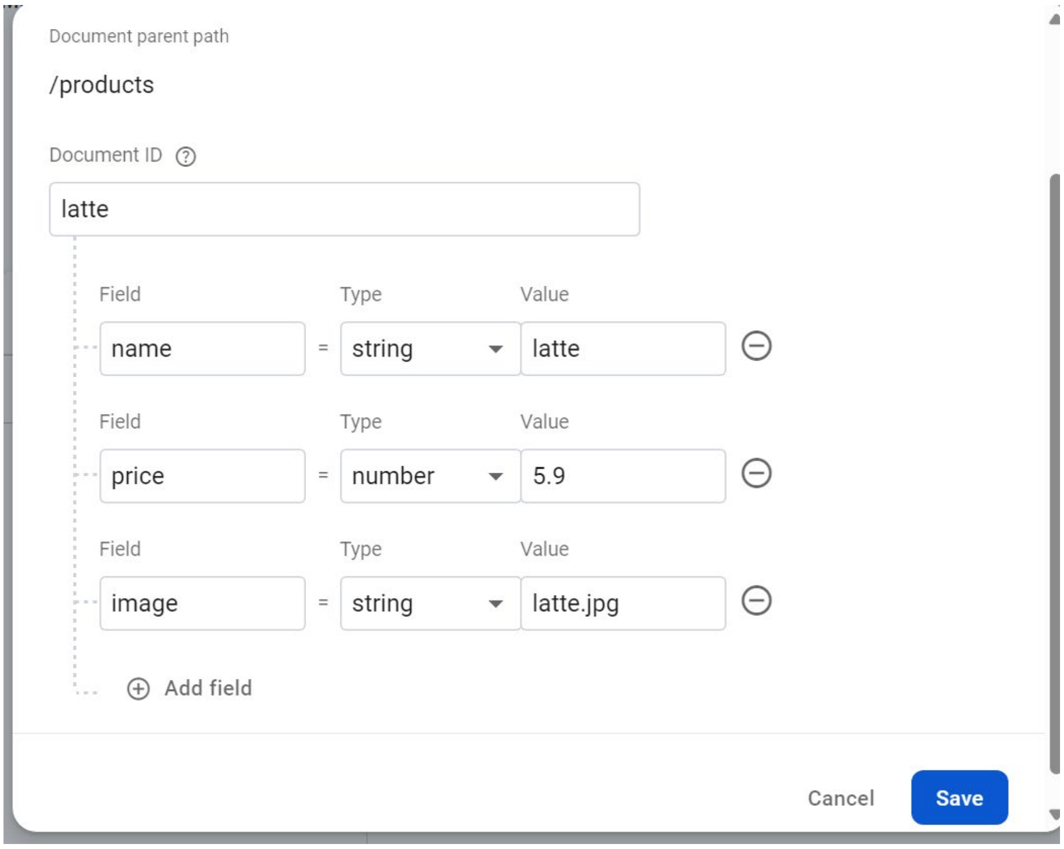
The default security rules for test mode allow anyone with your database reference to view, edit and delete all data in your database for the next 30 days

## 2.2 Add Data to Firebase

1. Select **+ Start collection**.



2. Create *'products'* collection and documents.Add the first document with document ID 'latte' and three data fields `name`, `price` and `image`.The image path must correspond to that added in Firebase Cloud Storage in the first task.
   Click **Save**.



Document parent path

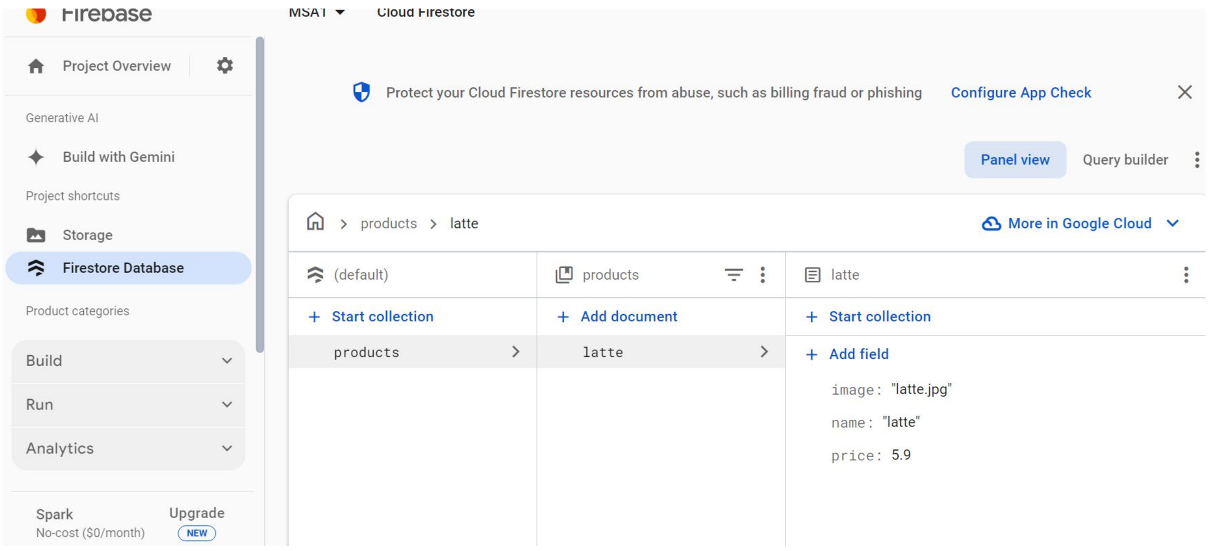
/products

Document ID

latte

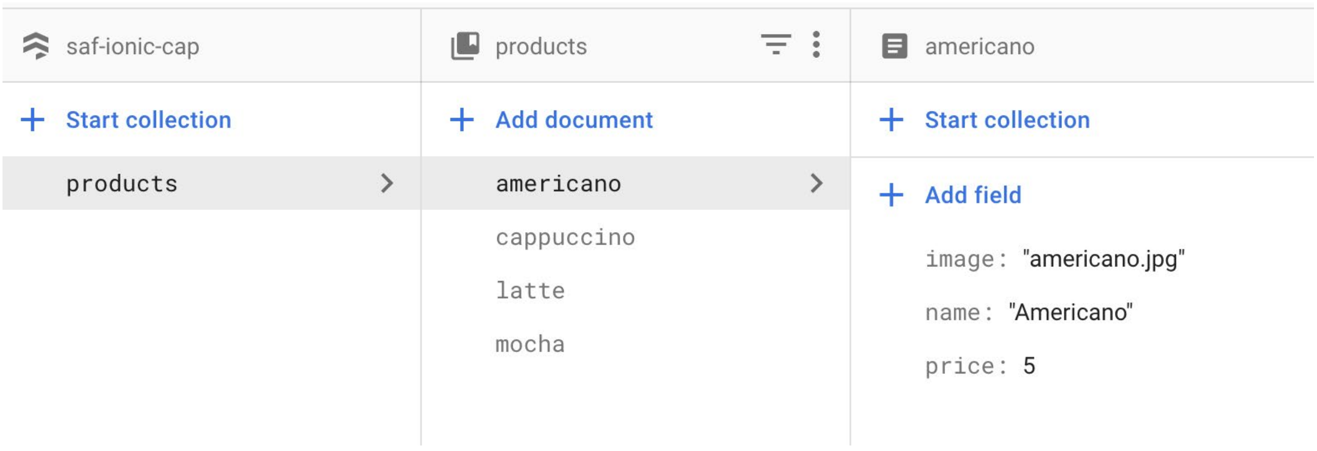
| Field | Type | Value |
|---|---|---|
| name | string | latte |
| price | number | 5.9 |
| image | string | latte.jpg |

3. You will see your document *'latte'* belonging to the *'products'* collection.



```
image: "latte.jpg"
name: "latte"
price: 5.9
```

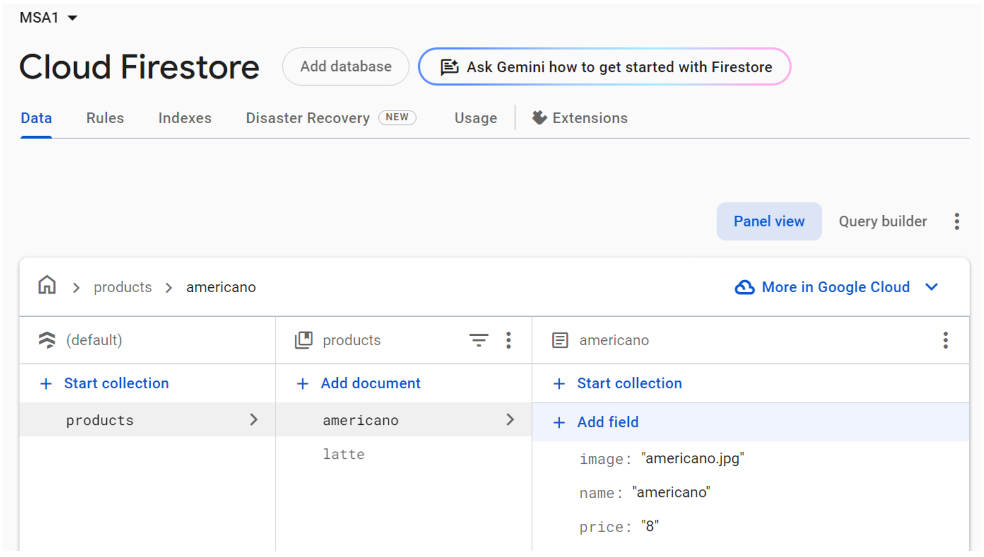
4. Create a few products.



```
image: "americano.jpg"
name: "Americano"
price: 5
```

## 2.3 Configure Firebase Database Security Rule

1. Next, we are going to create similar security rule for Firebase Database. Select the **Rules** tab.

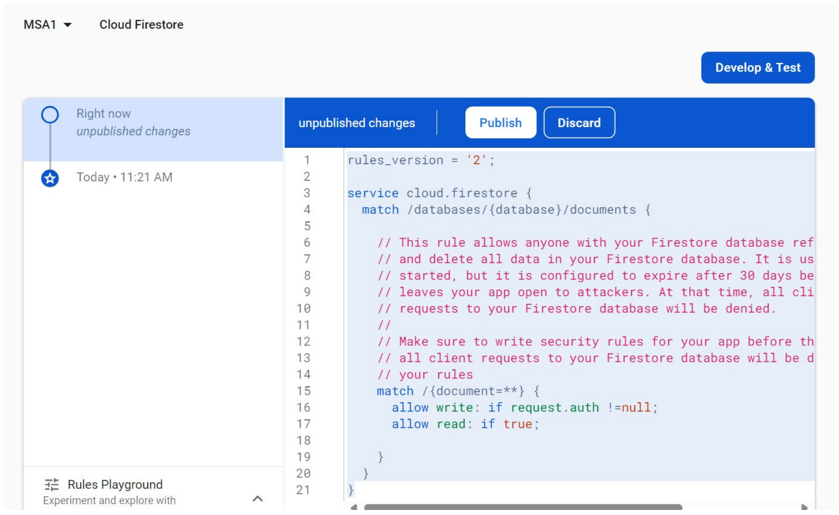


```
image: "americano.jpg"
name: "americano"
price: "8"
```

2. Type the following to allow anyone to read the Database, but only authenticated users to write.

```
rules_version = '2';

service cloud.firestore {
  match /databases/{database}/documents {

    // This rule allows anyone with your Firestore database
    reference to view, edit,
    // and delete all data in your Firestore database. It is
    useful for getting
    // started, but it is configured to expire after 30 days
    because it
    // leaves your app open to attackers. At that time, all
    client
    // requests to your Firestore database will be denied.
    //
    // Make sure to write security rules for your app before
    that time, or else
    // all client requests to your Firestore database will be
    denied until you Update
    // your rules
    match /{document=**} {
      allow write: if request.auth !=null;
      allow read: if true;

    }
  }
}
```

**3.** Click **Publish**.



```
rules_version = '2';

service cloud.firestore {
  match /databases/{database}/documents {

    // This rule allows anyone with your Firestore database ref
    // and delete all data in your Firestore database. It is us
    // started, but it is configured to expire after 30 days be
    // leaves your app open to attackers. At that time, all cli
    // requests to your Firestore database will be denied.
    //
    // Make sure to write security rules for your app before th
    // all client requests to your Firestore database will be d
    // your rules
    match /{document=**} {
      allow write: if request.auth !=null;
      allow read: if true;

    }
  }
}
```

*~ End ~*