

Firestore Workshop Session#1 March-2020

Firestore workshop by Cristian Arce & Alfredo Bonilla

Session #1

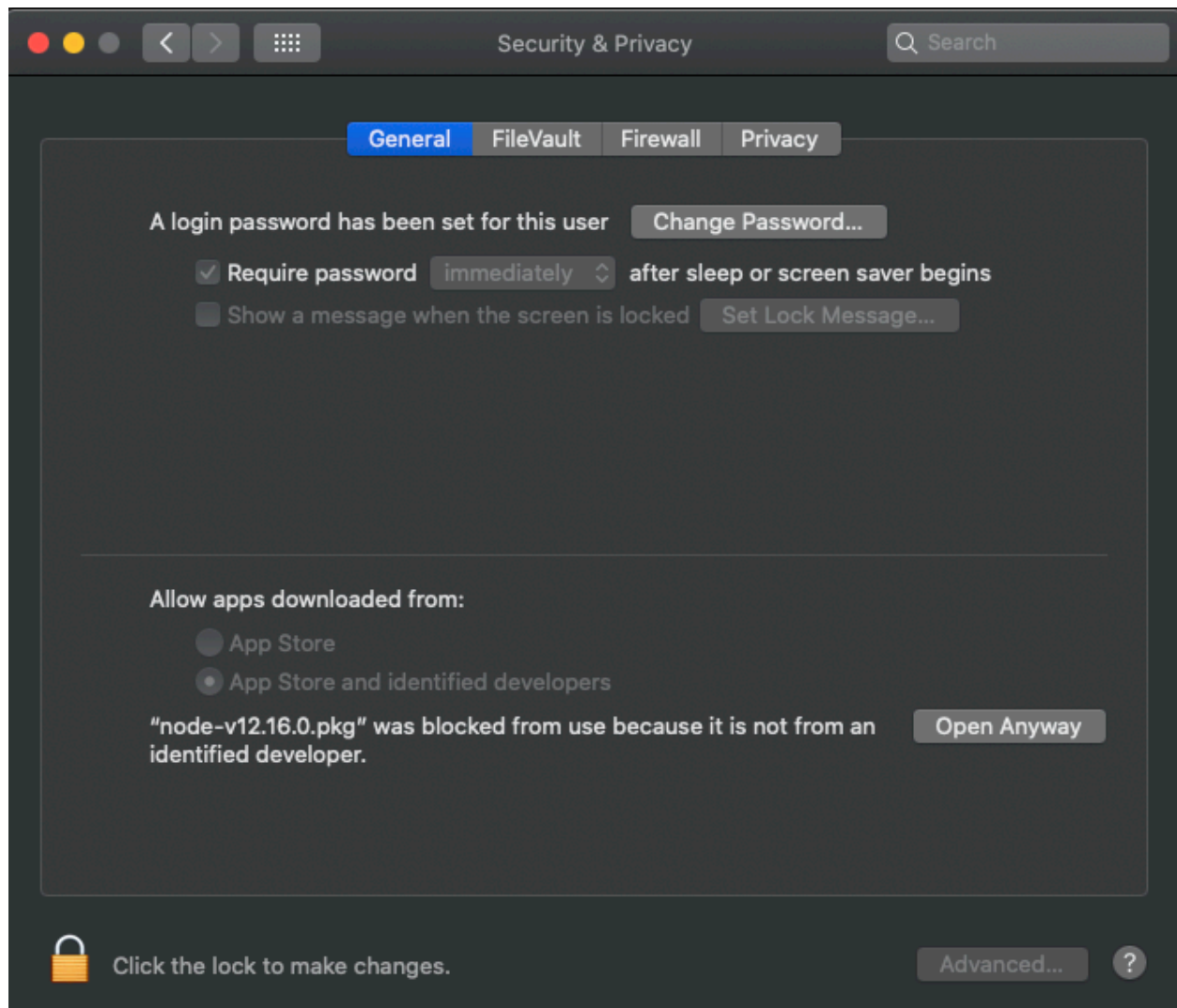
NodeJS and NPM Installation Steps

Step 1: Install **NodeJs** version for MacOS
from <https://nodejs.org/en/download/>

Note: In case of receiving an **error** such as: "***can't be opened because apple cannot check it for malicious software***":

Go to => **System Preferences**, click **Security & Privacy**, then click **General**

The node package is listed as "**node-v12.16.0.pkg**"(current version), click on "**Open Anyway**", this should allow you follow the installation steps.



Follow the installation wizard using the default values, that will finish **NodeJS** installation in your local machine, at the end of the process, check the node version installed running from terminal:

```
$ node -v
```

Step 2: Install/Update **npm** running from terminal

```
$ npm install npm@latest -g
```

Step 3: At the end of the process, check the **npm** version installed running from terminal

```
$ npm -v
```

Firestore Console Creation Steps

Step 1: Navigate to <https://firebase.google.com/>, it is required to access using a valid set of Google Credentials(gorilla account works)

After starting the session, there is the option of “**Add a Project**”, click there and the wizard will guide you through

Create a project:

First step of the wizard is to set a name to the project; it can be anything since it is only a label, the real id of the project will be auto-generated and displayed right below the **Project Name**

× Create a project (Step 1 of 3)

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

Project name

firebase-workshop

 fir-workshop-b4991

Google Analytics


The second step is to allow/deny the use of Google Analytics tools such as reporting, predictions, testing; for this workshop purposes either is ok, if you want to dig deeper later on your own, you can enable it.


✕ Create a project (Step 2 of 3)


Google Analytics for your Firebase project

Google Analytics is a free and unlimited analytics solution that enables targeting, reporting, and more in Firebase Crashlytics, Cloud Messaging, In-App Messaging, Remote Config, A/B Testing, Predictions, and Cloud Functions.


Google Analytics enables:


 A/B testing [?](#)

 User segmentation & targeting across
Firebase products [?](#)

 Predicting user behavior [?](#)

 Crash-free users [?](#)

 Event-based Cloud Functions triggers [?](#)

 Free unlimited reporting [?](#)

☒ Enable Google Analytics for this project
Recommended

[Previous](#)

[Continue](#)

Configure Analytics:

The third step is to configure the account attached for Google Analytics; you can select the **Default Account for Firebase**

Finally, click on **Create Project**

× Create a project (Step 3 of 3)

Configure Google Analytics

Choose or create a Google Analytics account ?

 Default Account for Firebase ▼

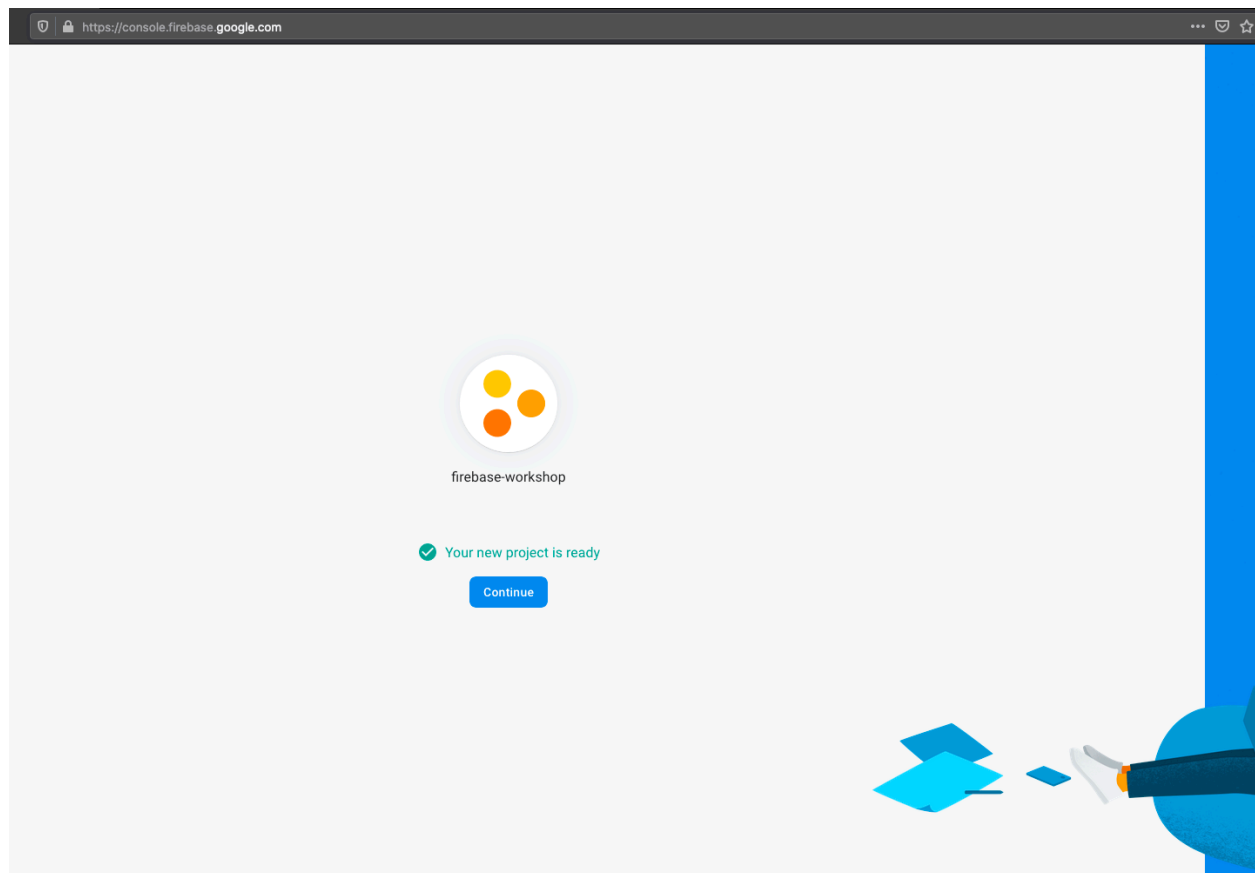
Automatically create a new property in this account ✎

Upon project creation, a new Google Analytics property will be created in your chosen Google Analytics account and linked to your Firebase project. This link will enable data flow between the products. Data exported from your Google Analytics property into Firebase is subject to the Firebase terms of service, while Firebase data imported into Google Analytics is subject to the Google Analytics terms of service. [Learn more](#).

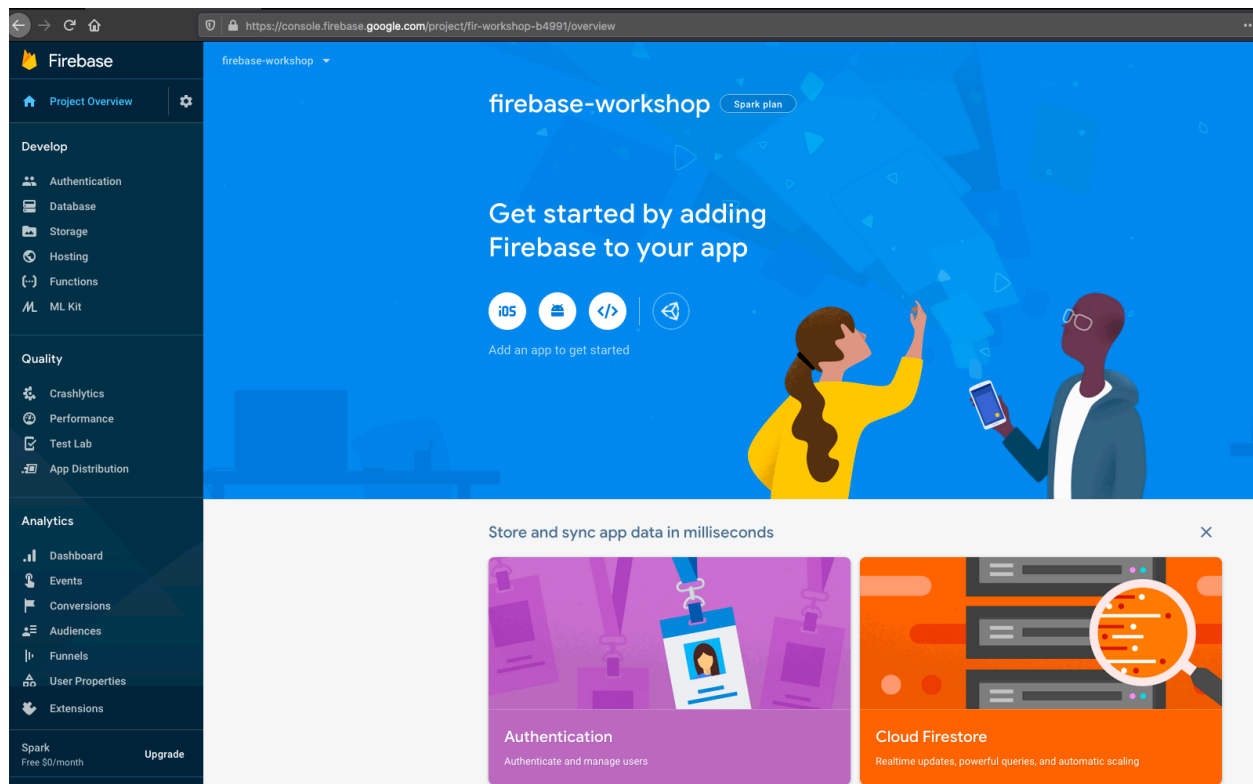
[Previous](#)

Create project

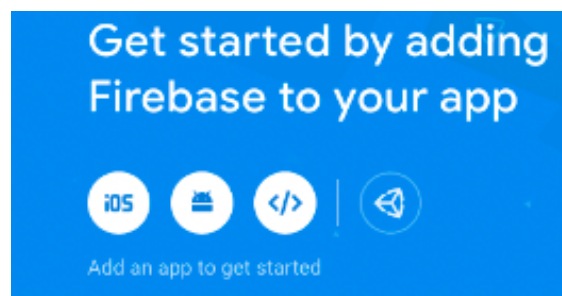
After a few seconds, the project is ready.



By default, we are using the free plan, it is called “**Spark Plan**”.



Step 2: It is required to add the Firebase configuration to a target; since it is a multiplatform tool, **iOS**, **Android** and **Web** are available; for this workshop purpose, the web option is the selected one(</>)



Firstly, an app nickname must be introduced, it can be the previous name of the label; also, check the option of “**Also set up Firebase Hosting for this app**”


← → ↻ 🏠 <https://console.firebase.google.com/project/fir-workshop-b4991/overview>

× Add Firebase to your web app


1 Register app

App nickname ?

firebase-workshop

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

Hosting can also be set up later. It's free to get started anytime.

 fir-workshop-b4991 (No deploys yet) ▼

Register app

2 Add Firebase SDK

Second, click “**Next**” in the “**Add Firebase SDK**”

× Add Firebase to your web app

✓ Register app

2 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="/__/firebase/7.8.2/firebase-app.js"></script>

<!-- TODO: Add SDKs for Firebase products that you want to use
https://firebase.google.com/docs/web/setup#available-libraries -->
<script src="/__/firebase/7.8.2/firebase-analytics.js"></script>

<!-- Initialize Firebase -->
<script src="/__/firebase/init.js"></script>
```



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

Next

3 Install Firebase CLI

4 Deploy to Firebase Hosting

Then, execute “***npm install -g firebase-tools***” and click “**Next**”

× Add Firebase to your web app

✓ Register app

✓ Add Firebase SDK

3 Install Firebase CLI

To host your site with Firebase Hosting, you need the Firebase CLI (a command line tool).

Run the following [npm](#) [🔗](#) command to install the CLI or update to the latest CLI version.

```
$ npm install -g firebase-tools
```



Doesn't work? Take a look at the [Firebase CLI reference](#) [🔗](#) or change your [npm permissions](#) [🔗](#)


Next

4 Deploy to Firebase Hosting

Last, click on “***Continue to console***” button, we are not running the other commands just yet.

× Add Firebase to your web app

- ✓ Register app
- ✓ Add Firebase SDK
- ✓ Install Firebase CLI
- 4 Deploy to Firebase Hosting

You can deploy now or [later](#) . To deploy now, open a terminal window, then navigate to or create a root directory for your web app.

Sign in to Google

```
$ firebase login
```



Initiate your project

Run this command from your app's root directory:

```
$ firebase init
```




When you're ready, deploy your web app

Put your static files (e.g., HTML, CSS, JS) in your app's deploy directory (the default is "public"). Then, run this command from your app's root directory:

```
$ firebase deploy
```

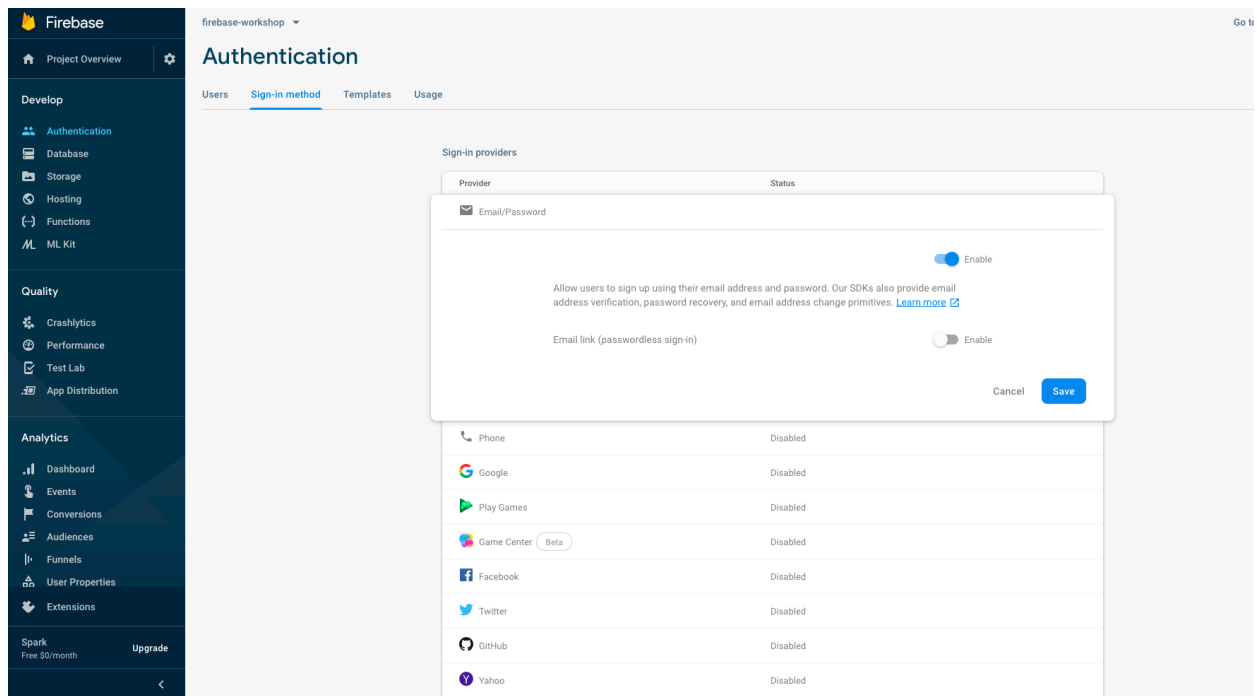


After deploying, view your app at fir-workshop-b4991.web.app 

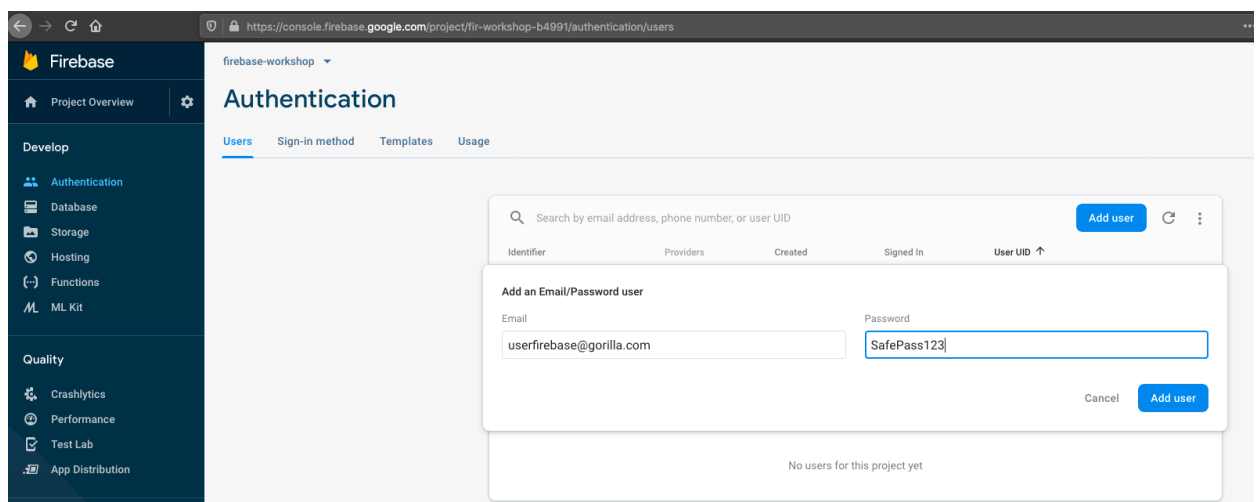
Need help? Check out the [Hosting docs](#) 

Continue to console

Step 3: Now it is necessary to enable the **Sign-in** providers; for this workshop purposes, only the **email/password** must be enabled



After enabling the provider, we must create a user for accessing the site. Go to **Authentication/Users** and add one manually; then save these credentials, they will be used in a while.

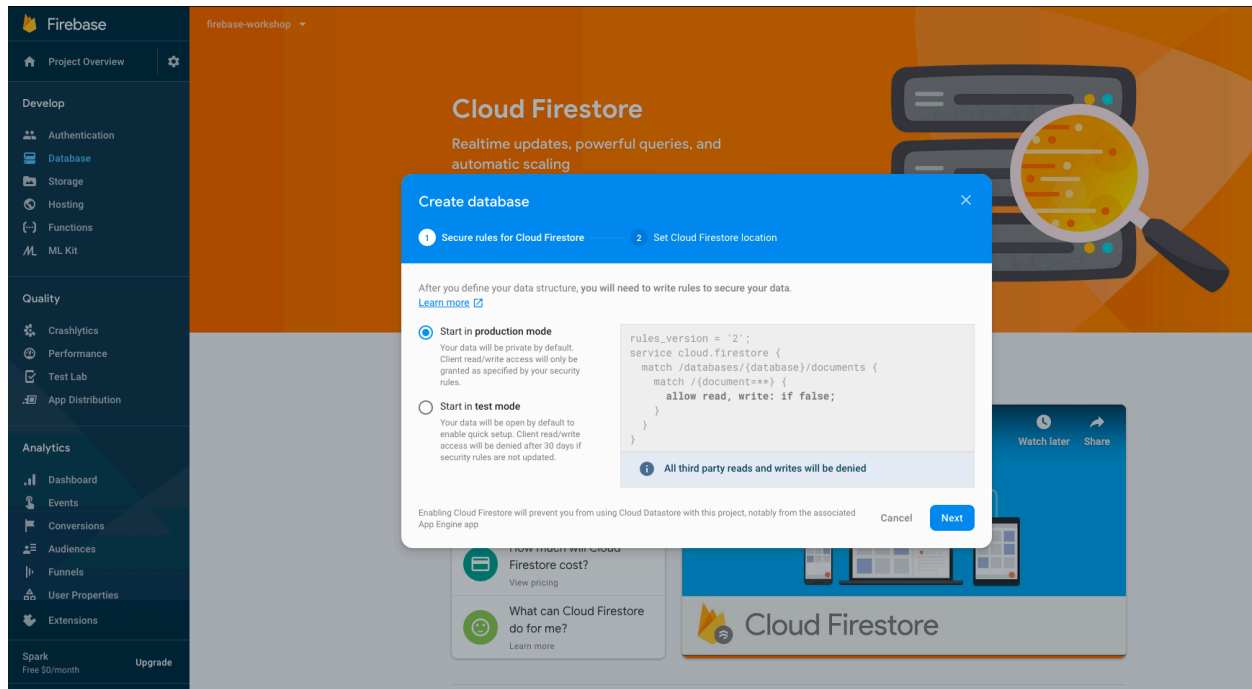


Step 4: Here comes the No-Sql!, **Firebase** enables two types

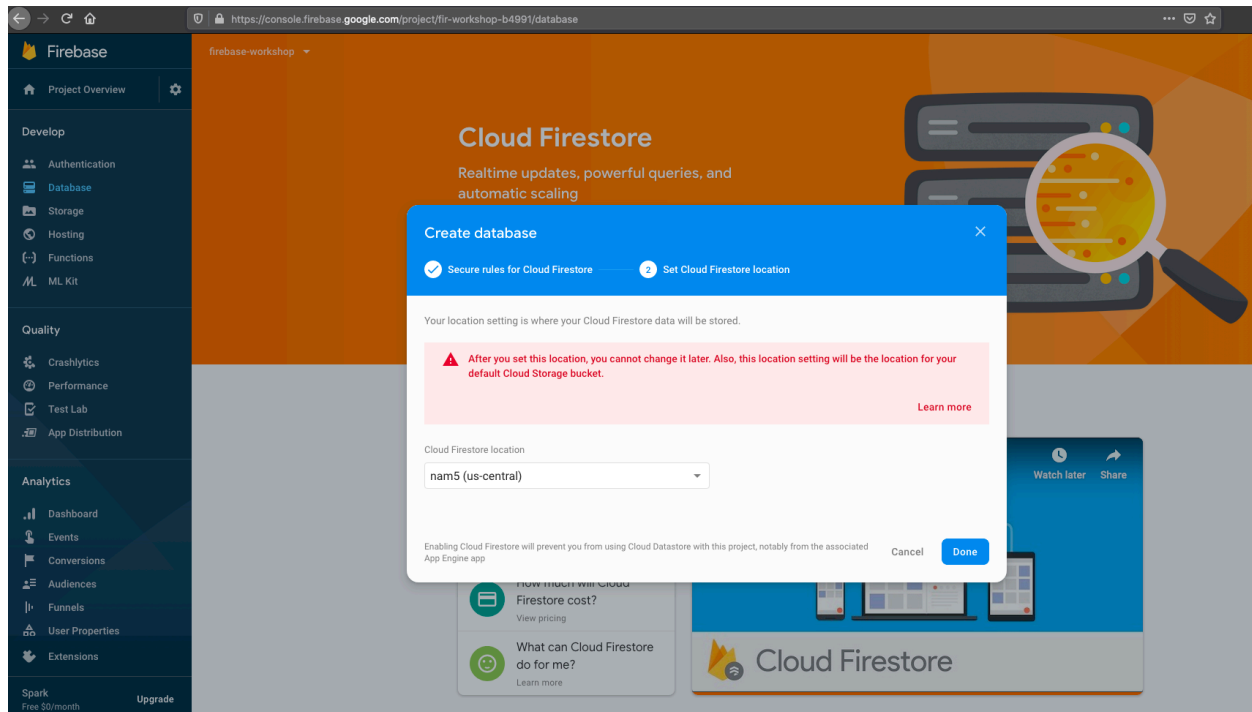
databases:

1. **Realtime**
2. **Cloud Firestore**

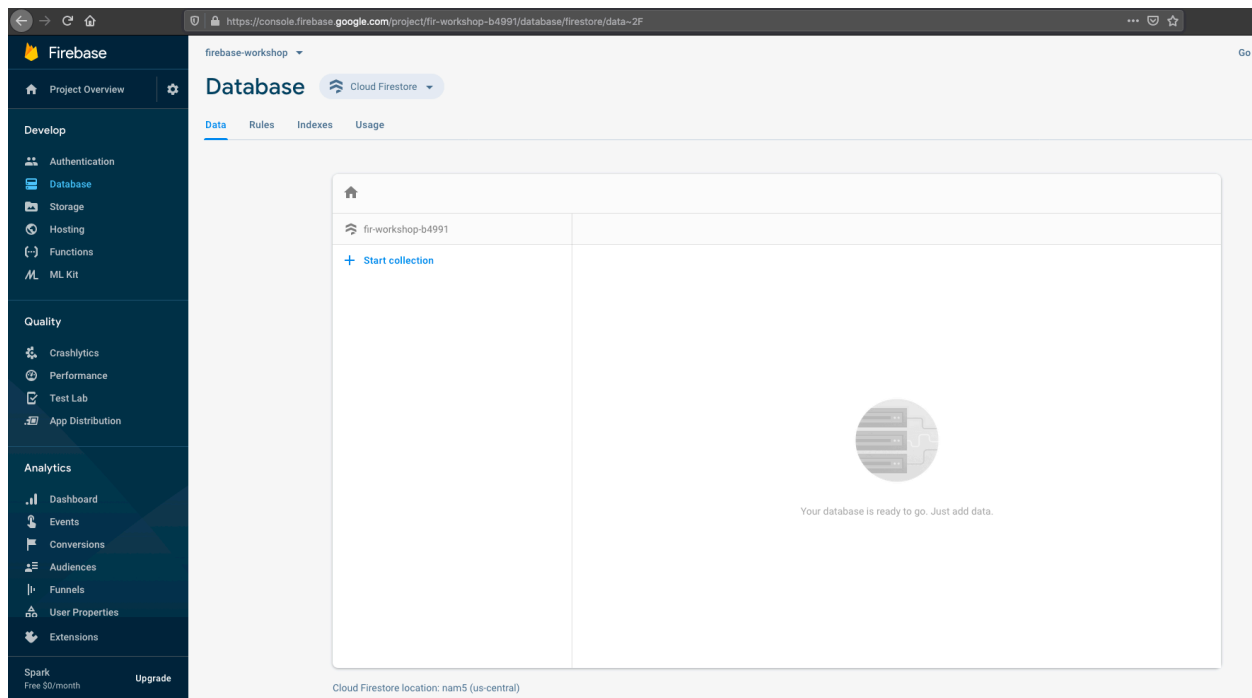
We will create the 2nd one, and start in **Production Mode**



It is also required to set a location; leave the default one



And now the database is ready to go!; by the moment we are leaving this empty cause we need to make the integration with the Angular +8 app using this configuration.



Angular Project

Step 1: Check your Angular CLI version running “*ng version*”, if a valid version is available, the output is something like:

```
$ > ng version
```



```
Angular CLI: 8.3.25
Node: 12.14.0
OS: darwin x64
Angular: 8.2.14
... animations, common, compiler, compiler-cli, core, forms
... language-service, platform-browser, platform-browser-dynamic
... router
```

Package	Version

@angular-devkit/architect	0.803.25
@angular-devkit/build-angular	0.803.25
@angular-devkit/build-optimizer	0.803.25
@angular-devkit/build-webpack	0.803.25
@angular-devkit/core	8.3.25
@angular-devkit/schematics	8.3.25
@angular/cdk	8.2.3
@angular/cli	8.3.25
@angular/flex-layout	8.0.0-beta.27
@angular/material	8.2.3

@ngtools/webpack	8.3.25
@schematics/angular	8.3.25
@schematics/update	0.803.25
rxjs	6.4.0
typescript	3.4.5
webpack	4.39.2

In case of not having an **Ng Version**, install the latest one executing:

```
$ npm i @angular/cli@8.3.23
```

Step 2: Clone the base project from repo in a folder

```
$ git clone git@github.com:cristianarceGL/firebase-workshop.git
```

```
$ cd firebase-workshop
```

```
$ git checkout Session1-Base
```

```
$ git pull
```

After pulling the repo, it is mandatory to install all dependencies and run the app

```
$ npm install && npm start
```

After running this, open the browser using the url <http://localhost:4200/>, a login page must be displayed, that's the one we will be modifying from now

Step 3: move to the folder with the code base and install the dependencies of **Firebase** and **Angular Fire**

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

Step 4: Next go to your Firebase **Project Overview**, then click on web and copy the **Config** entry

The screenshot shows the Firebase Project Overview page for a project named 'firebase-workshop'. The left sidebar contains navigation links for Project Overview, Develop (Authentication, Database, Storage, Hosting, Functions, ML Kit), Quality (Crashlytics, Performance, Test Lab, App Distribution), Analytics (Dashboard, Events, Conversions, Audiences, Funnels, User Properties, Extensions), and Spark (Free \$0/month, Upgrade). The main content area is titled 'Web apps' and shows a list of web apps with 'firebase-workshop' selected. To the right, the 'App nickname' is 'firebase-workshop' and the 'App ID' is '1:915544983278:web:4d8faaa30f4eb0553e95ff'. Below this, the 'Linked Firebase Hosting site' is 'fir-workshop-b4991'. The 'Firebase SDK snippet' section shows three options: Automatic, CDN, and Config (selected). Below the options, it says 'Copy and paste these scripts into the bottom of your <body> tag, but before you use any Firebase services:'. A code block contains the following JavaScript snippet:

```
const firebaseConfig = {
  apiKey: "AIzaSyAKVlX8x6zpgKoAstyY9NDV4lgowm9psA",
  authDomain: "fir-workshop-b4991.firebaseio.com",
  databaseURL: "https://fir-workshop-b4991.firebaseio.com",
  projectId: "fir-workshop-b4991",
  storageBucket: "fir-workshop-b4991.appspot.com",
  messagingSenderId: "915544983278",
  appId: "1:915544983278:web:4d8faaa30f4eb0553e95ff",
  measurementId: "G-MDSX30LE6P"
};
```

At the bottom right of the snippet is a 'Remove this app' button. At the very bottom of the page, there is a 'Delete project' button.

Next, open both, the **environments/environment.ts** and

environments/environment.prod.ts files in your **Angular 8** project and add the **firebaseConfig** entry object inside the environment object.

```
export const environment = {  
  ...  
  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',  
  }  
};
```

Step 5: Move to src/features and add a new folder named “**firebase**”, inside of the folder, a file named “**firebase.module.ts**”, and add the following content

```
import { NgModule } from '@angular/core';  
import { AngularFireModule } from '@angular/fire';  
import { AngularFireAuthModule } from  
'@angular/fire/auth';  
import { AngularFireAuthGuard } from  
'@angular/fire/auth-guard';  
import { AngularFireDatabaseModule,  
AngularFireDatabase } from '@angular/fire/database';  
import {  
  AngularFirestore,  
  AngularFirestoreDocument,
```

```

    AngularFireModule,
    DocumentChangeAction,
  } from '@angular/fire/firestore';
  import { environment } from
    '@enviroments/environment';

  export { AngularFire };
  export { AngularFireDatabase };
  export { DocumentChangeAction };
  export { AngularFirestoreDocument };

  const modules = [AngularFirestoreModule,
    AngularFireDatabaseModule, AngularFireAuthModule];

  @NgModule({
    imports: [...modules,
      AngularFireModule.initializeApp(environment.firebase
        Config)],
    exports: [...modules, AngularFireModule],
    providers: [AngularFireAuthGuard],
  })
  export class FirebaseModule {}

```

Step 6: Now that we have the **Firebase Module** created, we must import it into the **app.module.ts** file

```

...
...
import { FirebaseModule } from
  '@app/features/firebase/firebase.module';

```

```
@NgModule({
  ...
  imports: [BrowserModule, BrowserAnimationsModule,
    AppRoutingModule, LayoutModule, FirestoreModule],
  ...
})
export class AppModule {}
```

Step 7: Now we proceed to override the **app-routing.module.ts** with the following content

```
import { NgModule } from '@angular/core';
import { redirectUnauthorizedTo } from
  '@angular/fire/auth-guard';
import { Routes, RouterModule, PreloadAllModules }
  from '@angular/router';
import { canActivate, redirectLoggedInTo } from
  '@angular/fire/auth-guard';

const redirectLoggedInToItems =
  redirectLoggedInTo(['user']);
const redirectUnauthorizedToLogin =
  redirectUnauthorizedTo(['login']);

const routes: Routes = [
  { path: '', redirectTo: 'login', pathMatch:
    'full' },
```

```

        { path: 'login', loadChildren:
'@app/features/login/login.module#LoginModule',
...canActivate(redirectLoggedInToItems), },
        { path: 'user', loadChildren: () =>
import('@app/features/admin-user/admin-user.module')
        .then(mod => mod.AdminUserModule),
...canActivate(redirectUnauthorizedToLogin),
        },
];

@NgModule({
  imports: [
    RouterModule.forRoot(routes, {
      preloadingStrategy: PreloadAllModules,
    }),
  ],
  exports: [RouterModule],
})
export class AppRoutingModule {}

```

Step 8: Now that we have the **Firestore Module** created, we must import it also into the **login.module.ts** file

```

...
...

import { FirestoreModule } from
'@app/features/firebase/firebase.module';

```

```
const modules = [  
  
  CommonModule,  
  LoginRoutingModule,  
  RouterModule,  
  HttpClientModule,  
  MaterialModule,  
  ReactiveFormsModule,  
  FirestoreModule,  
];  
  
...  
...  
export class LoginModule {}
```

Step 9: Last but no less, we must override the **login.service.ts** fo using **Firestore** functions

```
import { Observable } from 'rxjs';  
import { map } from 'rxjs/operators';  
import { Router } from '@angular/router';  
import { Injectable } from '@angular/core';  
import { AngularFireAuth } from  
  '@angular/fire/auth';  
import { User } from '../models/user';  
import { Authenticate } from  
  '../models/authenticate';
```

```
@Injectable({
  providedIn: 'root',
})
export class LoginService {

  public user: User;

  public get isAuthenticated$():
Observable<boolean> {
    return this.afAuth.authState.pipe(map(user
=> user !== null));
  }

  public get currentUser$(): Observable<User |
undefined> {
    return this.afAuth.authState.pipe(map(user
=> user));
  }

  constructor(private afAuth: AngularFireAuth,
public router: Router) {
    this.afAuth.authState.subscribe(user => {
      user !== null ? (this.user = user) :
this.router.navigate(['user']);
    });
  }

  public async login(authenticate: Authenticate):
Promise<void> {
    await this.afAuth.auth
```



```

    .signInWithEmailAndPassword(authenticate.email,
authenticate.password)
        .then(_ => this.router.navigate(['user']))
        .catch(_ => console.log('error while logging
user in'));
    }

    public async logOut(): Promise<void> {
        await this.afAuth.auth
            .signOut()
            .then(_ => console.log('user successfully
logged out'))
            .catch(_ => console.log('error while logging
user out'));
        this.router.navigate(['login']);
    }
}

```

Step 10: For testing the changes out, run the app and try to log into the site executing

```
$ npm start
```