Vente POS

Documentation

Web and Mobile Application

1. Introduction

Thank you for your interest in Vente POS.

This guide was implemented to help you set up this project successfully. For the process to go smoothly, it's essential to follow all the steps in this file.

Vente POS is an online point of sale system developed with Flutter at the front end and Firebase at the back end. Leveraging the power of Flutter, this application is available on the web, Android, and iOS. It's available in English and French, possibly adding even more languages.

POS Software allows your business to accept payments from customers and keep track of sales. Vente Point of Sale software also helps you handle orders and inventory, reach customers, and manage your team.

Vente POS uses Firebase as the backend service. Firebase, a product by Google, is a backend service that offers an online database, authentication service, storage, and much more.

This guide will walk you through how to link your Flutter project to Firebase.

2. Prerequisite

Some basic development knowledge will be needed in order to install Vente POS successfully. Here's what's needed:

a. IDE for mobile and web development, we recommend VSCode
b. Flutter SDK and JDK with path setup on your local machine
c. Basic knowledge of Google services and Firebase in particular.

3. Environment Setup

To run the project, some environment setups are needed:

a. You'll have to download and install Flutter on your system. You can check the documentation on the following link: **Install | Flutter**

b. You’ll need to create an account with Firebase by following this link: **Firebase | Google’s Mobile and Web App Development Platform**
c. You’ll have to download and install Node.js which you can do by following this link: **Node.js (nodejs.org)**
d. To set cors you’ll need to install **gsutil** from this link: **Install gsutil | Cloud Storage | Google Cloud**

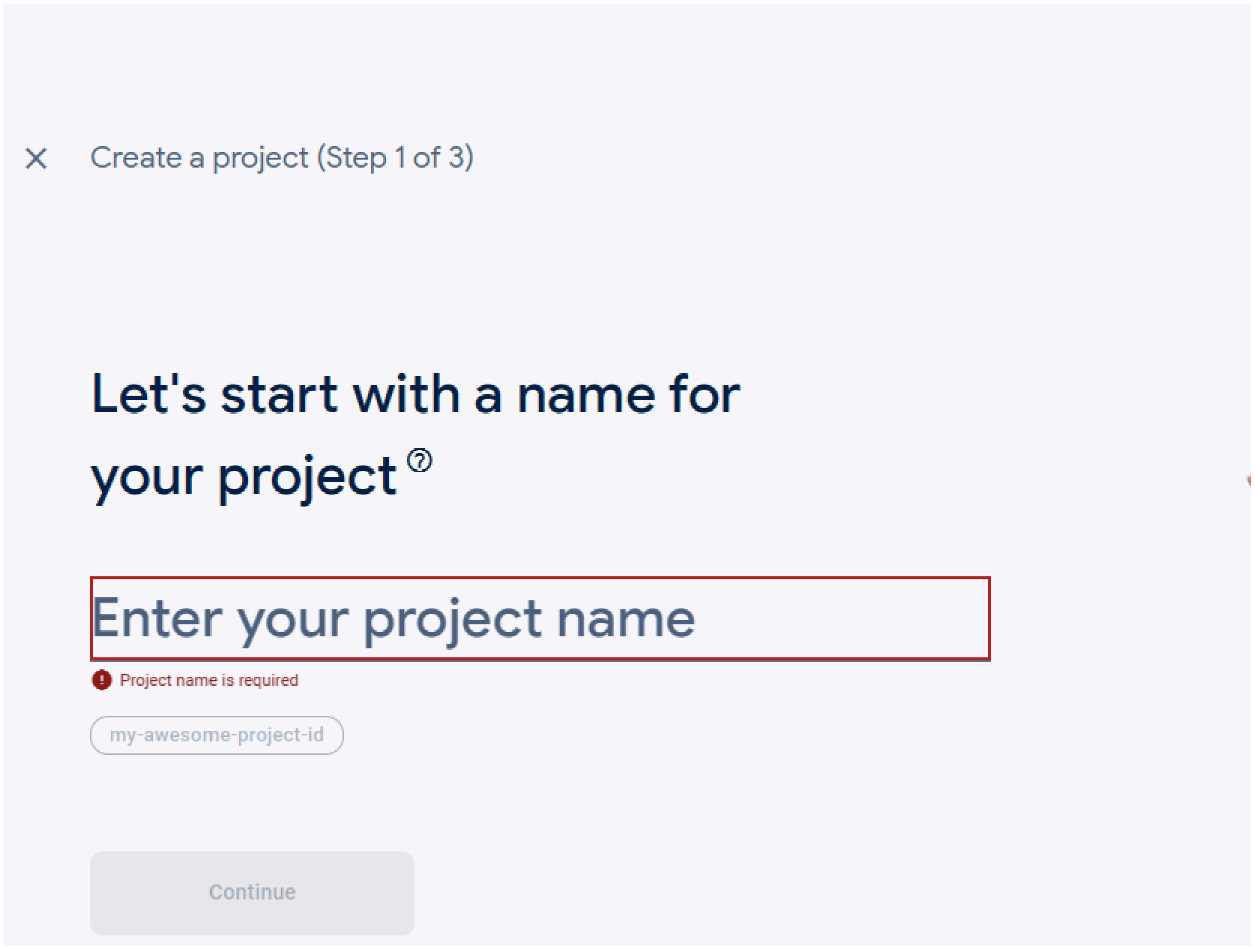
4. Basic Setup

a. Download the compressed file provided and unzip it on your machine.
b. Unzip vente file containing the Flutter project
c. Open vente folder in VS Code or Android Studio
d. Unzip vente_functions file which will be needed later

5. Firebase Setup

This project was configured to be linked with multiple Firebase projects. In general, you want to use different Firebase projects during development and production to avoid messing with your production data during the development process. Because of that, two flavors on the app with two different Firebase projects can be configured: Production, and Development flavors.
You don’t have to configure all of them and can just use the production flavor. The following steps focus on the production flavor, but the same steps can be repeated to add the development flavor. More info on this link: Multi-environment Flutter Projects with Flavors (sebastien-arbogast.com)

a. Go on and create a new Firebase project

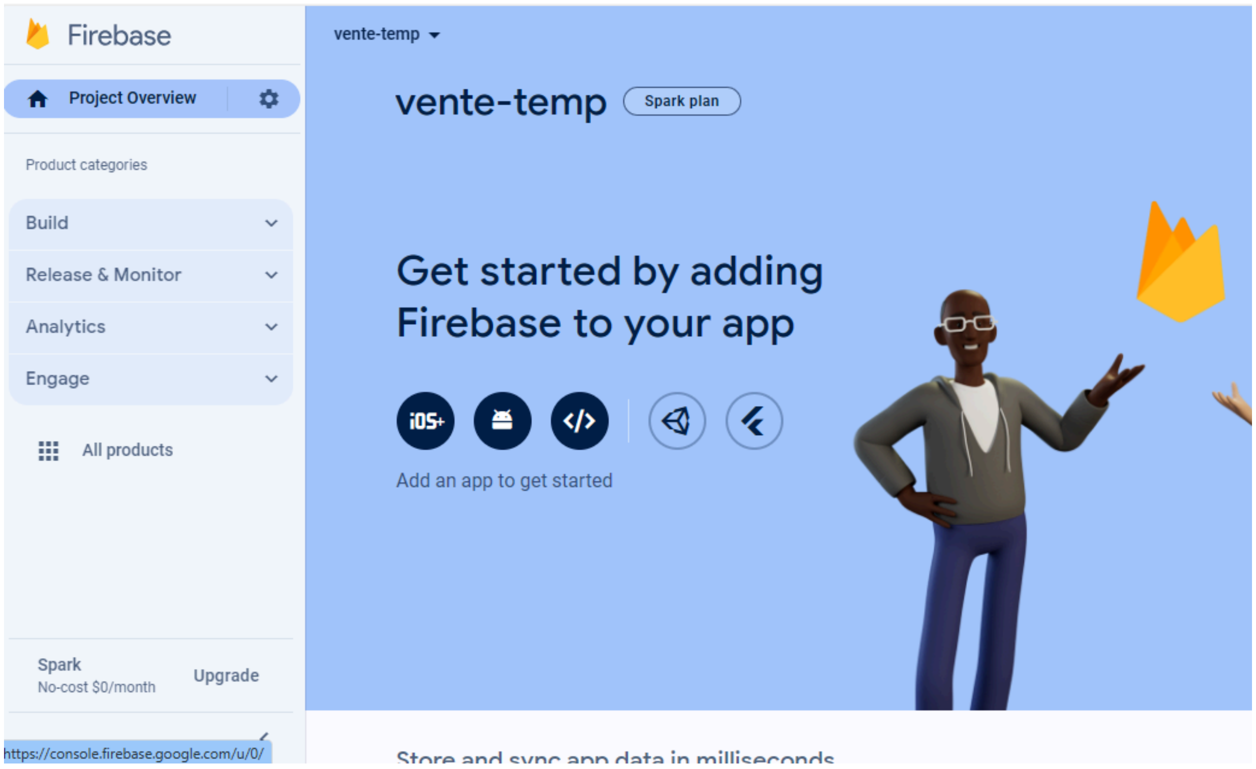

Create a project (Step 1 of 3)

Let's start with a name for your project

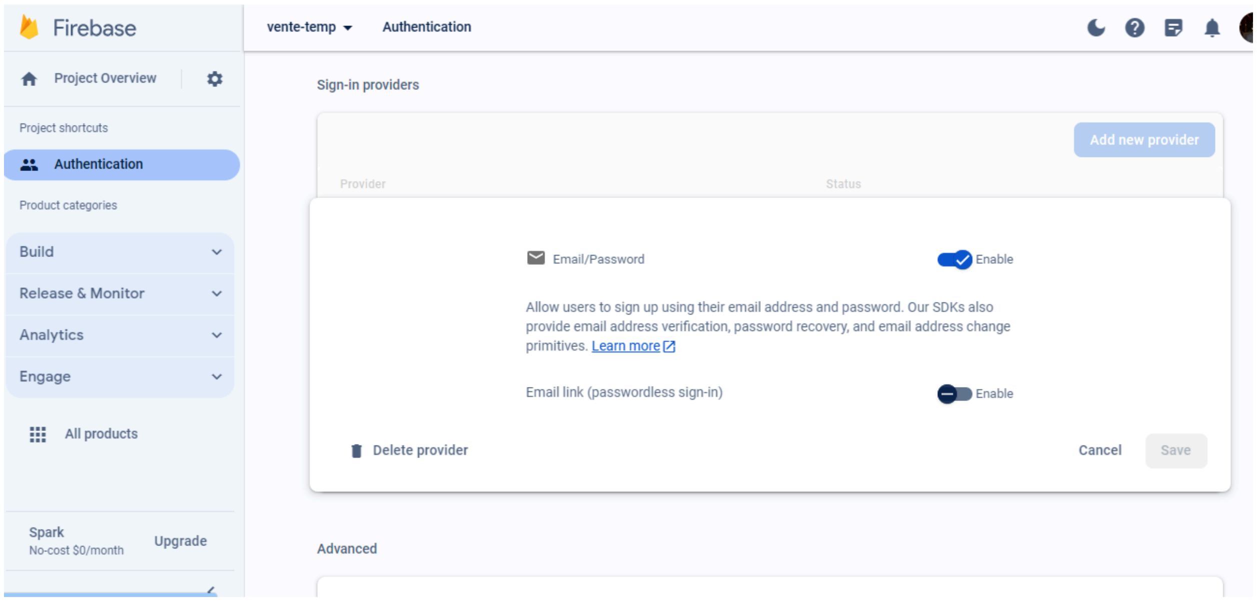
Enter your project name

Project name is required

b. Once created, You'll get to the overview of your Firebase project



c. Head to **Build > Authentication** to get started enable Email/Password authentication and click save

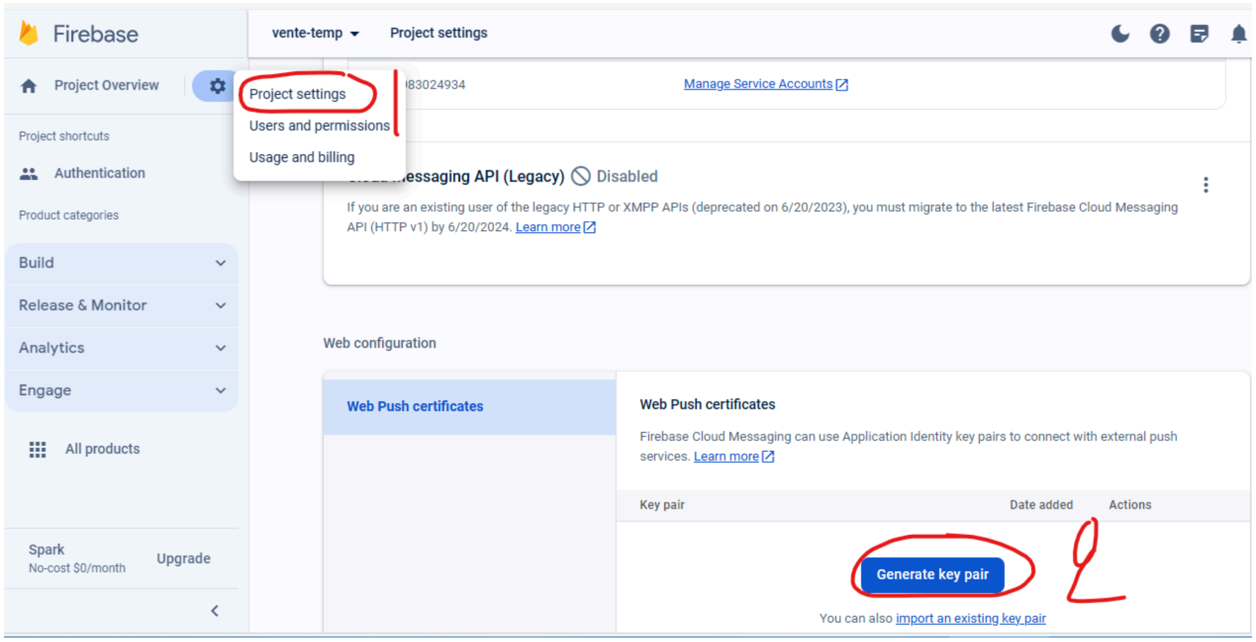


Email/Password — Enable: on

Allow users to sign up using their email address and password. Our SDKs also provide email address verification, password recovery, and email address change primitives. Learn more

Email link (passwordless sign-in) — Enable: off

d. Head to the project settings and click Generate key pair under cloud messaging**.** Copy the key pair generated.



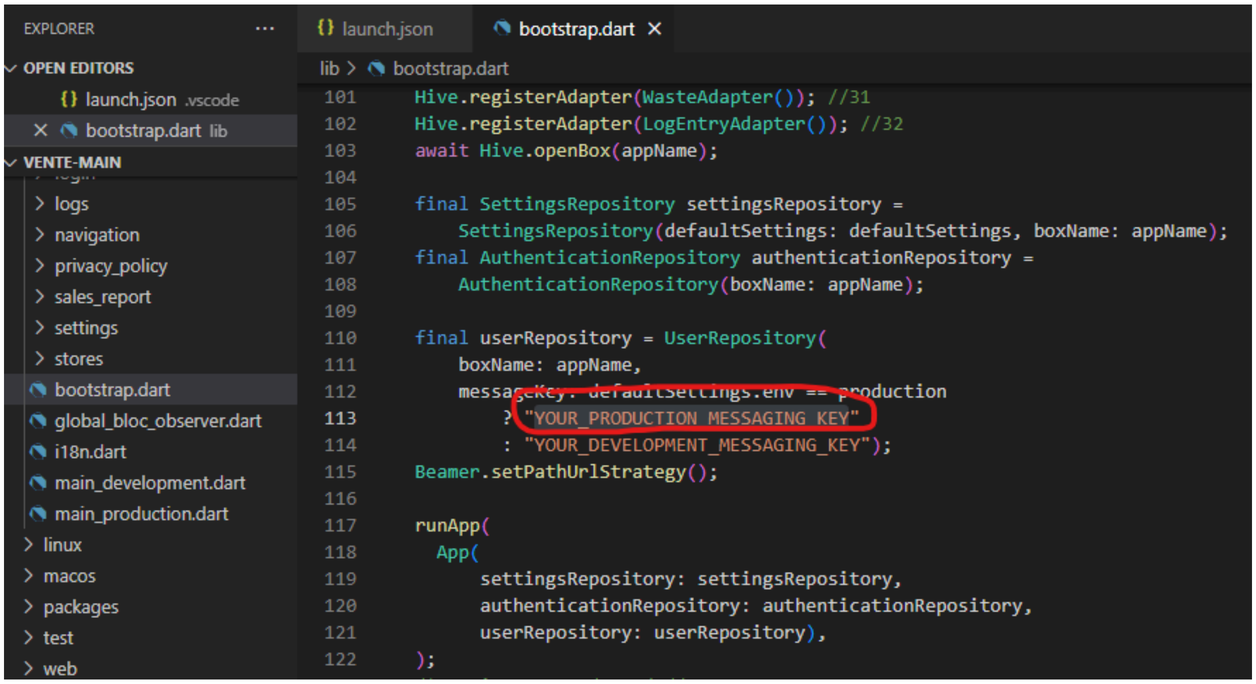
If you are an existing user of the legacy HTTP or XMPP APIs (deprecated on 6/20/2023), you must migrate to the latest Firebase Cloud Messaging API (HTTP v1) by 6/20/2024. Learn more

Web Push certificates

Firebase Cloud Messaging can use Application Identity key pairs to connect with external push services. Learn more

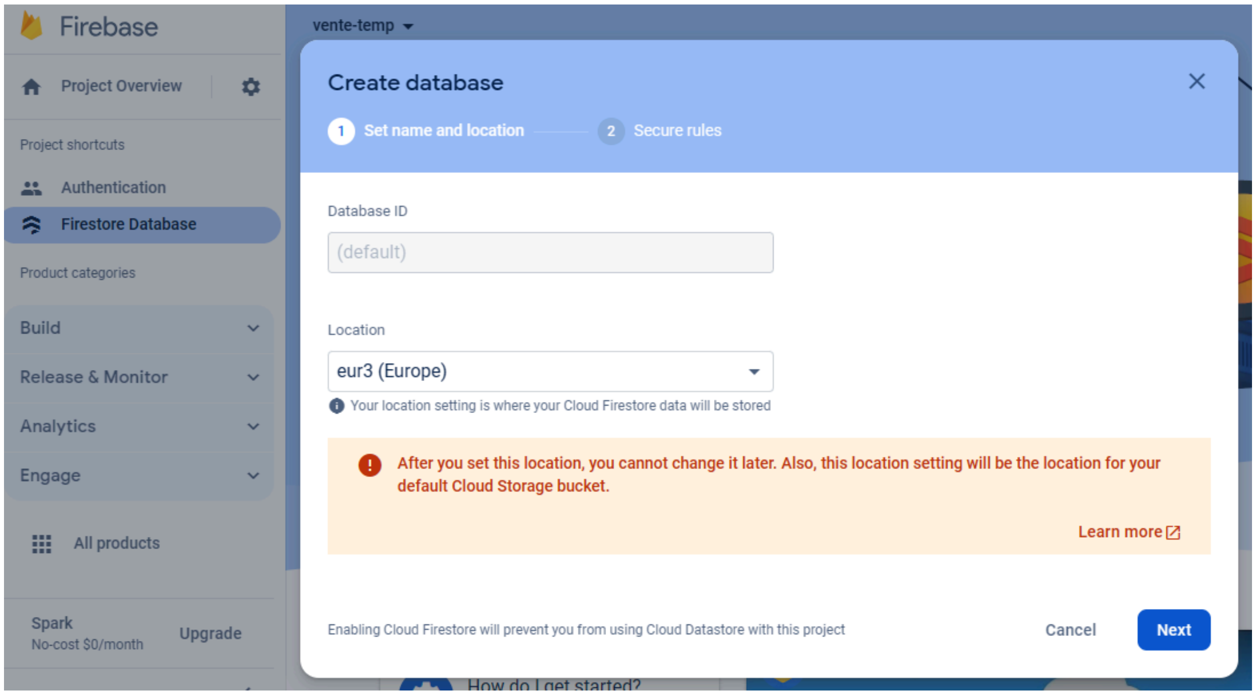
You can also import an existing key pair

e. Once copied, open **bootstrap.dart** in VS Code and paste the key pair in the **YOUR_PRODUCTION_MESSAGING_KEY** parameter. This key is needed in order to be able to send notifications to users.



```
Hive.registerAdapter(WasteAdapter()); //31
Hive.registerAdapter(LogEntryAdapter()); //32
await Hive.openBox(appName);

final SettingsRepository settingsRepository =
    SettingsRepository(defaultSettings: defaultSettings, boxName: appName);
final AuthenticationRepository authenticationRepository =
    AuthenticationRepository(boxName: appName);

final userRepository = UserRepository(
    boxName: appName,
    messagingKey: defaultSettings.env == production
        ? "YOUR_PRODUCTION_MESSAGING_KEY"
        : "YOUR_DEVELOPMENT_MESSAGING_KEY");
Beamer.setPathUrlStrategy();

runApp(
  App(
      settingsRepository: settingsRepository,
      authenticationRepository: authenticationRepository,
      userRepository: userRepository),
);
```

f. Head to **Build > Firestore Database** create your database in production mode and choose your database location.



Create database

1 Set name and location — 2 Secure rules

Database ID

(default)

Location

eur3 (Europe)

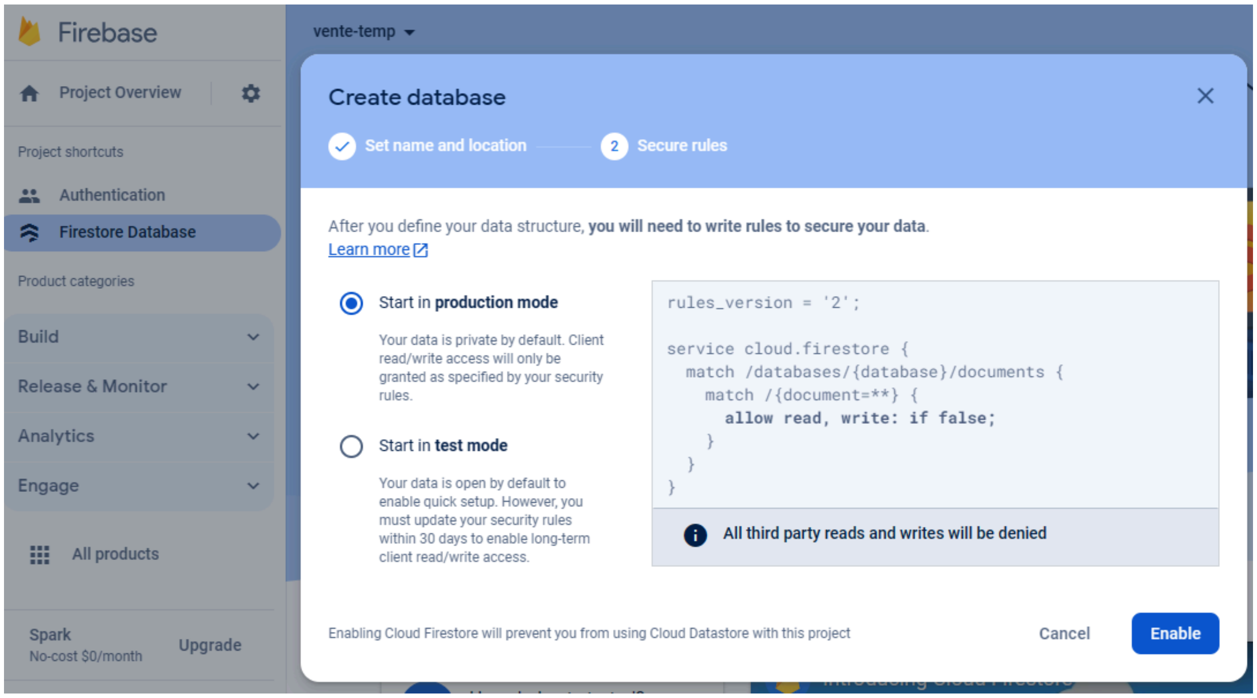
Your location setting is where your Cloud Firestore data will be stored

After you set this location, you cannot change it later. Also, this location setting will be the location for your default Cloud Storage bucket.

Learn more

Enabling Cloud Firestore will prevent you from using Cloud Datastore with this project

Cancel Next



Create database

✓ Set name and location — 2 Secure rules

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 cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if false;
    }
  }
}
```

All third party reads and writes will be denied

Enabling Cloud Firestore will prevent you from using Cloud Datastore with this project

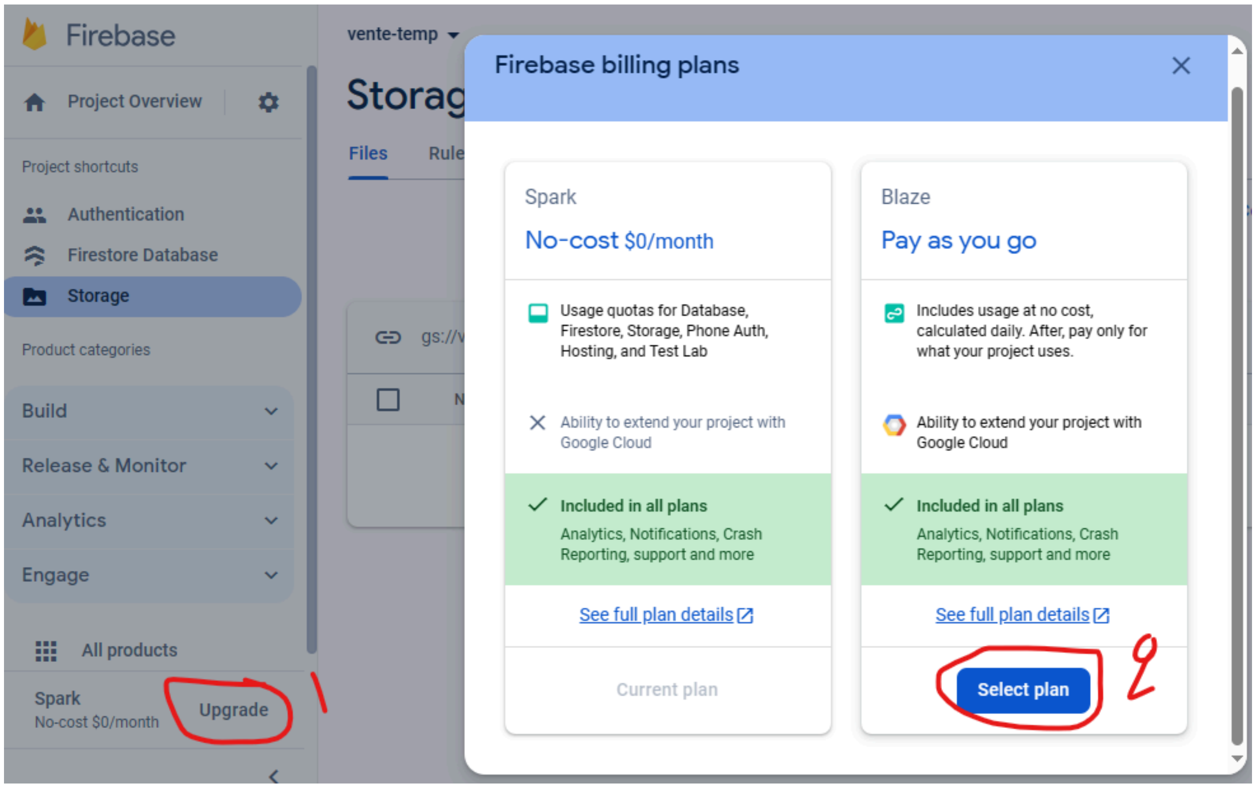
Cancel Enable

g. Head to **Build > Storage** and click Get Started. Start in production mode and set up your storage location.

6. Firebase Functions Setup

Now we will configure Firebase functions, Firestore rules and indexes, and Storage rules. This is optional but will ensure database indexes and some Firebase cloud functions are set up. Unzip the downloaded file from Envato and unzip the **vente_functions** file. Open the decompressed folder **vente_functions** in VSCode.

a. Upgrade your Firebase project to the Blaze plan. This is required in order to use cloud functions



Usage quotas for Database, Firestore, Storage, Phone Auth, Hosting, and Test Lab

Ability to extend your project with Google Cloud

Included in all plans
Analytics, Notifications, Crash Reporting, support and more

Includes usage at no cost, calculated daily. After, pay only for what your project uses.

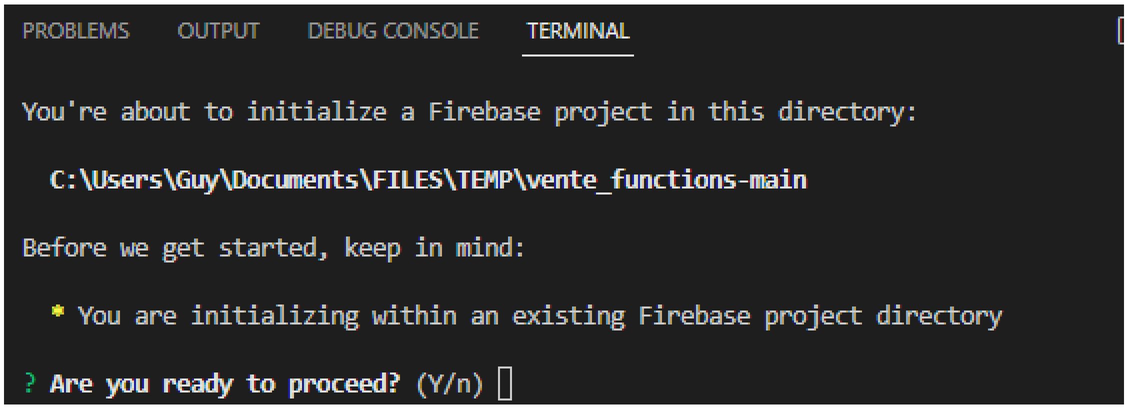
Ability to extend your project with Google Cloud

Included in all plans
Analytics, Notifications, Crash Reporting, support and more

b. In the **vente_functions** project terminal run **npm install -g firebase-tools**

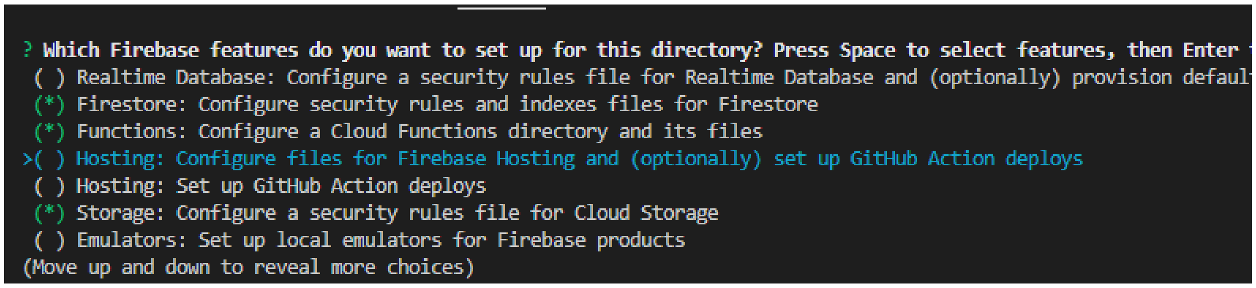
c. Then run **firebase login** to login to your account

d. Run **firebase init**. Type Y when it asks you to override the previous project



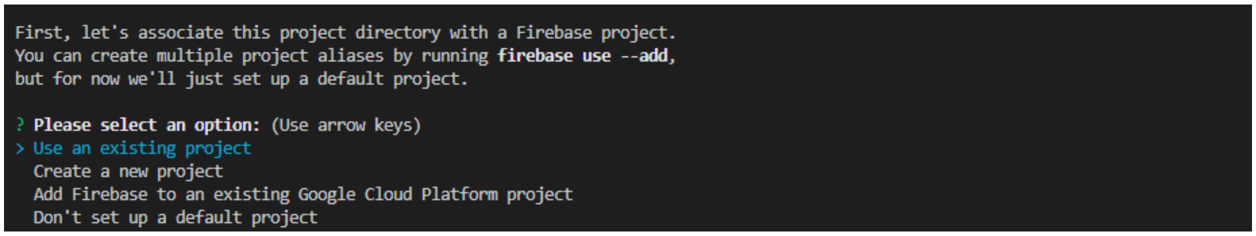
```
You're about to initialize a Firebase project in this directory:

  C:\Users\Guy\Documents\FILES\TEMP\vente_functions-main

Before we get started, keep in mind:

  * You are initializing within an existing Firebase project directory

? Are you ready to proceed? (Y/n)
```

e. On the list of features to configure, select, **firestore, functions, and storage** using the space key



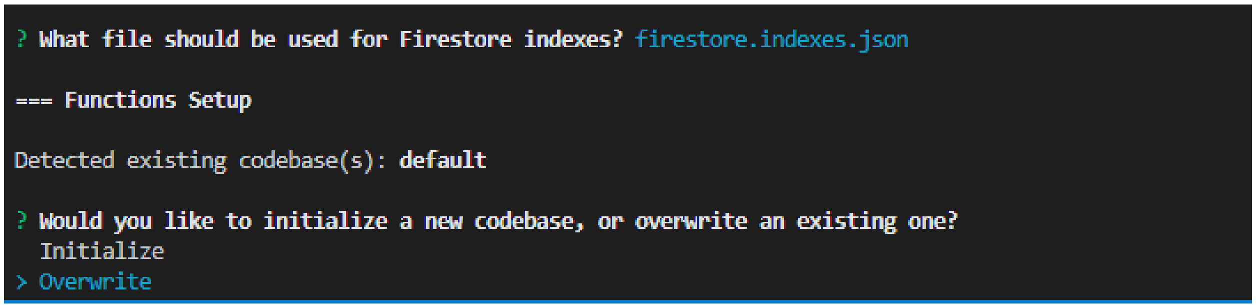
```
? Which Firebase features do you want to set up for this directory? Press Space to select features, then Enter
 ( ) Realtime Database: Configure a security rules file for Realtime Database and (optionally) provision defaul
 (*) Firestore: Configure security rules and indexes files for Firestore
 (*) Functions: Configure a Cloud Functions directory and its files
>( ) Hosting: Configure files for Firebase Hosting and (optionally) set up GitHub Action deploys
 ( ) Hosting: Set up GitHub Action deploys
 (*) Storage: Configure a security rules file for Cloud Storage
 ( ) Emulators: Set up local emulators for Firebase products
(Move up and down to reveal more choices)
```

f. In the next step select **Use an existing project** and choose the project we previously created in Firebase

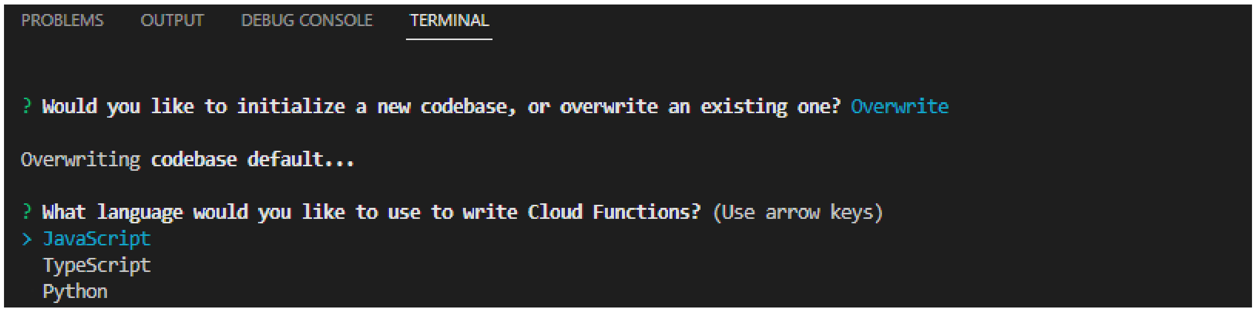


```
First, let's associate this project directory with a Firebase project.
You can create multiple project aliases by running firebase use --add,
but for now we'll just set up a default project.

? Please select an option: (Use arrow keys)
> Use an existing project
  Create a new project
  Add Firebase to an existing Google Cloud Platform project
  Don't set up a default project
```

g. When asked what to call Firestore rules, just press enter
h. When asked what to call Firestore indexes, just press enter
i. When asked to override any of the files, select no and continue
j. When asked whether to initialize a new codebase or override it, select **Overwrite**, and press enter


```
? What file should be used for Firestore indexes? firestore.indexes.json

=== Functions Setup

Detected existing codebase(s): default

? Would you like to initialize a new codebase, or overwrite an existing one?
  Initialize
> Overwrite
```

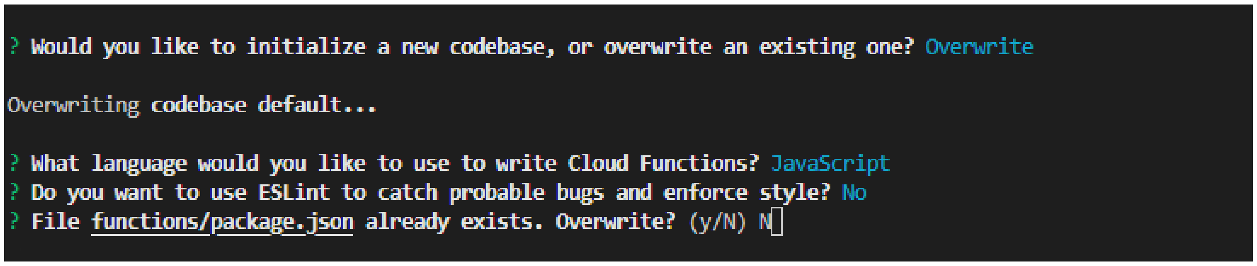
k. Choose javascript as the language for cloud functions and press enter



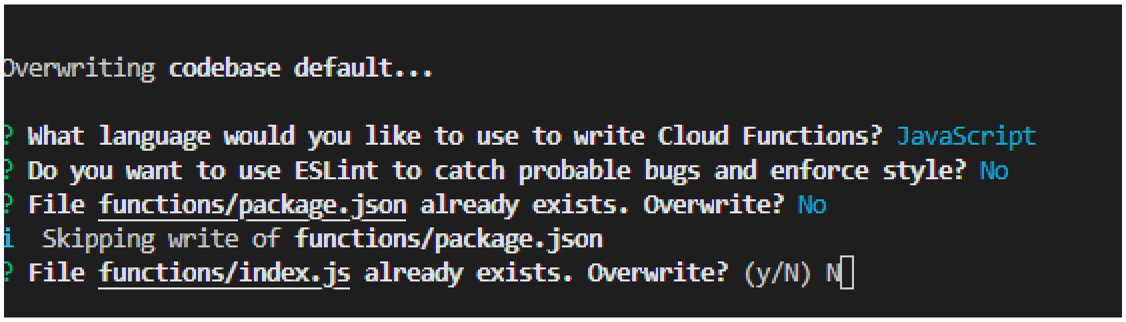
```
? Would you like to initialize a new codebase, or overwrite an existing one? Overwrite

Overwriting codebase default...

? What language would you like to use to write Cloud Functions? (Use arrow keys)
> JavaScript
  TypeScript
  Python
```

l. Just press enter when it asks to use ESLint

m. When asked to override **functions/package.json**, Enter N and press enter

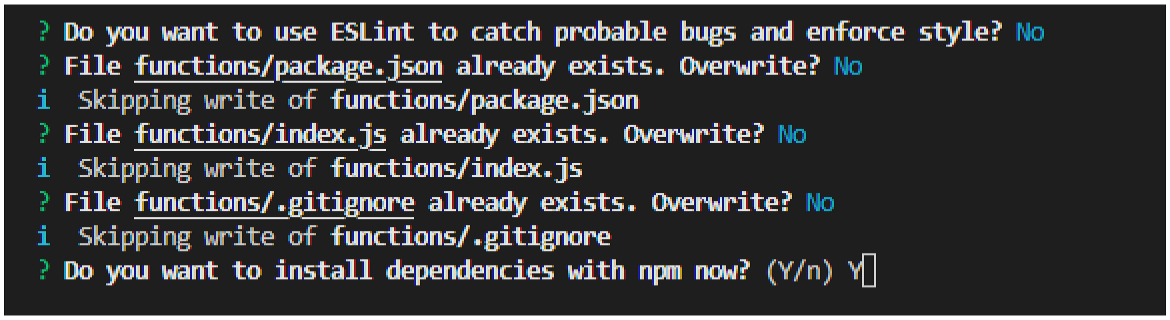

```
? Would you like to initialize a new codebase, or overwrite an existing one? Overwrite

Overwriting codebase default...

? What language would you like to use to write Cloud Functions? JavaScript
? Do you want to use ESLint to catch probable bugs and enforce style? No
? File functions/package.json already exists. Overwrite? (y/N) N
```

n. When asked to override **functions/index.js**, Enter N and press enter


```
Overwriting codebase default...

? What language would you like to use to write Cloud Functions? JavaScript
? Do you want to use ESLint to catch probable bugs and enforce style? No
? File functions/package.json already exists. Overwrite? No
i  Skipping write of functions/package.json
? File functions/index.js already exists. Overwrite? (y/N) N
```

o. When asked to override **functions/.gitignore**, Enter N and press enter

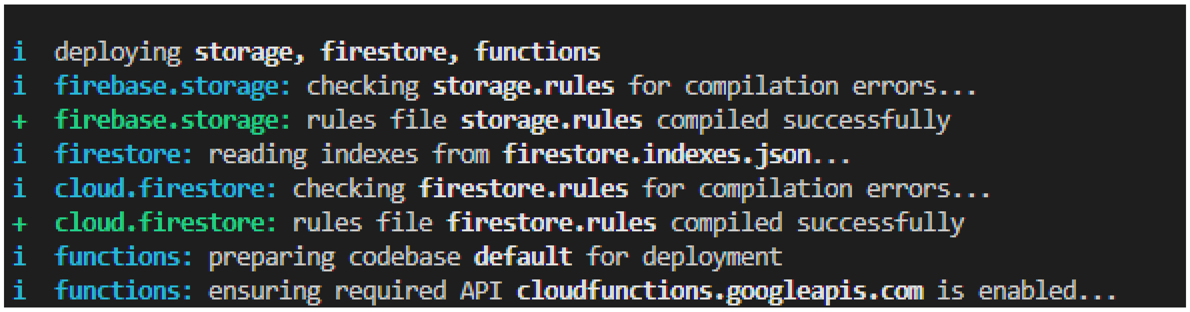
p. When asked to install dependencies enter Y and press enter and the dependencies will start installing.



```
? Do you want to use ESLint to catch probable bugs and enforce style? No
? File functions/package.json already exists. Overwrite? No
i  Skipping write of functions/package.json
? File functions/index.js already exists. Overwrite? No
i  Skipping write of functions/index.js
? File functions/.gitignore already exists. Overwrite? No
i  Skipping write of functions/.gitignore
? Do you want to install dependencies with npm now? (Y/n) Y
```

q. When asked what to call **Storage rules**, just press enter

r. Finally, run **firebase deploy**. You may need to run this command 2 times to be successful



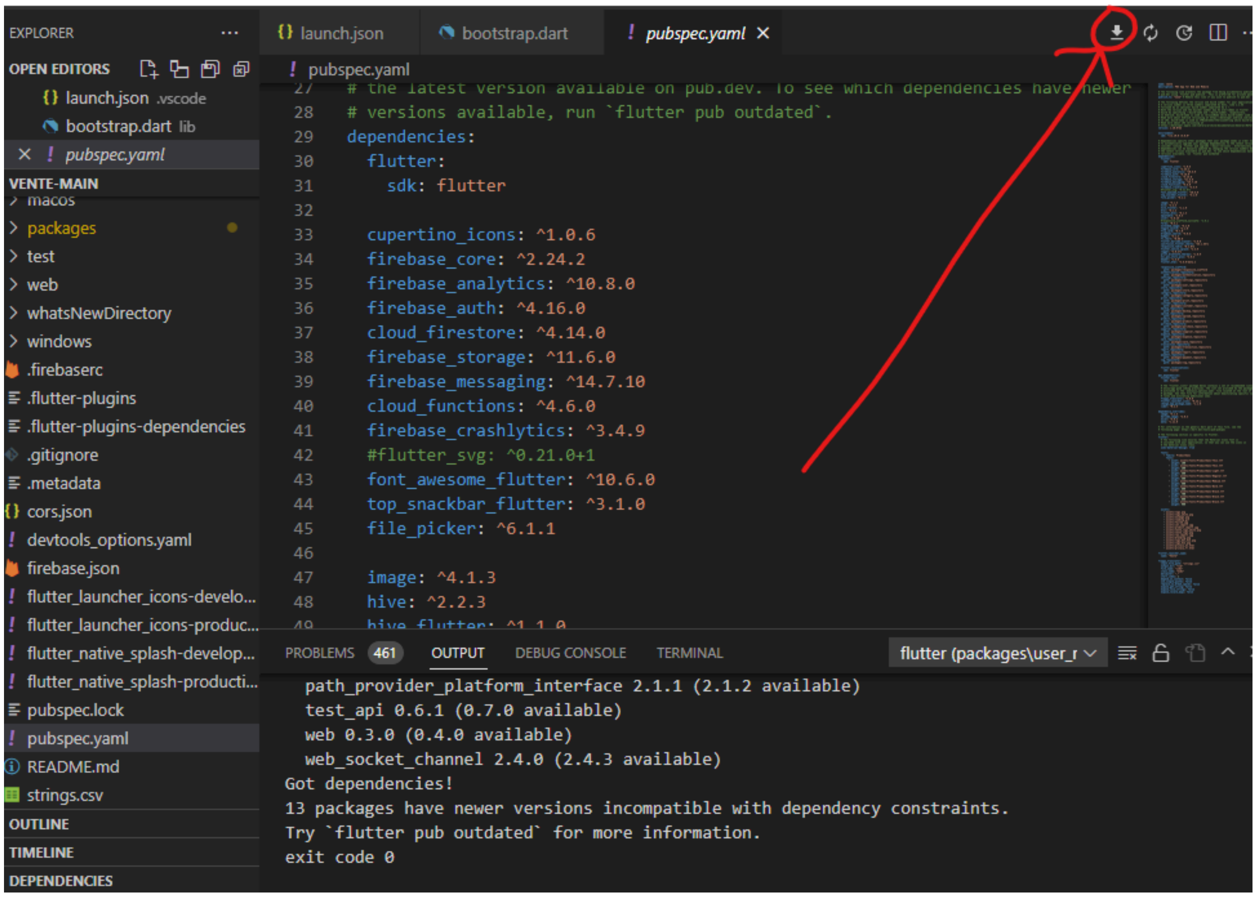
```
i  deploying storage, firestore, functions
i  firebase.storage: checking storage.rules for compilation errors...
+  firebase.storage: rules file storage.rules compiled successfully
i  firestore: reading indexes from firestore.indexes.json...
i  cloud.firestore: checking firestore.rules for compilation errors...
+  cloud.firestore: rules file firestore.rules compiled successfully
i  functions: preparing codebase default for deployment
i  functions: ensuring required API cloudfunctions.googleapis.com is enabled...
```

s. Head to Functions and click Get Started. You should see the functions we installed in the previous step

7. Flutter Setup

Now that we've set up our Firebase projects, open the Flutter project in VS Code and follow these steps to link it to Firebase.

a. Open **pubspec.yaml** file and run **flutter pub get** to download all the dependencies

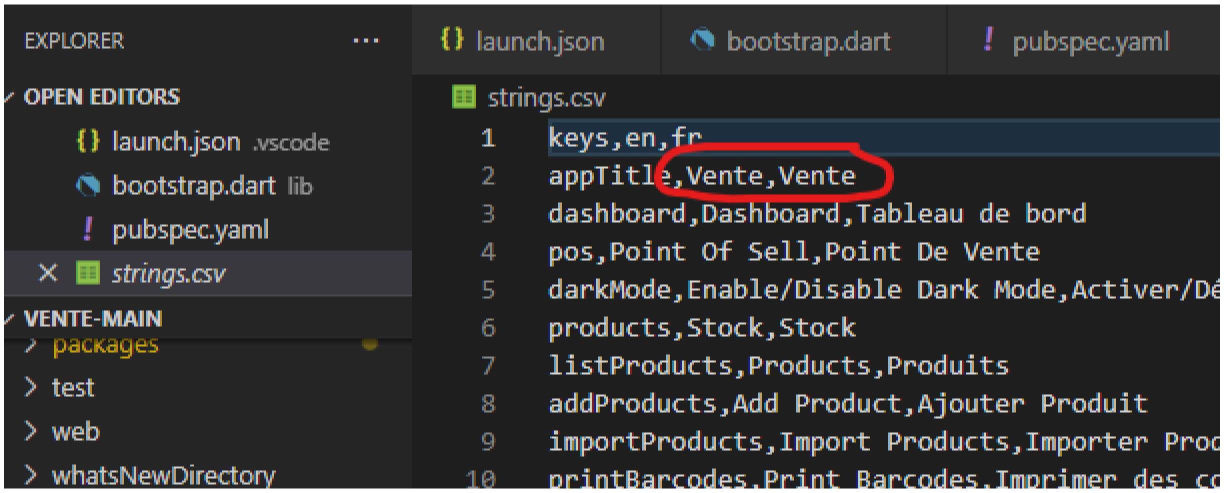


```
# the latest version available on pub.dev. To see which dependencies have newer
# versions available, run `flutter pub outdated`.
dependencies:
  flutter:
    sdk: flutter

  cupertino_icons: ^1.0.6
  firebase_core: ^2.24.2
  firebase_analytics: ^10.8.0
  firebase_auth: ^4.16.0
  cloud_firestore: ^4.14.0
  firebase_storage: ^11.6.0
  firebase_messaging: ^14.7.10
  cloud_functions: ^4.6.0
  firebase_crashlytics: ^3.4.9
  #flutter_svg: ^0.21.0+1
  font_awesome_flutter: ^10.6.0
  top_snackbar_flutter: ^3.1.0
  file_picker: ^6.1.1

  image: ^4.1.3
  hive: ^2.2.3
  hive_flutter: ^1.1.0
```

```
  path_provider_platform_interface 2.1.1 (2.1.2 available)
  test_api 0.6.1 (0.7.0 available)
  web 0.3.0 (0.4.0 available)
  web_socket_channel 2.4.0 (2.4.3 available)
Got dependencies!
13 packages have newer versions incompatible with dependency constraints.
Try `flutter pub outdated` for more information.
exit code 0
```

b. To change the package name, run **dart run change_app_package_name:main com.new.package.name** and replace **com.new.package.name** with your package name

c. To change your flutter app name run the command **flutter pub global activate rename**. Run then **rename setAppName --targets ios,android,web,macos --value "YOUR_APP_NAME"** where YOU_APP_NAME is the name of your app. Go to **strings.csv** and under the key appName, change appTitle values to your own name. Run then **dart run flappy_translator**

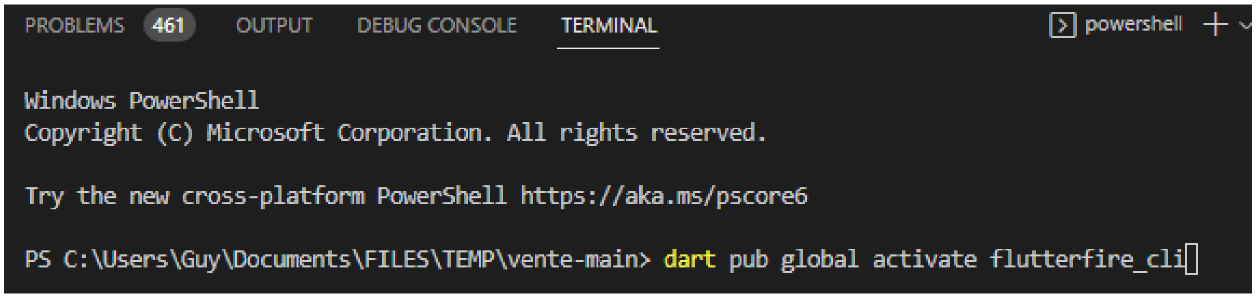


```
keys,en,fr
appTitle,Vente,Vente
dashboard,Dashboard,Tableau de bord
pos,Point Of Sell,Point De Vente
darkMode,Enable/Disable Dark Mode,Activer/Dé
products,Stock,Stock
listProducts,Products,Produits
addProducts,Add Product,Ajouter Produit
importProducts,Import Products,Importer Proc
printBarcodes,Print Barcodes,Imprimer des co
```

d. To change the logo, convert your logo to PNG format and rename it to logo. Move it to the assets folder to replace the file also named logo.png. Run then **dart run flutter_launcher_icons**

e. To change the splash screen, replace the images **logo_512.png** and **branding.png** in the assets folder with your own images with the same name and format then run **dart run flutter_native_splash:create**

f. Run in **vente** project terminal **dart pub global activate flutterfire_cli** to install the flutterfire cli

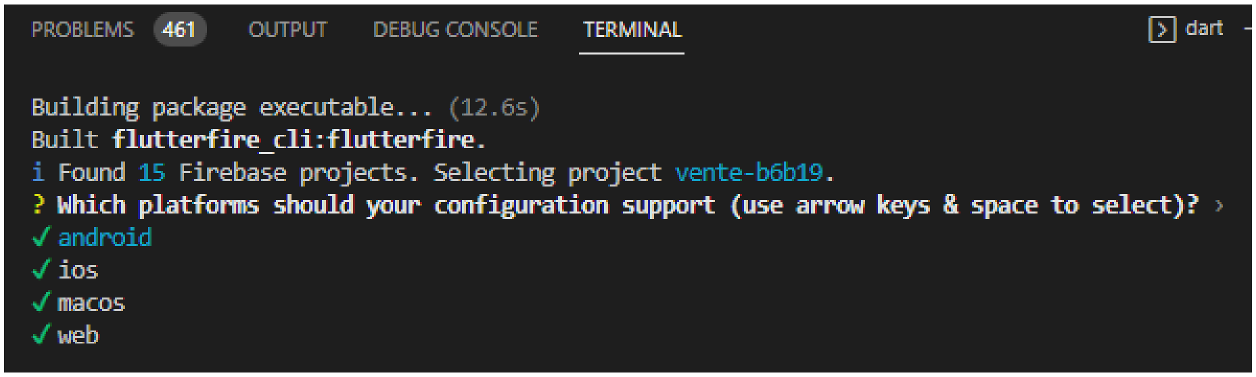


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Guy\Documents\FILES\TEMP\vente-main> dart pub global activate flutterfire_cli
```

g. Now run **flutterfire configure -o lib/firebase/prod/firebase_options.dart**. This will create the Firebase file needed to launch in your production environment. Note that if you change your package name after running this command you will have to run it again

h. In the next step, select which project to link

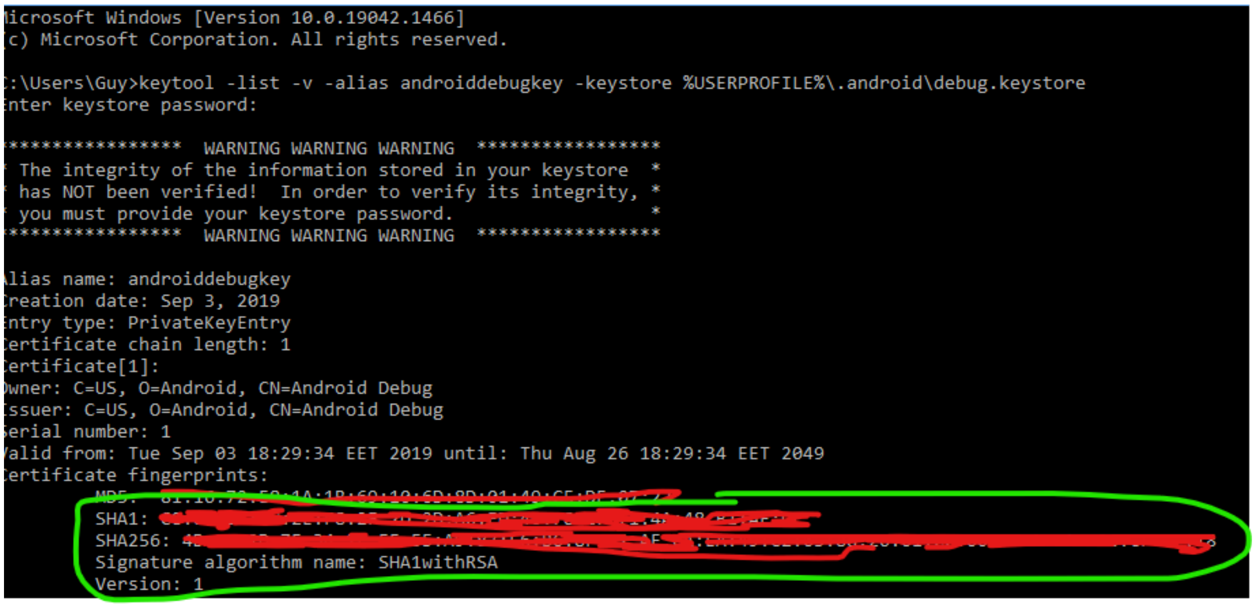
i. Make sure all platforms are selected and press enter



```
Building package executable... (12.6s)
Built flutterfire_cli:flutterfire.
i Found 15 Firebase projects. Selecting project vente-b6b19.
? Which platforms should your configuration support (use arrow keys & space to select)? ›
✓ android
✓ ios
✓ macos
✓ web
```

j. Open a terminal and run **keytool -list -v -alias androiddebugkey -keystore %USERPROFILE%\.android\debug.keystore** to generate the SHA1 and SH256 of your system. If it doesn't work, make sure java sdk is installed with the path environment well set up. If it still doesn't work, try this:

   i. Go to this path or wherever you have your keytool.exe file like **C:\Program Files\Java\jre7\bin** for example
   ii. Hold shift and right click -> then press Open command window here
   iii. The terminal will pop up, paste then the above keytool command

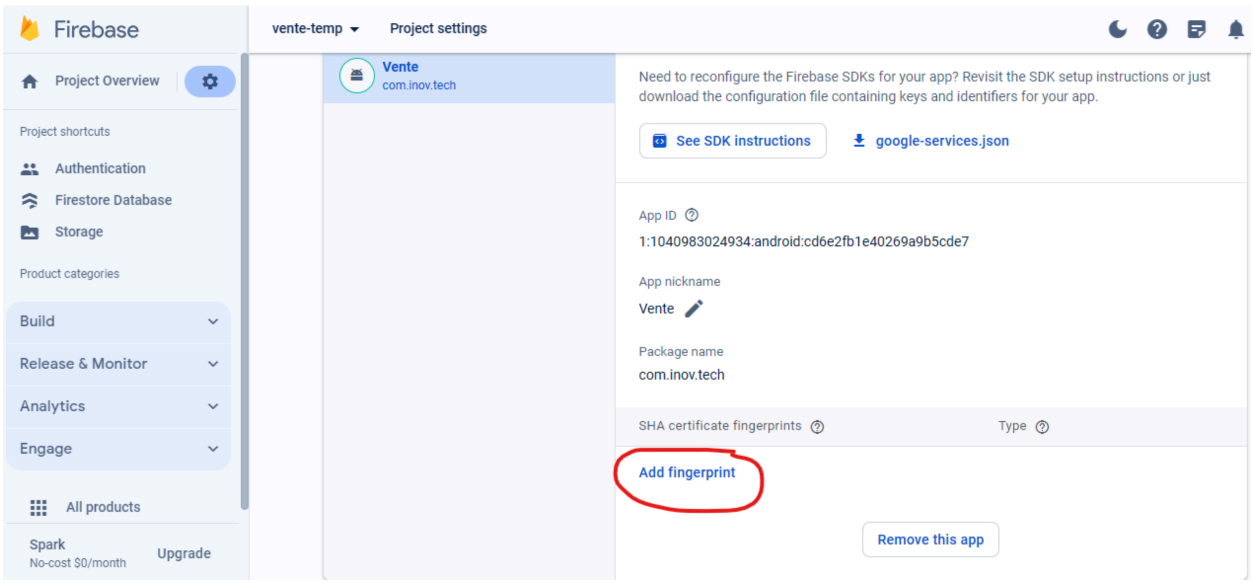


```
icrosoft Windows [Version 10.0.19042.1466]
c) Microsoft Corporation. All rights reserved.

:\Users\Guy>keytool -list -v -alias androiddebugkey -keystore %USERPROFILE%\.android\debug.keystore
nter keystore password:

****************  WARNING WARNING WARNING  *****************
 The integrity of the information stored in your keystore  *
 has NOT been verified!  In order to verify its integrity, *
 you must provide your keystore password.                  *
****************  WARNING WARNING WARNING  *****************

lias name: androiddebugkey
reation date: Sep 3, 2019
ntry type: PrivateKeyEntry
ertificate chain length: 1
ertificate[1]:
wner: C=US, O=Android, CN=Android Debug
ssuer: C=US, O=Android, CN=Android Debug
erial number: 1
alid from: Tue Sep 03 18:29:34 EET 2019 until: Thu Aug 26 18:29:34 EET 2049
ertificate fingerprints:
         MD5:  [redacted]
         SHA1: [redacted]
         SHA256: [redacted]
         Signature algorithm name: SHA1withRSA
         Version: 1
```

k. You may be asked to enter a password. If you've never entered a password, just press enter or try **android** as the password

l. The following key will allow you to launch in debug mode. For more information on debug and release keys check this post [Authenticating Your Client | Google Play services | Google for Developers]

m. Go back to your Firebase project and go to project settings. Under General locate the Android app and add the SHA1 and SHA256 previously generated.



Need to reconfigure the Firebase SDKs for your app? Revisit the SDK setup instructions or just download the configuration file containing keys and identifiers for your app.

App ID
1:1040983024934:android:cd6e2fb1e40269a9b5cde7

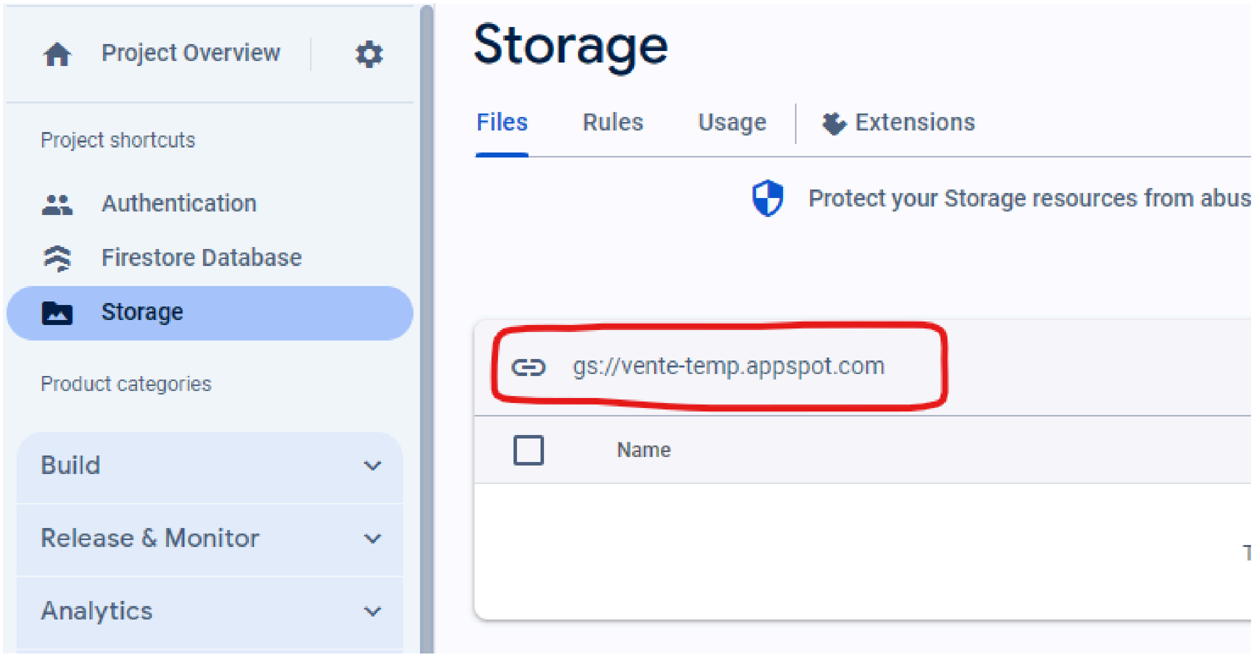
App nickname
Vente

Package name
com.inov.tech

8. Other Installation

You may notice that images do not display on the web. This is related to a CORS configuration. Generally, Firebase doesn't allow access to storage from unknown domains. The Flutter project contains a cors.json file to allow any domain to display images in Firebase Storage.

- Install gsutil from this link [Install gsutil | Cloud Storage | Google Cloud]
- Run in the terminal **gcloud init**. You may have to restart vscode for this to work
- Run in the terminal **gsutil cors set cors.json gs://YOUR_BUCKET_NAME** where **YOUR_BUCKET_NAME** is the name of your Firebase Storage Bucket which you can find in **Build > Storage** on Firebase. In my case I'd run **gsutil cors set cors.json gs://vente-temp.appspot.com**



gs://vente-temp.appspot.com

- If you wish to use **GitHub CI/CD** for deployment, you'll need to configure the **android-production-release.yaml** and **android-development-release.yaml** files in the .github/workflows folder to create releases on Github. Check this post for more info: **Deploy your Flutter App to Firebase App Distribution using GitHub Actions - Android (bernos.dev)**
- To host your app, run **flutter build web --release -t lib/main_production.dart** and the folder **build/web** will be generated which you can deploy on your server. Alternatively, you can configure Firebase hosting and GitHub actions to deploy your website every time you push your project to the main branch. The project has GitHub workflow files to help you deploy quickly with GitHub actions. You'll need to upload the different secret keys needed however to complete the workflows. Check these articles for more info:

  Deploy your Flutter App to Firebase App Distribution using GitHub Actions - Android (bernos.dev)

  Automating Flutter Web Deployments to Firebase Hosting using GitHub Actions | by Quentin Estrach | Medium

  Integrate GitHub Actions with Slack, Say Goodbye to Email Notifications (tvaidyan.com)