

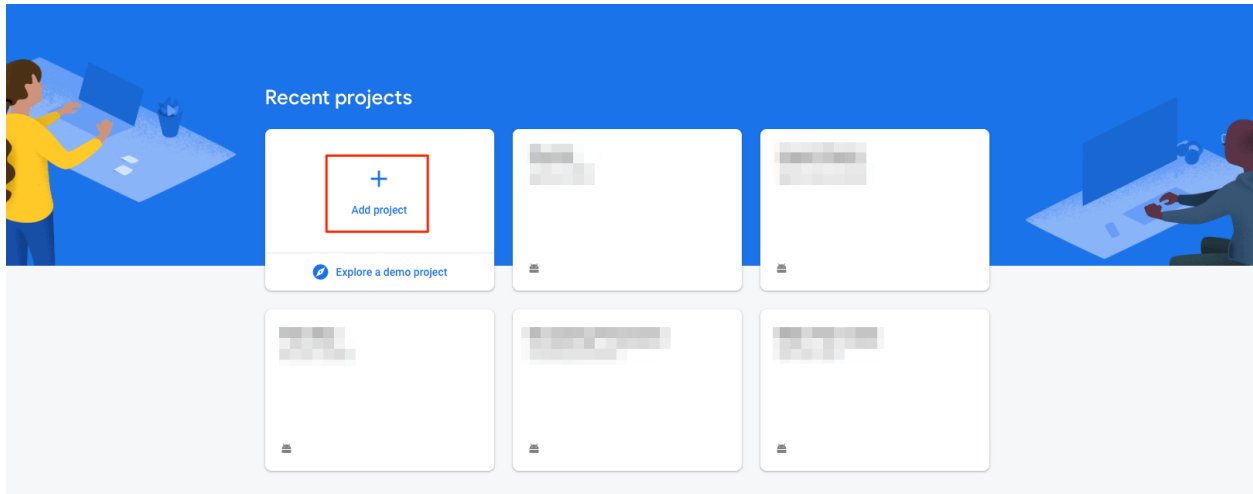
FIREBASE

Index:

- Setting up Firebase
- Installation
- Setting up Realtime Database
- Setting up Firestore Database
- Windows PC
- Mac
- Configuring Application locally
- Running Test Cases and Basic Commands
- How the Automation code is organized in Firebase
- Jenkins Integration

Setting up Firebase

To add a project to firebase, one needs to sign into firebase account.



We can create a new project by clicking on **Add project**. First step is to give a name for the project (say Firebase-test) and click **Continue**.

× Create a project (Step 1 of 3)

Let's start with a name for your project[?]

Project name

Firebase-test

fir-test-c76c5

Second step is **Google Analytics**. We can enable or disable as per requirement and click **Continue**.

✕ Create a project (Step 2 of 3)

Google Analytics enables:



A/B testing ?



Crash-free users ?



User segmentation & targeting across ?
Firebase products



Event-based Cloud Functions triggers ?



Free unlimited reporting ?



Enable Google Analytics for this project
Recommended

[Previous](#)

[Continue](#)

Click on **Create project**.

✕ Create a project (Step 2 of 2)

Google Analytics enables:



A/B testing ?



Crash-free users ?



User segmentation & targeting across ?
Firebase products



Event-based Cloud Functions triggers ?



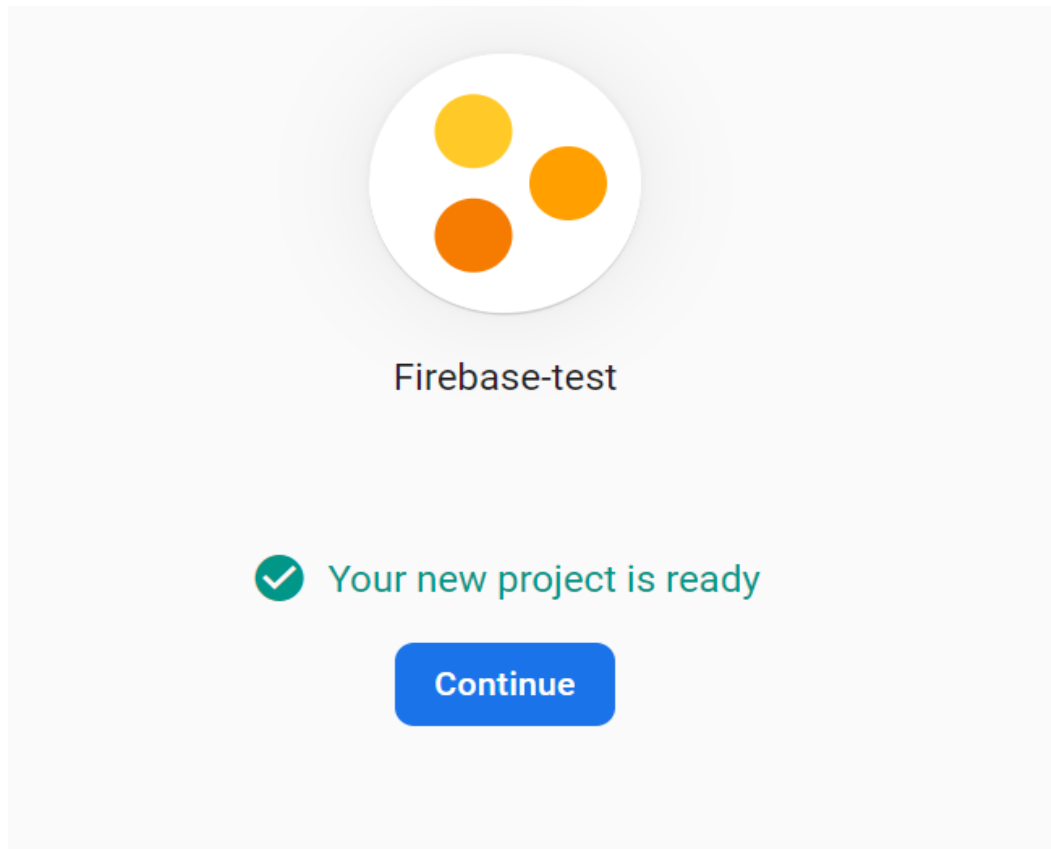
Free unlimited reporting ?



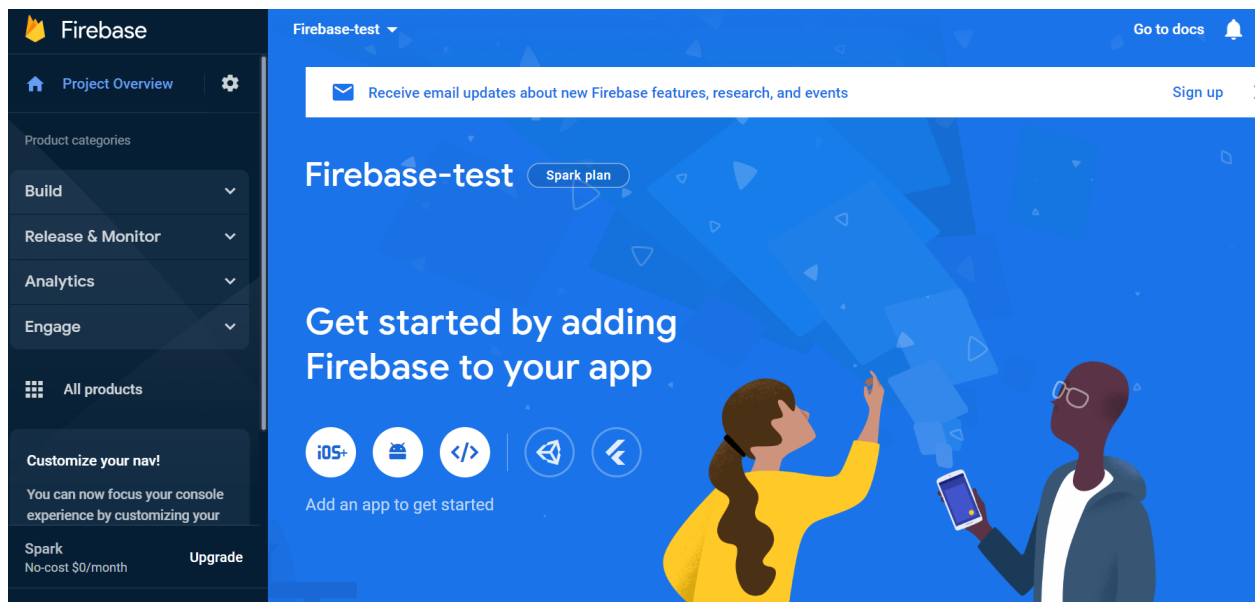
Enable Google Analytics for this project
Recommended

[Previous](#)

[Create project](#)



Click on **Continue** to see the overview of the project.



Add a Web app in the firebase project, click on the icon.



To register the web app, give a name for the app. (say- firebase-website)

×

Add Firebase to your web app

1 Register app

App nickname ?

firebase-website

☐ Also set up **Firebase Hosting** for this app. [Learn more](#) ↗

Hosting can also be set up later. There is no cost to get started anytime.

Register app

2 Add Firebase SDK

Click on the **Register app**. We can begin using the SDKs for the products we'd like to use.

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
const firebaseConfig = {
  apiKey: "AIzaSyB0sk-64ncK28GHjdHxFjeKE5C82ub8t8k",
  authDomain: "fir-test-9604c.firebaseio.com",
  projectId: "fir-test-9604c",
  storageBucket: "fir-test-9604c.appspot.com",
  messagingSenderId: "585579375033",
  appId: "1:585579375033:web:e88696ea5b08cc151b6e68"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
```

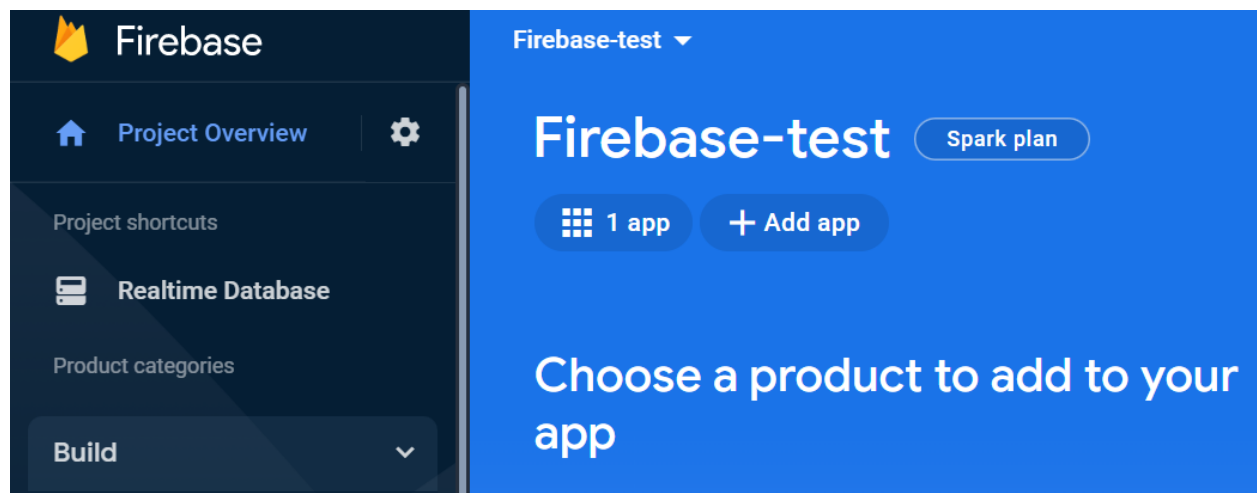


Note: This option uses the [modular JavaScript SDK](#), which provides reduced SDK size.

Learn more about Firebase for web: [Get Started](#), [Web SDK API Reference](#), [Samples](#)

Continue to console

Click on the **Continue to console** and we can see that our new app is created.



Installation

Prerequisites:

- [Install Visual Studio Code IDE](#) [You can use any IDE]
- [Install NodeJS](#)


Firebase Environment Setup: (Windows)

One needs to have Node.js preinstalled in their system as Firebase is a Node-based application. Node.js is a JavaScript runtime environment. One can download and install Node.js for a particular operating system from their [download page](#). The Node.js installation also covers the installation of npm (Node package manager).

Downloads

Latest LTS Version: **18.12.1** (includes npm 8.19.2)

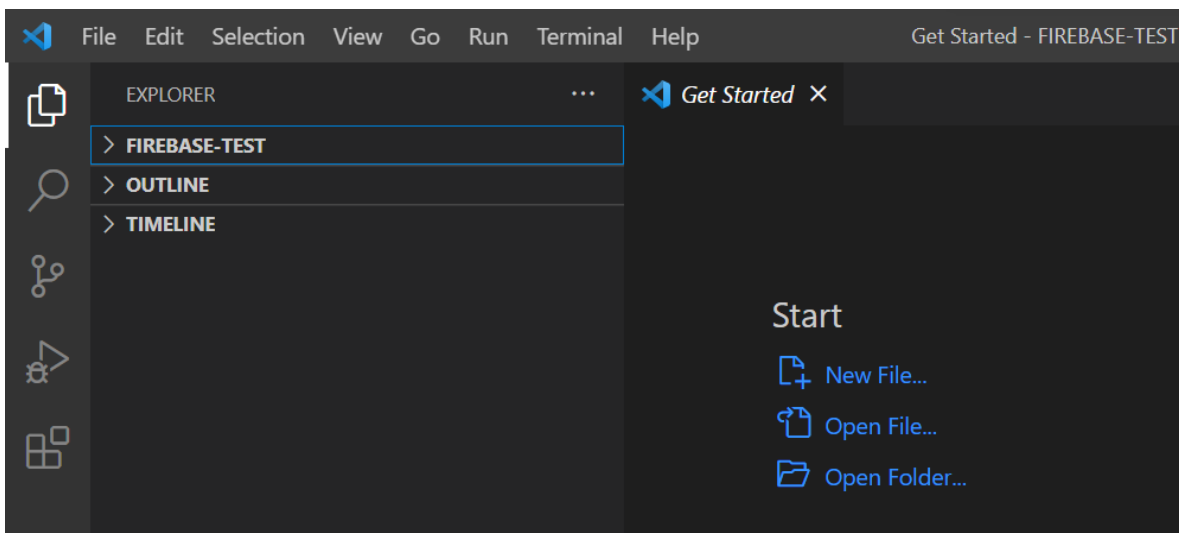
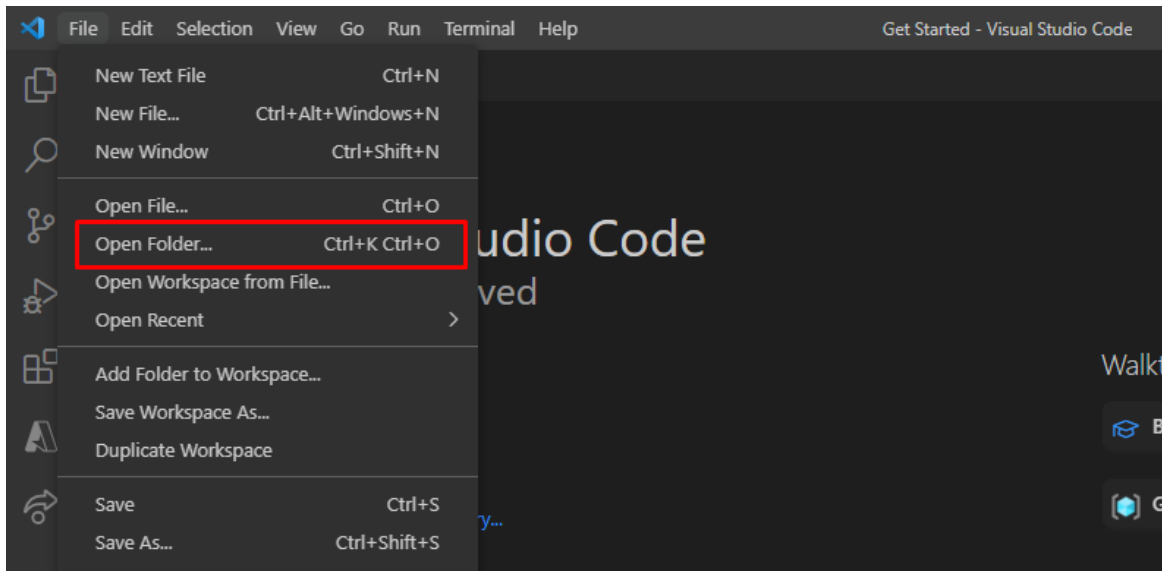
Download the Node.js source code or a pre-built installer for your platform, and start developing today.

	LTS Recommended For Most Users	Current Latest Features	
	 Windows Installer node-v18.12.1-x64.msi	 macOS Installer node-v18.12.1.pkg	 Source Code node-v18.12.1.tar.gz
Windows Installer (.msi)	32-bit	64-bit	
Windows Binary (.zip)	32-bit	64-bit	
macOS Installer (.pkg)	64-bit / ARM64		
macOS Binary (.tar.gz)	64-bit	ARM64	
Linux Binaries (x64)	64-bit		
Linux Binaries (ARM)	ARMv7	ARMv8	
Source Code	node-v18.12.1.tar.gz		

Cross verify the Node.js installation by running the command: **node --version** in the terminal. To verify the npm version, run the **npm --version** command.

Once node.js installation is done, we shall create an empty folder (say FIREBASE-TEST) in any desired location.

In Visual Studio, select the option Open Folder from the File menu. Then, add the FIREBASE-TEST folder (that we have created before) to the Visual Studio Code.



We need to create the package.json file with the **npm init** command from VS Code terminal. We have to enter details like the package name, description, and so on, as mentioned in the image given below.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Windows PowerShell

Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! <https://aka.ms/PSWindows>

PS D:\FIREBASE-TEST> npm init

npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.

package name: (firebase-test) firebaseproject

version: (1.0.0)

description: firebase features

entry point: (index.js)

test command:

git repository:

keywords:

author: QA

license: (ISC)

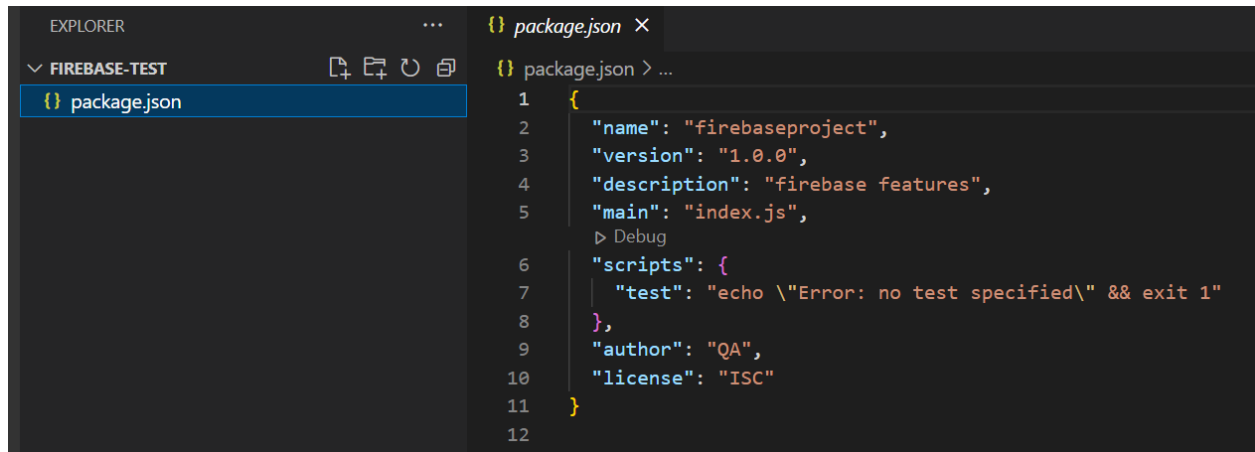
About to write to D:\FIREBASE-TEST\package.json:

```
{
  "name": "firebaseproject",
  "version": "1.0.0",
  "description": "firebase features",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "QA",
  "license": "ISC"
}
```

Is this OK? (yes) yes

PS D:\FIREBASE-TEST> █

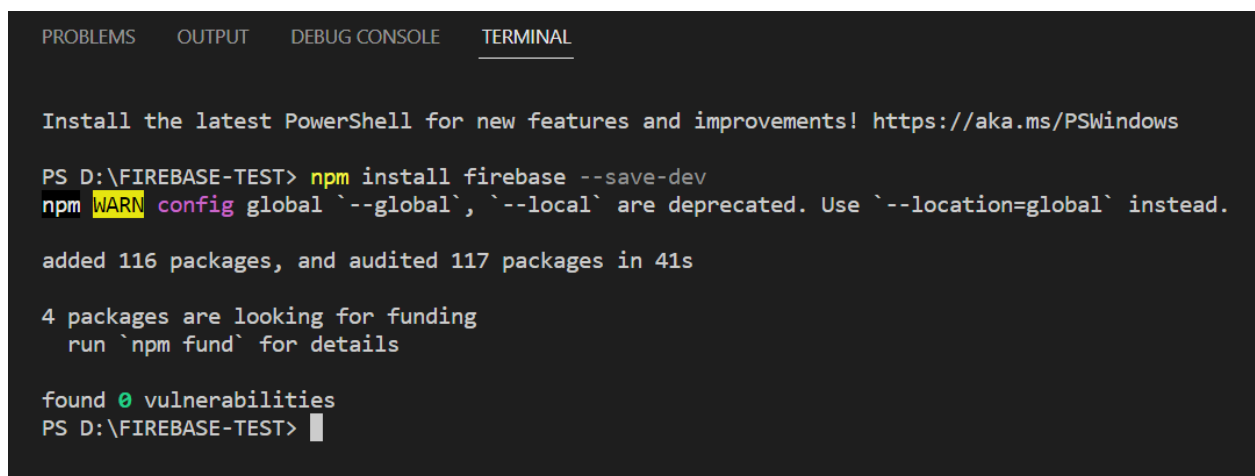
Once done, the package.json file gets created within the project folder with the information we have provided.



The screenshot shows the VS Code Explorer sidebar on the left with a folder named "FIREBASE-TEST" containing a file named "package.json". The main editor area displays the content of "package.json" with line numbers 1 through 12. The JSON content is as follows:

```
1 {
2   "name": "firebaseproject",
3   "version": "1.0.0",
4   "description": "firebase features",
5   "main": "index.js",
6   "scripts": {
7     "test": "echo \"Error: no test specified\" && exit 1"
8   },
9   "author": "QA",
10  "license": "ISC"
11 }
12
```

Finally, to install **Firebase** run the **npm install firebase --save-dev** command in the terminal.



The screenshot shows the VS Code terminal with the "TERMINAL" tab selected. The output of the command `npm install firebase --save-dev` is displayed. The terminal text is as follows:

```
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\FIREBASE-TEST> npm install firebase --save-dev
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.

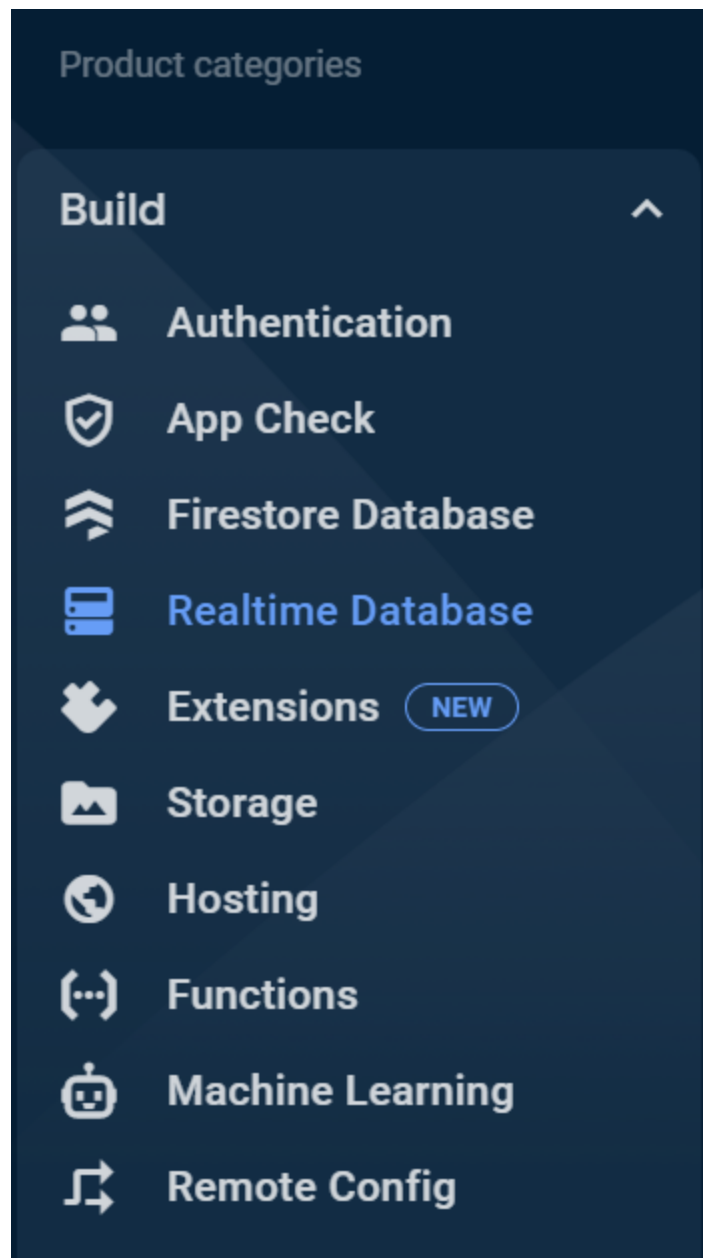
added 116 packages, and audited 117 packages in 41s

4 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS D:\FIREBASE-TEST> 
```

Setting up Realtime Database

In the Firebase console, Click on **Build** under Project categories. In the dropdown menu, Click on Realtime Database.



Click on **Create Database**.

Firebase-test ▼

Realtime Database

Store and sync data in real time

Create Database

In the pop up window, One can choose any Realtime Database location and Click On **Next**.

Set up database

1 Database options

2 Security rules

Your location setting is where your Realtime Database data will be stored.

Realtime Database location

United States (us-central1)

Belgium (europe-west1)

Singapore (asia-southeast1)

Cancel

Next

Select **Start in test mode** and Click **Enable**.

Set up database

1 Database options

2 Security rules

Once you have defined your data structure **you will have to write rules to secure your data.**
[Learn more](#)

☐

Start in locked 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": {
    ".read": "now < 1671561000000", // 2022-12-21
    ".write": "now < 1671561000000", // 2022-12-21
  }
}
```

!

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

Cancel

Enable



In the Visual Studio editor, copy the SDK code to the app we created.

```
<script type="module">
  // Import the functions you need from the SDKs you need
  import { initializeApp } from "https://www.gstatic.com/firebasejs/9.14.0/firebase-app.js";
  // TODO: Add SDKs for Firebase products that you want to use
  // https://firebase.google.com/docs/web/setup#available-libraries

  // Your web app's Firebase configuration
  const firebaseConfig = {
    apiKey: "AIzaSyB0sk-64ncK28GHjdxFjeKE5C82ub8t8k",
    authDomain: "fir-test-9604c.firebaseio.com",
    projectId: "fir-test-9604c",
    storageBucket: "fir-test-9604c.appspot.com",
    messagingSenderId: "585579375033",
    appId: "1:585579375033:web:e88696ea5b08cc151b6e68"
  };

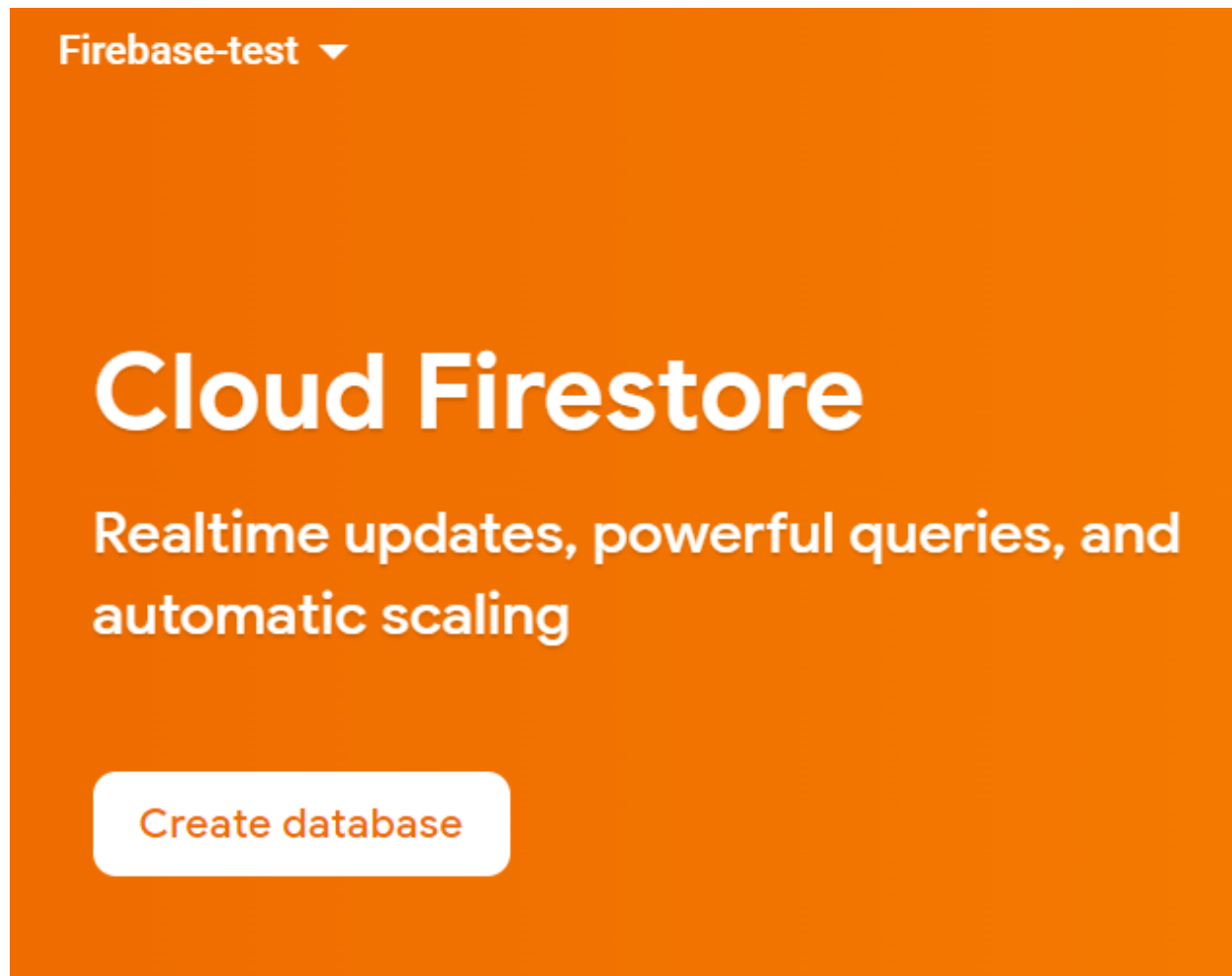
  // Initialize Firebase
  const app = initializeApp(firebaseConfig);
```

Import the commands from firebase/database to Read, Write, Update and Delete data in Firebase Database.

```
import {getDatabase, ref, get, set, child, update, remove}  
from "https://www.gstatic.com/firebasejs/9.14.0/firebase-database.js"
```

Setting up Firestore Database

To setup Firestore Database, Click on Create Database.



Select **Start in test mode** and click on **Next**.

Create database



- 1 Secure rules for Cloud Firestore — 2 Set Cloud Firestore location

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
        request.time < timestamp.date(2022, 12, 21);
    }
  }
}
```

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

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

Cancel

Next

Select the Cloud Firestore Location and Click on **Enable**.

Create database

Secure rules for Cloud Firestore

2 Set Cloud Firestore location

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

Multi-region

eur3 (Europe)

nam5 (United States)

Regional

asia-east1 (Taiwan)

asia-east2 (Hong Kong)

asia-northeast1 (Tokyo)

this project

Cancel

Enable

Now the database is created. This database is split up into Collections and Documents.

Firestore-test

Cloud Firestore

Data

Rules

Indexes

Usage

Extensions

NEW

Panel view

Query builder

🏠

📁 fir-test-9604c

+ Start collection

