angular-firebase-set-up

Prerequisites

Before starting this tutorial, you first need to make sure, you have:

- A recent version of Node.js (v8.9+) and NPM installed on your system,
- The latest Angular CLI 7 installed on your system.

This is a PWA sample using Angular CLI.

This sample contains following features.

- Angular Service Worker
- App Shell with Angular Universal

Setting up project (Tutorial)

- The starter project can be found @ https://github.com/puku0x/angular-pwa-sample

1. Install Angular CLI

```
$ npm i -g @angular/cli
```

2. Create an app

```
$ ng new my-app --routing --style=scss
$ cd my-app
```

3. Add service worker

```
$ ng add @angular/pwa --project=my-app
```

4. Generate App Shell

```
$ ng g app-shell --client-project=my-app --universal-project=my-app
```

5. Run `ng run` to build the app

```
$ ng run my-app:app-shell:production
```

Congratulations ! Your Angular application is built in `dist/my-app`.

You can see the app by using a static file server.

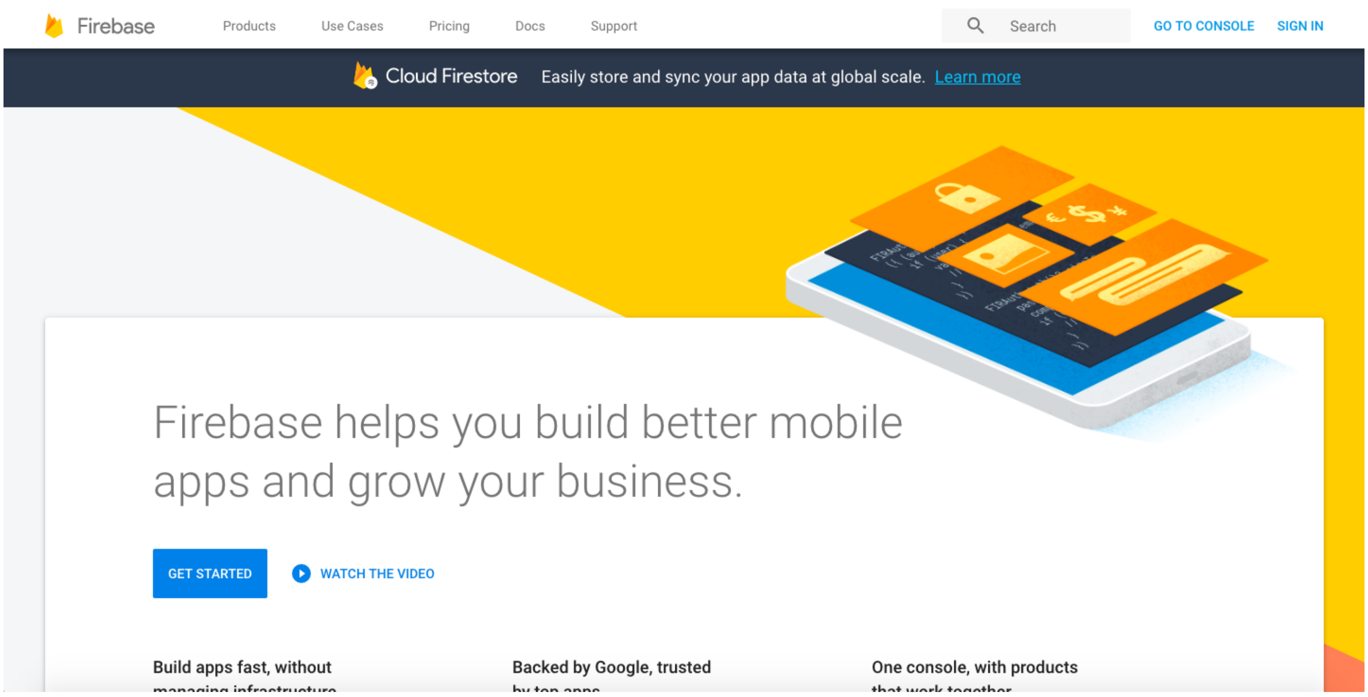
```
$ npx node-static ./dist/my-app --spa
```

Setting Up

How can I get API Key, Auth Domain, Database URL and Storage Bucket from my Firebase account?

For obtaining API Key, Auth Domain, Database URL and Storage Bucket follow below mention steps:

Step 1 : Go to Firebase.com



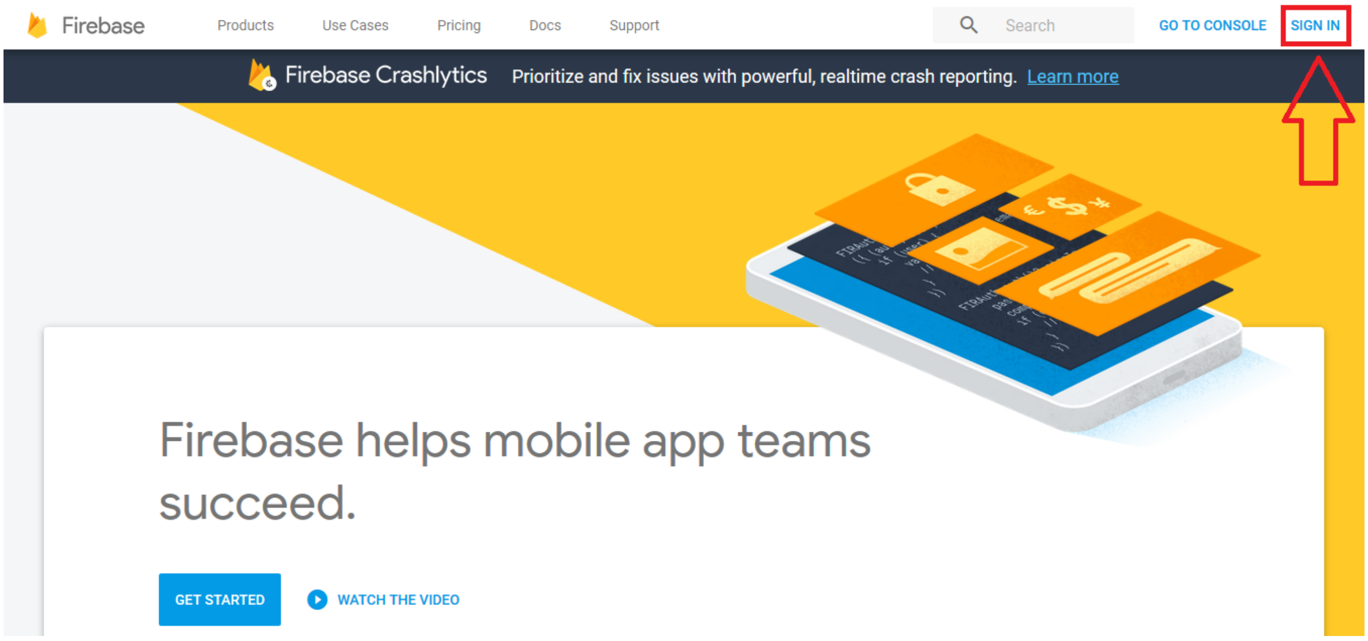
Firebase helps you build better mobile apps and grow your business.

Build apps fast, without managing infrastructure

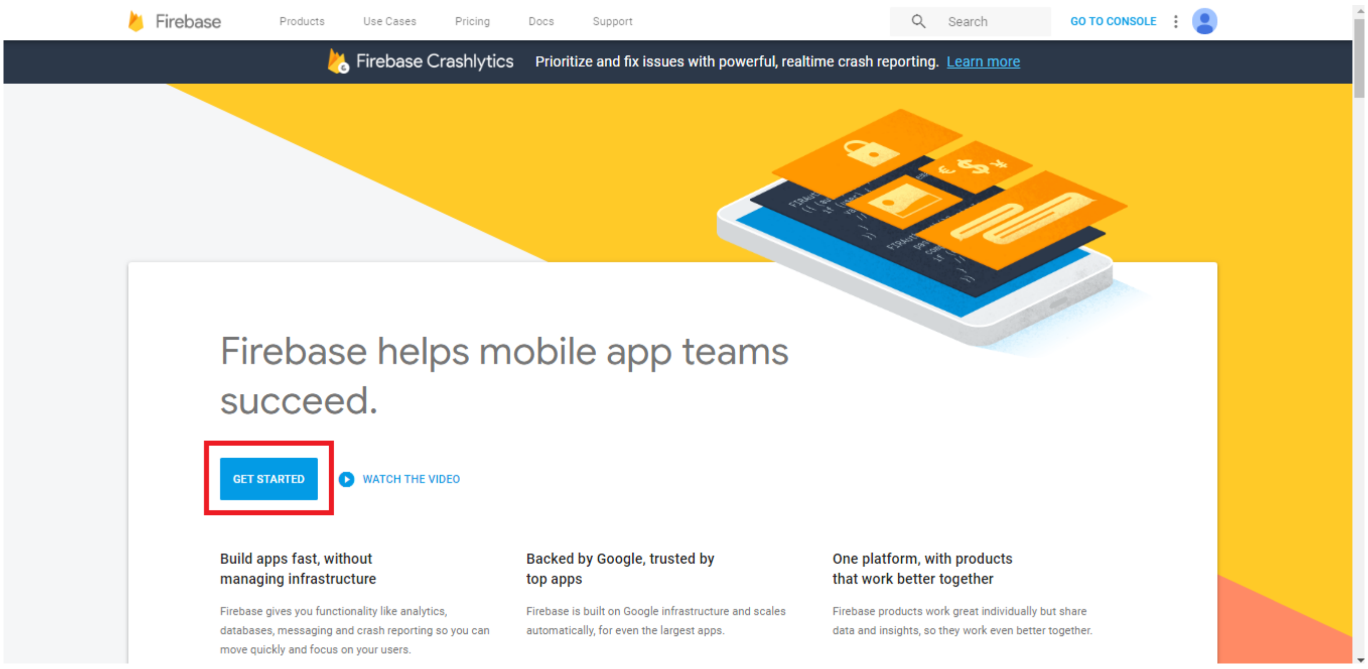
Backed by Google, trusted by top apps

One console, with products that work together

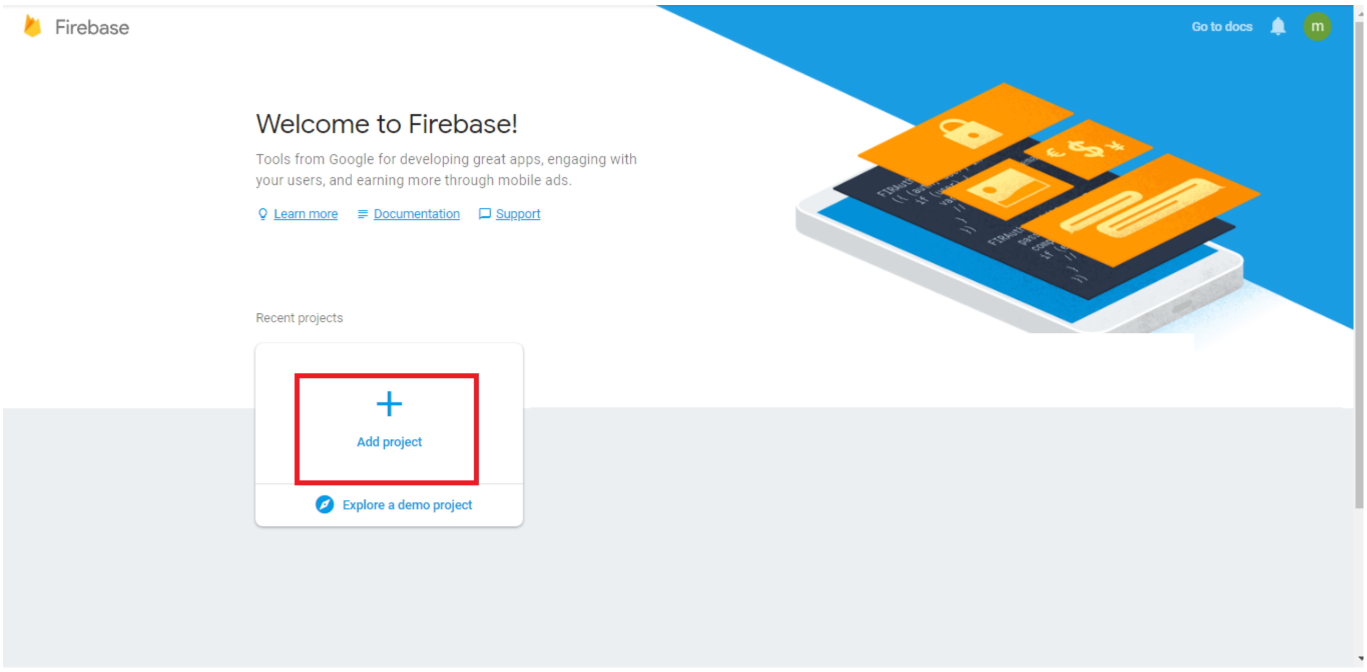
Step 2 : **If you have a **Firebase account, **Signin**, else **Signup**


Firebase
Products
Use Cases
Pricing
Docs
Support
Search
GO TO CONSOLE
SIGN IN
Firebase Crashlytics
Prioritize and fix issues with powerful, realtime crash reporting. Learn more
Firebase helps mobile app teams succeed.
GET STARTED
WATCH THE VIDEO

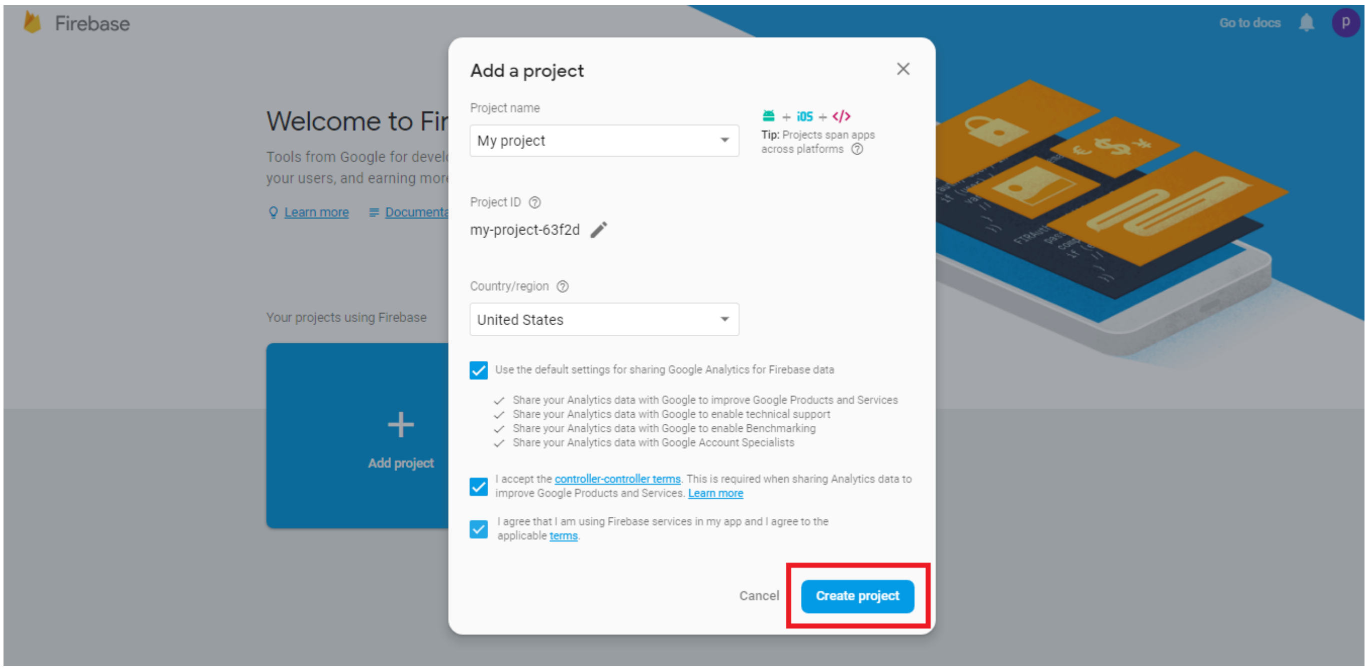
Step 3 : **Click on **Get Started


Firebase
Products
Use Cases
Pricing
Docs
Support
Search
GO TO CONSOLE
Firebase Crashlytics
Prioritize and fix issues with powerful, realtime crash reporting. Learn more
Firebase helps mobile app teams succeed.
GET STARTED
WATCH THE VIDEO
Build apps fast, without managing infrastructure
Firebase gives you functionality like analytics, databases, messaging and crash reporting so you can move quickly and focus on your users.
Backed by Google, trusted by top apps
Firebase is built on Google infrastructure and scales automatically, for even the largest apps.
One platform, with products that work better together
Firebase products work great individually but share data and insights, so they work even better together.

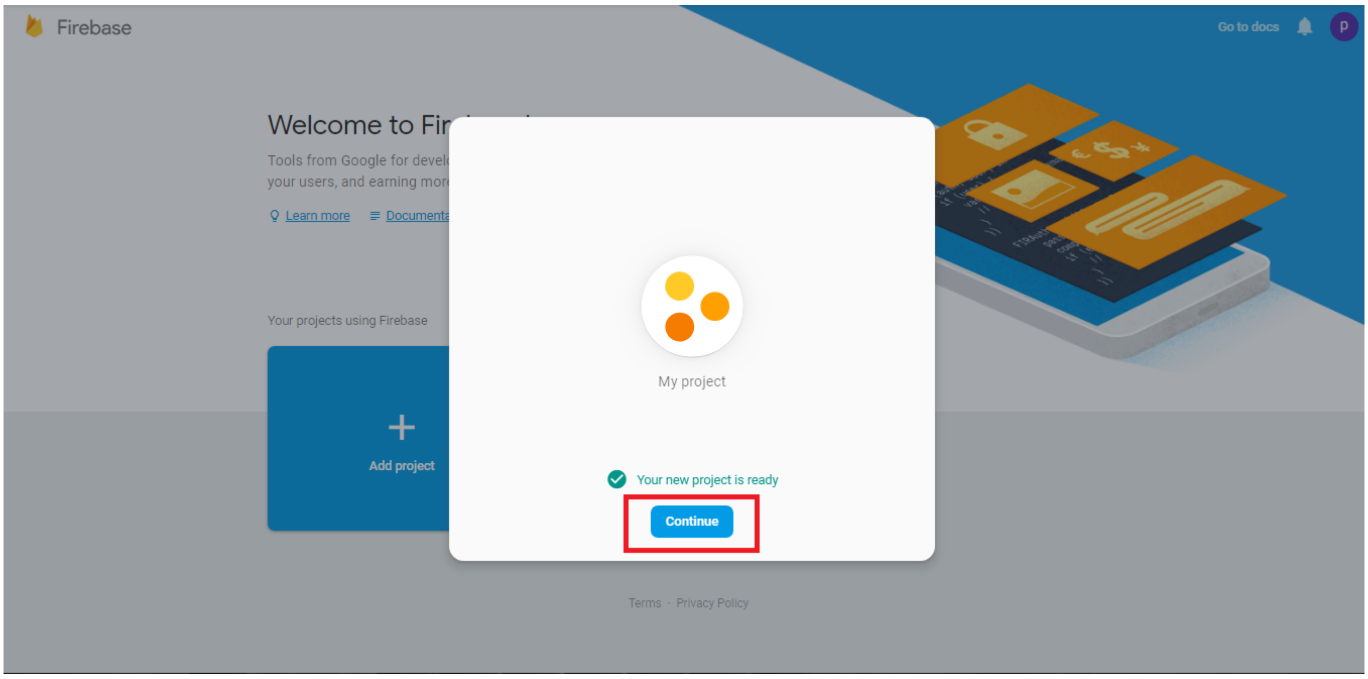
Step 4 : **Click on **Add Project



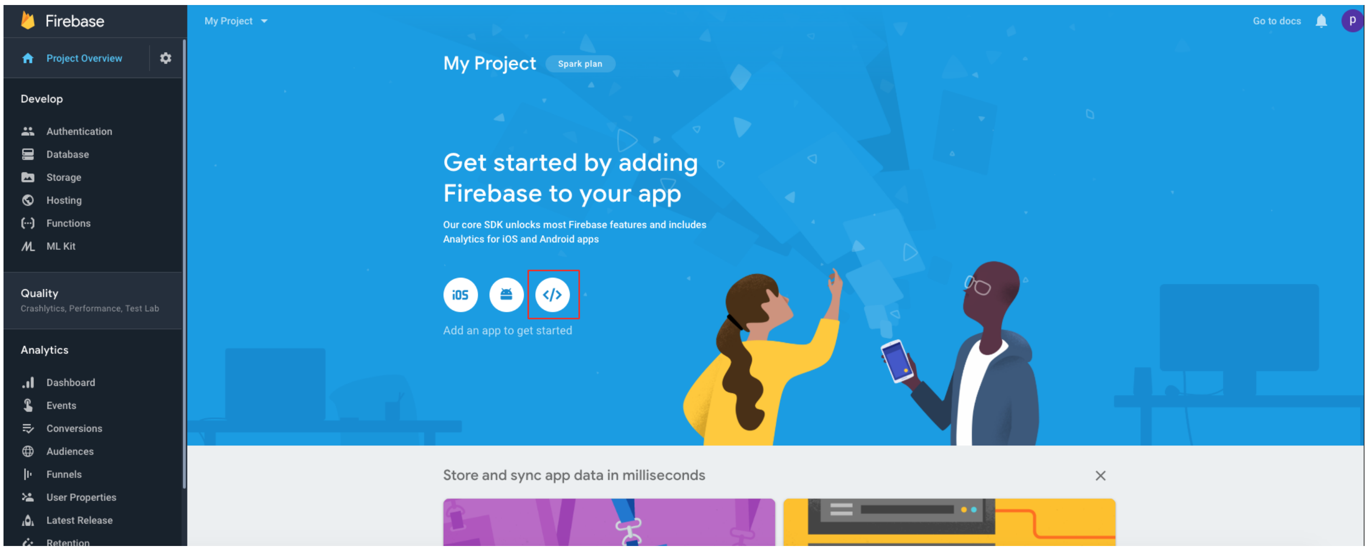
Step 5 : **Enter specific details, and click on **Create Project


Firebase
Go to docs
Add a project
Project name
My project
Tip: Projects span apps across platforms
Project ID
my-project-63f2d
Country/region
United States
Use the default settings for sharing Google Analytics for Firebase data
Share your Analytics data with Google to improve Google Products and Services
Share your Analytics data with Google to enable technical support
Share your Analytics data with Google to enable Benchmarking
Share your Analytics data with Google Account Specialists
I accept the controller-controller terms. This is required when sharing Analytics data to improve Google Products and Services. Learn more
I agree that I am using Firebase services in my app and I agree to the applicable terms.
Cancel
Create project

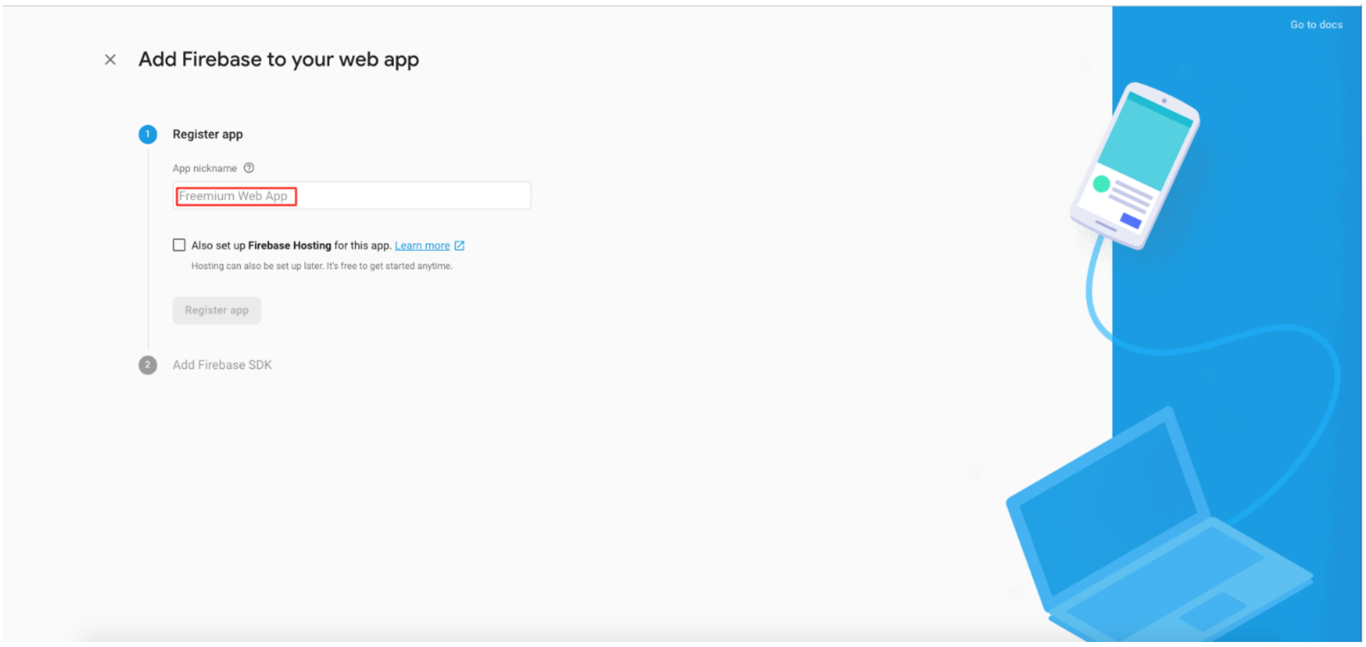
Step 6 : **Click on **Continue


Firebase
Go to docs
Welcome to Fir
Tools from Google for devel
your users, and earning mor
Learn more
Documenta
Your projects using Firebase
Add project
My project
Your new project is ready
Continue
Terms · Privacy Policy

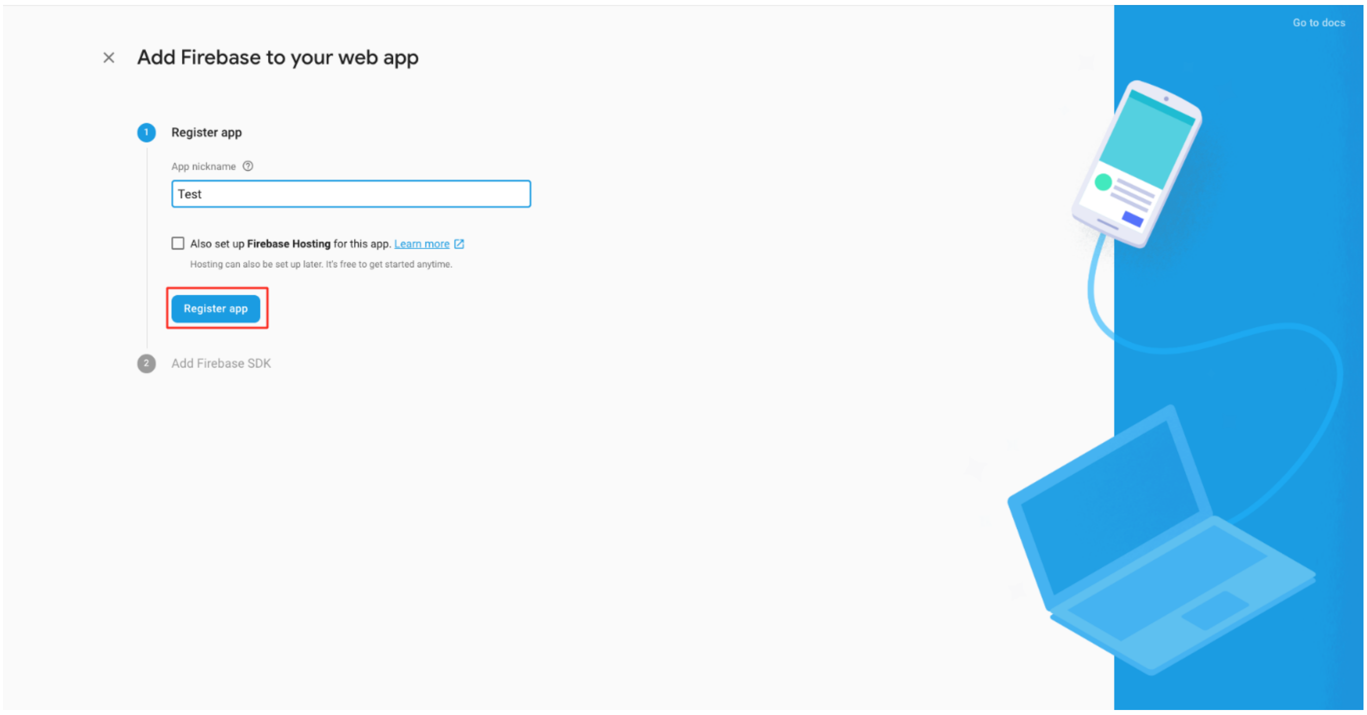
Step 7 : **Click on **Add Firebase to your web app


Firebase
Project Overview
Develop
Authentication
Database
Storage
Hosting
Functions
ML Kit
Quality
Crashlytics, Performance, Test Lab
Analytics
Dashboard
Events
Conversions
Audiences
Funnels
User Properties
Latest Release
Retention
My Project
Go to docs
My Project
Spark plan
Get started by adding Firebase to your app
Our core SDK unlocks most Firebase features and includes Analytics for iOS and Android apps
iOS
Add an app to get started
Store and sync app data in milliseconds

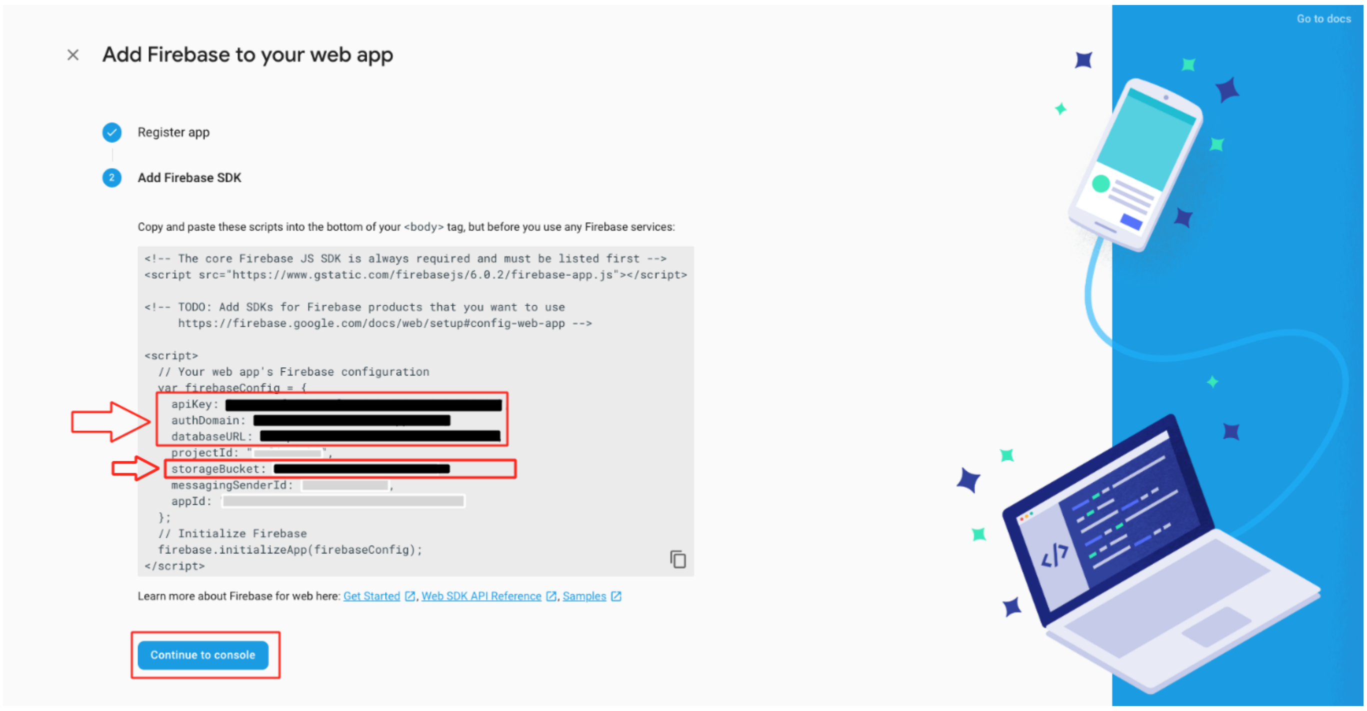
Step 8 : Enter **App nickname**


Go to docs
Add Firebase to your web app
Register app
App nickname
Freemium Web App
Also set up Firebase Hosting for this app. Learn more
Hosting can also be set up later. It's free to get started anytime.
Register app
Add Firebase SDK

Step 9 : Click on **Register app**


Go to docs
Add Firebase to your web app
Register app
App nickname
Test
Also set up Firebase Hosting for this app. Learn more
Hosting can also be set up later. It's free to get started anytime.
Register app
Add Firebase SDK

Step 10 : Here is your **apiKey, authDomain,databaseURL and Storage Bucket**. Copy appropriate keys and paste them in **My Apps >> Editor >> Messenger>> Enter apiKey, authDomain** and **databaseURL, Storage Bucket** section and click on **Continue to Console**


Add Firebase to your web app

Register app

Add Firebase SDK

Copy and paste these scripts into the bottom of your `<body>` tag, but before you use any Firebase services:

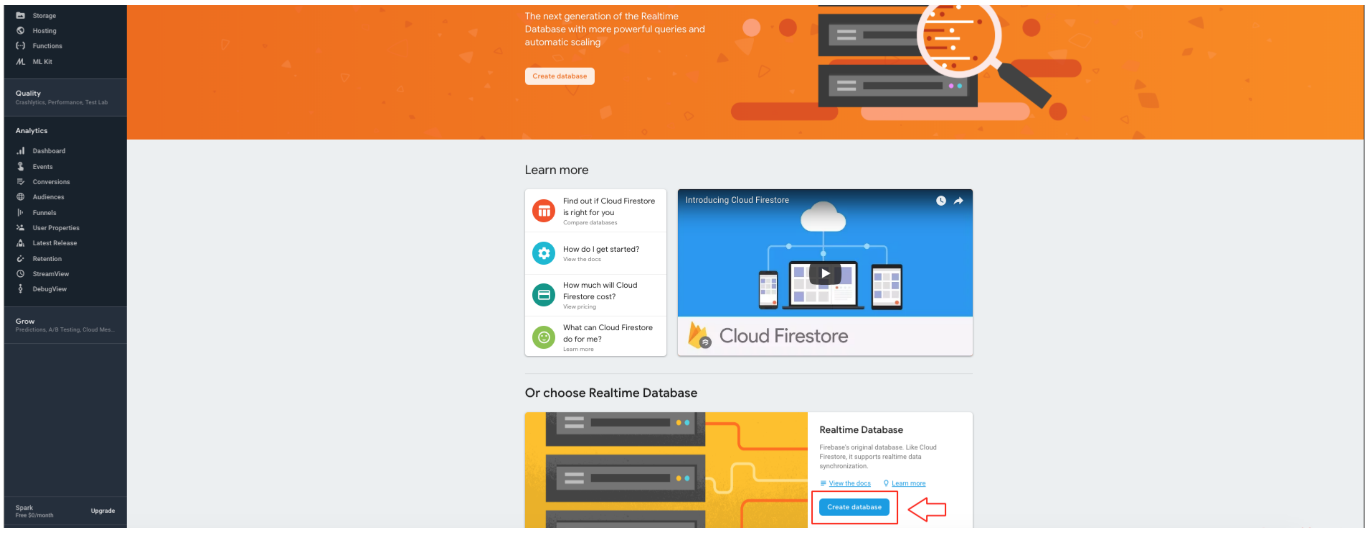
```
<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/6.0.2/firebase-app.js"></script>

<!-- TODO: Add SDKs for Firebase products that you want to use
     https://firebase.google.com/docs/web/setup#config-web-app -->

<script>
  // Your web app's Firebase configuration
  var firebaseConfig = {
    apiKey: [redacted]
    authDomain: [redacted]
    databaseURL: [redacted]
    projectId: "[illegible]",
    storageBucket: [redacted]
    messagingSenderId: [illegible],
    appId: [illegible]
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
</script>
```

Learn more about Firebase for web here: Get Started, Web SDK API Reference, Samples

Step 11 : In **Real time Database ,** Click on ****Create database .



The next generation of the Realtime Database with more powerful queries and automatic scaling

Learn more

Find out if Cloud Firestore is right for you
Compare databases

How do I get started?
View the docs

How much will Cloud Firestore cost?
View pricing

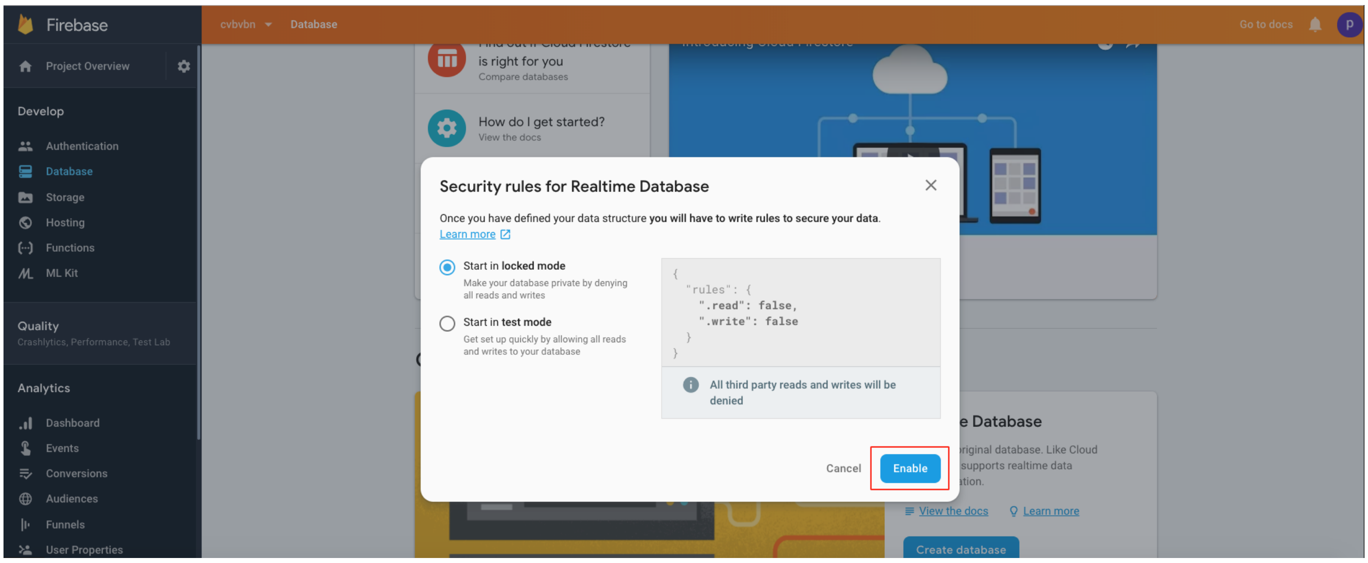
What can Cloud Firestore do for me?
Learn more

Or choose Realtime Database

Realtime Database

Firebase's original database. Like Cloud Firestore, it supports realtime data synchronization.

Step 12 : **Click on **Enable



Security rules for Realtime Database

Once you have defined your data structure **you will have to write rules to secure your data.**
Learn more

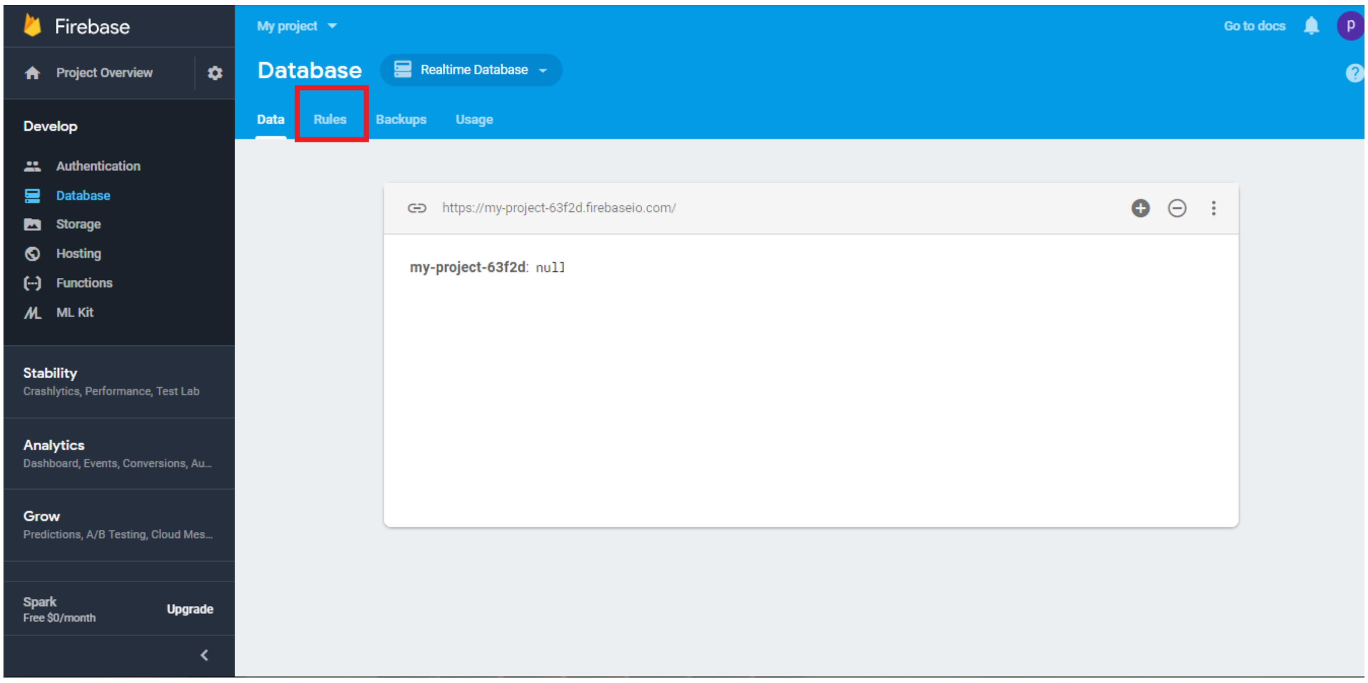
[✓ Start in **locked mode**]
Make your database private by denying all reads and writes

Start in **test mode**
Get set up quickly by allowing all reads and writes to your database

```
{
  "rules": {
    ".read": false,
    ".write": false
  }
}
```

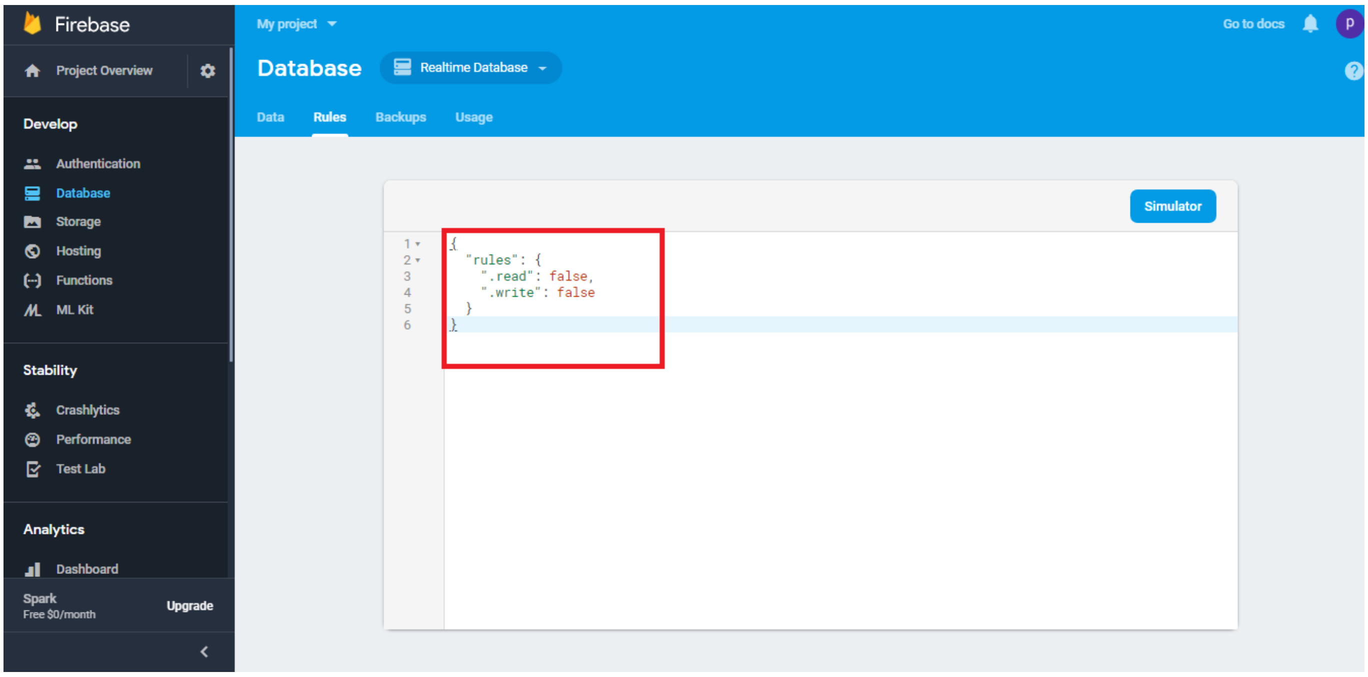
All third party reads and writes will be denied

Step 13 : **Click on **Rules



```
my-project-63f2d: null
```

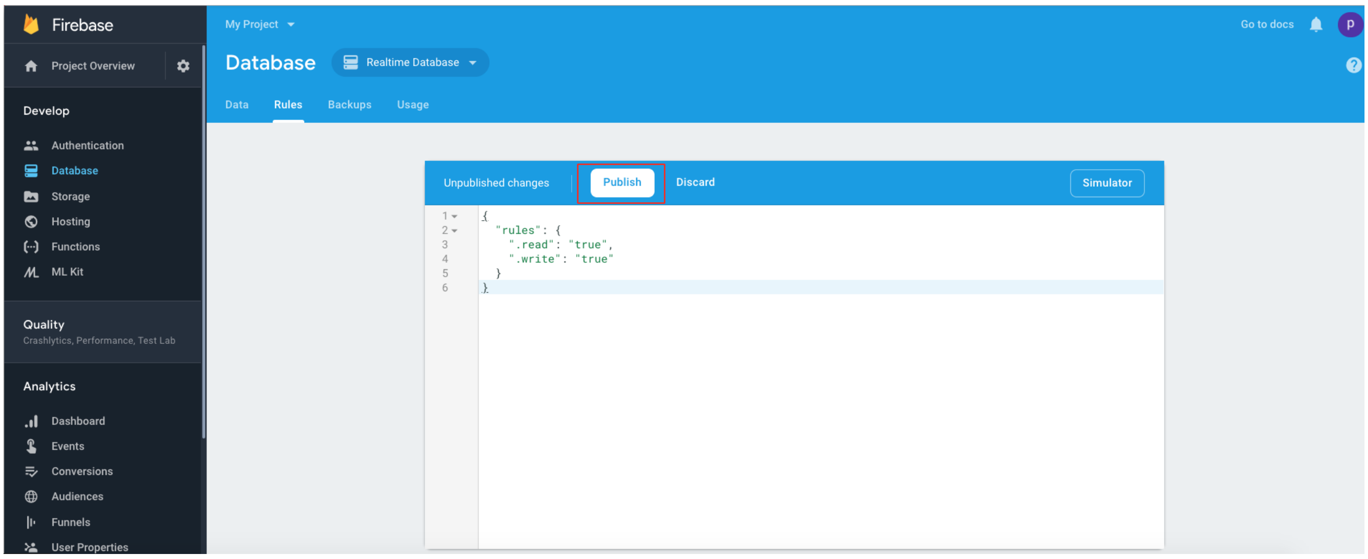
**Step 14 : **Delete the entire code



```
{
  "rules": {
    ".read": false,
    ".write": false
  }
}
```

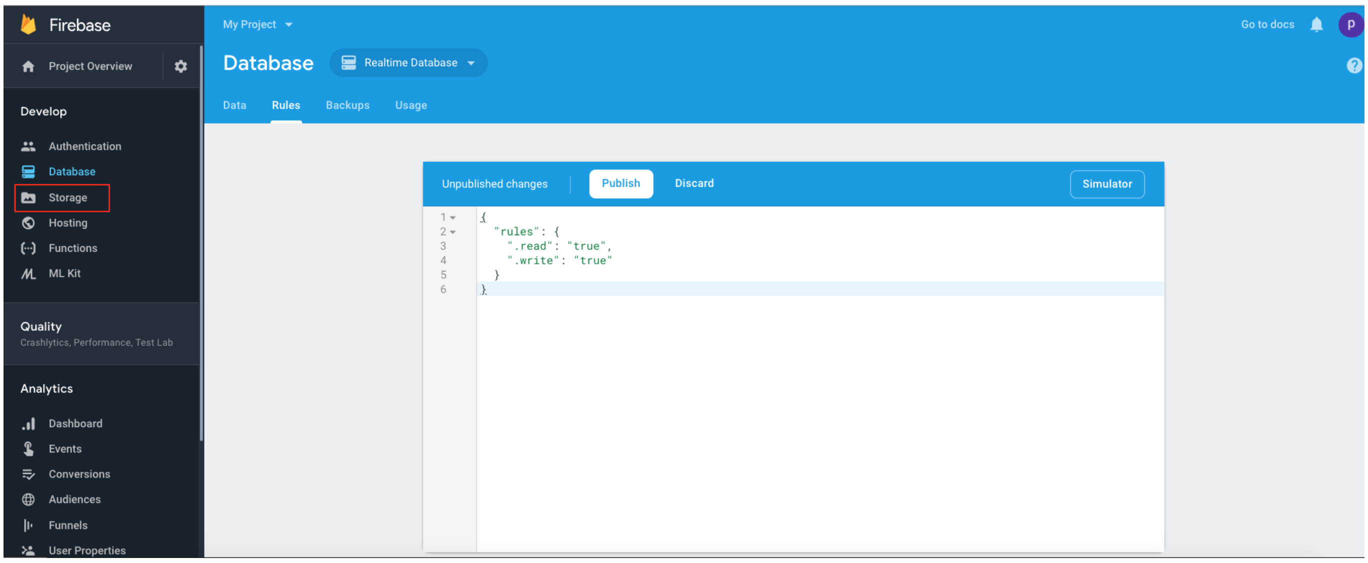
Step 15 : **Now enter the following code, and click on **Publish

```
{
  "rules": {
    ".read": "true",
    ".write": "true"
  }
}
```



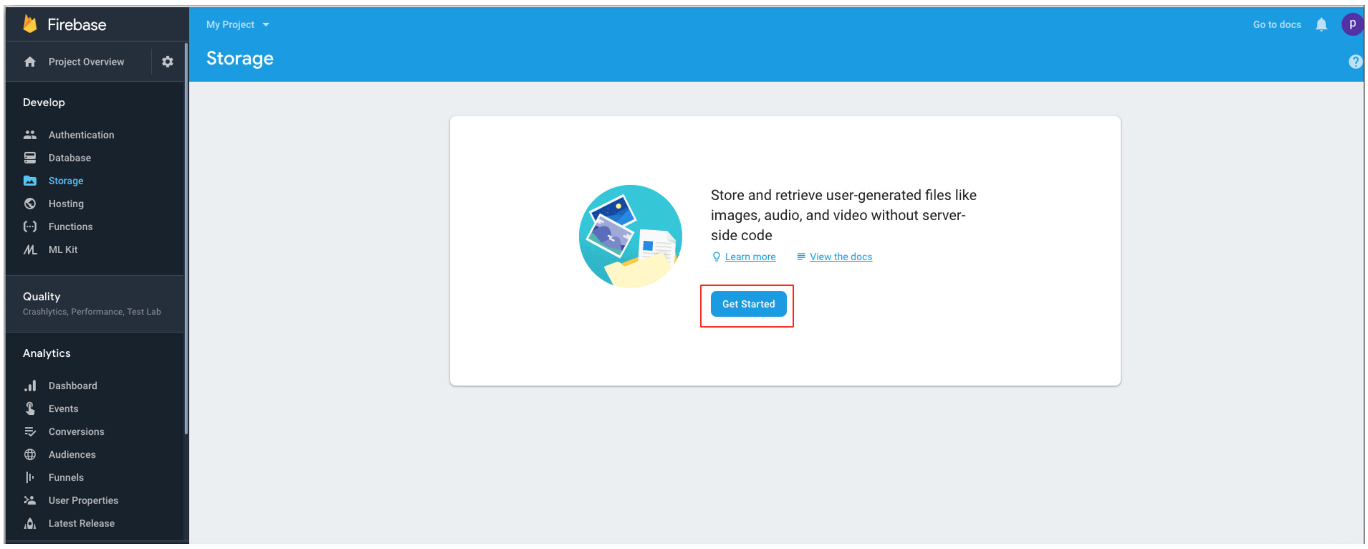
```
{
  "rules": {
    ".read": "true",
    ".write": "true"
  }
}
```

Step 16 : Click on **Storage** from left menu.



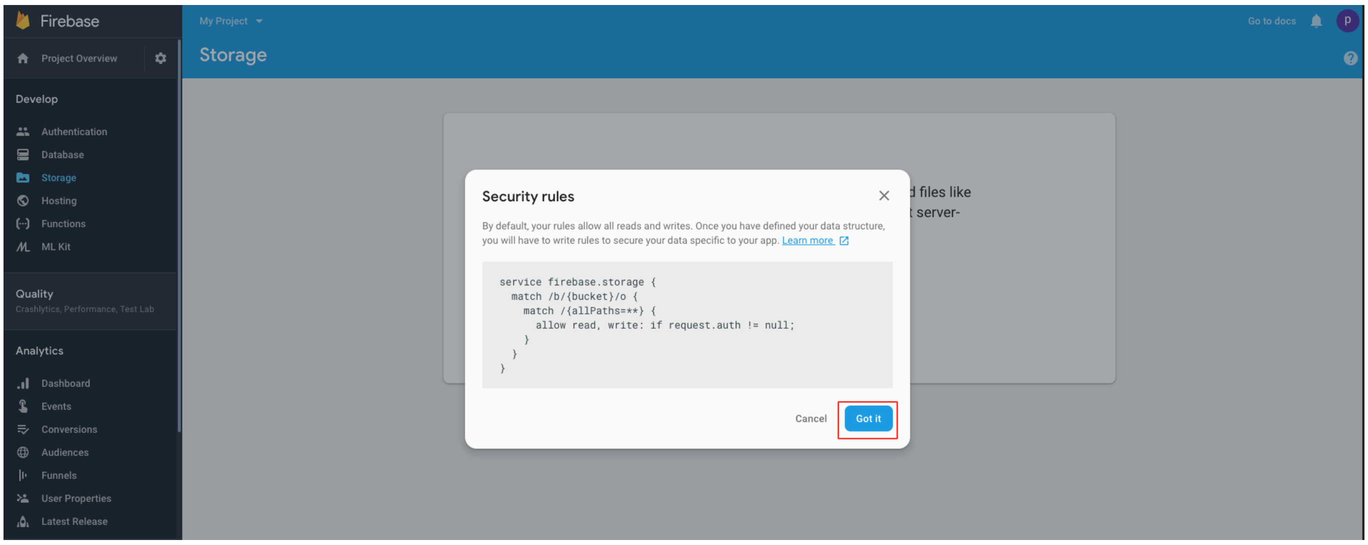
```
{
  "rules": {
    ".read": "true",
    ".write": "true"
  }
}
```

Step 17 : click on **Get Started**



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

Step 18 : Click on **Got It**

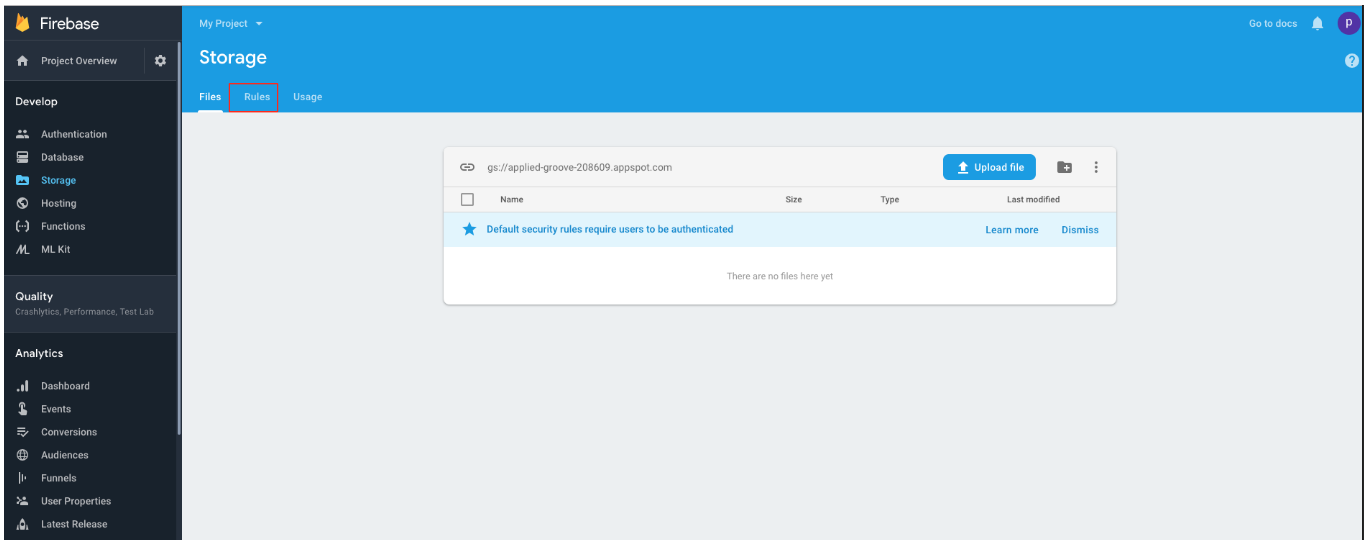


Security rules

By default, your rules allow all reads and writes. Once you have defined your data structure, you will have to write rules to secure your data specific to your app. Learn more

```
service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      allow read, write: if request.auth != null;
    }
  }
}
```

Step 19 : Switch to **Rules** tab



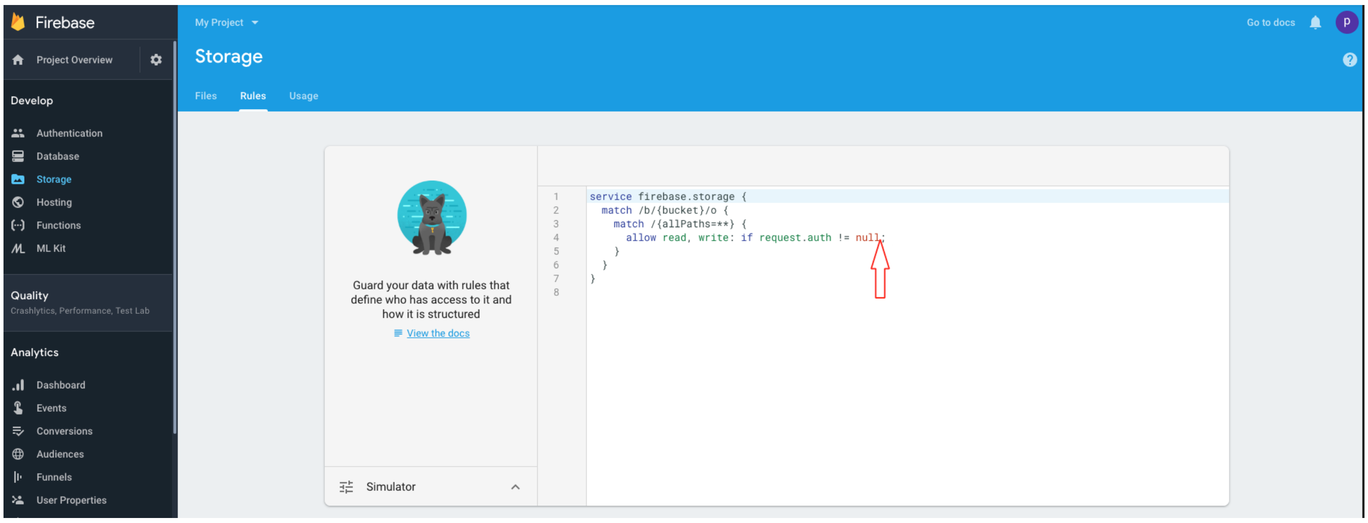
gs://applied-groove-208609.appspot.com

Name	Size	Type	Last modified

Default security rules require users to be authenticated

There are no files here yet

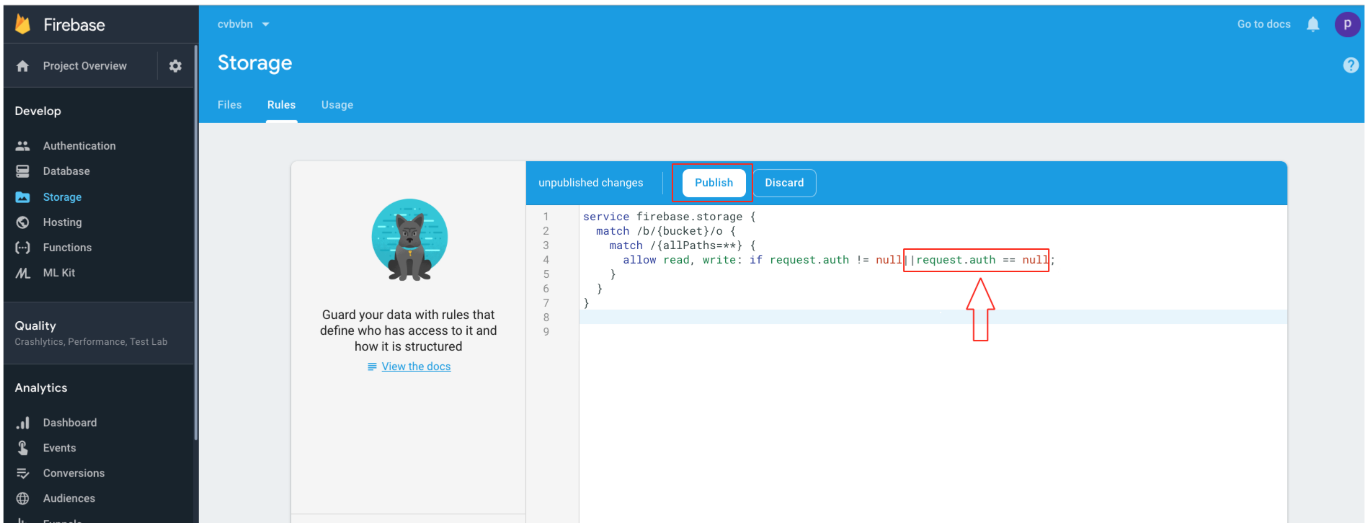
Step 20 : Add "|| request.auth == null" at the last in to the code



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

```
service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      allow read, write: if request.auth != null;
    }
  }
}
```

Step 21 : The code(same as into below image), then click on **Publish** button



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

```
service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      allow read, write: if request.auth != null||request.auth == null;
    }
  }
}
```

Receive new Angular 8 tutorials.

In this tutorial, you'll be using Angular 7 with Firebase and Firestore to create a project that implements the common CRUD operations.

CRUD stands for Create, Read, Update and Delete that refer to the operations that most apps need to make against a database. In this example, the database is a Firestore database that exists on the cloud.

> **Note**: This tutorial works with Angular 6 and Angular 7.

Creating a Firebase Project and Firestore Database

Now that your project is generated, you need to proceed by creating a Firebase project and a Firestore database. For this matter, you simply need to go to the Firebase Console and create a new Firebase project.

Next head to the `Project Overview >Develop >Database` page and create a new Firestore database. For now, make sure you enable test mode so you don't need any permissions to access the database.

Installing and Adding Firebase to your Angular Project.

After creating your Firebase project and your Firestore database you next need to install the `firebase` and `@angular/fire` packages:

```
$ npm install --save firebase @angular/fire
```

Next go to your Firebase project overview then click on *web* and copy the *config* data.

Next, open the `environments/environment.ts` file in your Angular project and add the `firebaseConfig` object inside the `environment` object.

```
export const environment = {
  production: false,
  firebaseConfig : {
    apiKey: "YOUR_API_KEY",
    authDomain: "YOUR_AUTH_DOMAIN",
    databaseURL: "YOUR_DATABASE_URL",
    projectId: "YOUR_PROJECT_ID",
    storageBucket: "YOUR_STORAGE_BUCKET",
    messagingSenderId: "YOUR_MESSAGING_SENDER_ID"
  }
};
```

Finally, you have to setup Firebase in your project. Open the `src/app/app.module.ts` file and update it accordingly:

```
import { AngularFireModule } from '@angular/fire';
import { AngularFireDatabaseModule } from '@angular/fire/database';
import { environment } from '../environments/environment';

@NgModule({
        // [...]
    imports: [
        // [...]
        AngularFireModule.initializeApp(environment.firebaseConfig),
        AngularFireDatabaseModule
    ],
```

You simply import `AngularFireModule` and `AngularFireDatabaseModule` and you add them to the `imports` array of the main application module.

You also call the `initializeApp()` method of `AngularFireModule` to pass the configuration object that you added earlier to `environments/environment.ts` file.

That's it, you now have added Firebase and Firestore addded to your Angular 7 project.

Create an Angular 7 Model

After setting up Firestore in your project, you can proceed by creating a model class. In the simple example, we suppose that you are creating an insurance app where we need to manage a set of policies.

An insurance application will often contain more that one type of data like clients, employees and policies etc. In this example, we'll just focus on the policy entity:

Let's create a model for our insurance policy entity:

```
$ ng g class policy --type=model
```

Open the `src/policy.model.ts` file and update it accordingly:

```
export class Policy {
    id: string;
    policyNumber: string;
    creationDate: Date;
    effectiveDate: Date;
    expireDate: Date;
    paymentOption: string;
    policyAmount: number;
    extraInfo: string;
}
```

This is an example of an insurance policy with many fields and relationships with other entities omitted for the sake of simplicity.

Creating an Angular 7 Service

An Angular service allows you to encapsulate the code that could be repeated in many places in your project. Using the Angular CLI, run the following command to generate a service:

```
$ ng g service policy
```

Open the `src/policy.service.ts` file and update it accordingly. First, import `AngularFirestore` and the `Policy` model at the top of the file:

```
import { AngularFirestore } from '@angular/fire/firestore';
import { Policy } from 'src/app/policy.model';
```

Next, inject `AngularFirestore` in your service via its constructor:

```
export class PolicyService {
  constructor(private firestore: AngularFirestore) { }
}
```

Next, add the `getPolicies()` method to retrieve the available policies from the Firestore collection:

```
getPolicies() {
    return this.firestore.collection('policies').snapshotChanges();
}
```

You also need to add the `createPolicy()` method to persist an insurance policy in the Firestore database:

```
createPolicy(policy: Policy){
    return this.firestore.collection('policies').add(policy);
}
```

Next, you need to add the `updatePolicy()` method to update an insurance policy by its identifier:

```
updatePolicy(policy: Policy){
    delete policy.id;
    this.firestore.doc('policies/' + policy.id).update(policy);
}
```

Finally, you can add the `deletePolicy()` method to delete an insurance policy by its identifier:

```
deletePolicy(policyId: string){
    this.firestore.doc('policies/' + policyId).delete();
}
```

Creating the Component for making CRUD Operations

After creating the model and service for create, read, update and delete insurance policies, you now need to create the component for creating, updating, deleting and displaying them:

Using Angular CLI v7 run the following command to generate a component:

```
$ ng g c policy-list
```

Now, open the `src/app/policy-list/policy-list.component.ts` file and update it accordingly:

```
import { Component, OnInit } from '@angular/core';
import { PolicyService } from 'src/app/policy.service';
import { Policy } from 'src/app/policy.model';

@Component({
  selector: 'policy-list',
  templateUrl: './policy-list.component.html',
  styleUrls: ['./policy-list.component.css']
})
export class PolicyListComponent implements OnInit {

  policies: Policy[];
  constructor(private policyService: PolicyService) { }
  ngOnInit() {
    this.policyService.getPolicies().subscribe(data => {
      this.policies = data.map(e => {
        return {
          id: e.payload.doc.id,
          ...e.payload.doc.data()
        } as Policy;
      })
    });
  }

  create(policy: Policy){
      this.policyService.createPolicy(policy);
  }

  update(policy: Policy) {
    this.policyService.updatePolicy(policy);
  }

  delete(id: string) {
    this.policyService.deletePolicy(id);
  }
}
```

Updating the Component Template

Now let's update the component's template to display insurance policies and also display buttons that can be used to create, update and delete policies:

Open the `src/app/policy-list.component.html` file and add the following HTML code:

```
<table>
  <thead>
    <th>Number</th>
    <th>Created At</th>
    <th>Expire At</th>
    <th>Amount</th>
  </thead>
  <tbody>
    <tr *ngFor="let policy of policies">

      <td>{{policy.policyNumber}}</td>
      <td>{{policy.creationDate}}</td>
      <td>{{policy.expireDate}}</td>
      <td>{{policy.policyAmount}}</td>
      <td>
          <button (click)="delete(policy.id)">Delete</button>
      </td>
    </tr>
  </tbody>
</table>
```

Below the `<table>` markup, you can also add a form to create an insurance policy.

Conclusion

In this tutorial, we've seen how to add CRUD operations to your Angular 7 project that allow you to create, read, update and delete data from a Firebase/Firestore database.