Firebase Workshop Session#1 March-2020

Firebase workshop by Cristian Arce & Alfredo Bonilla

Session #1

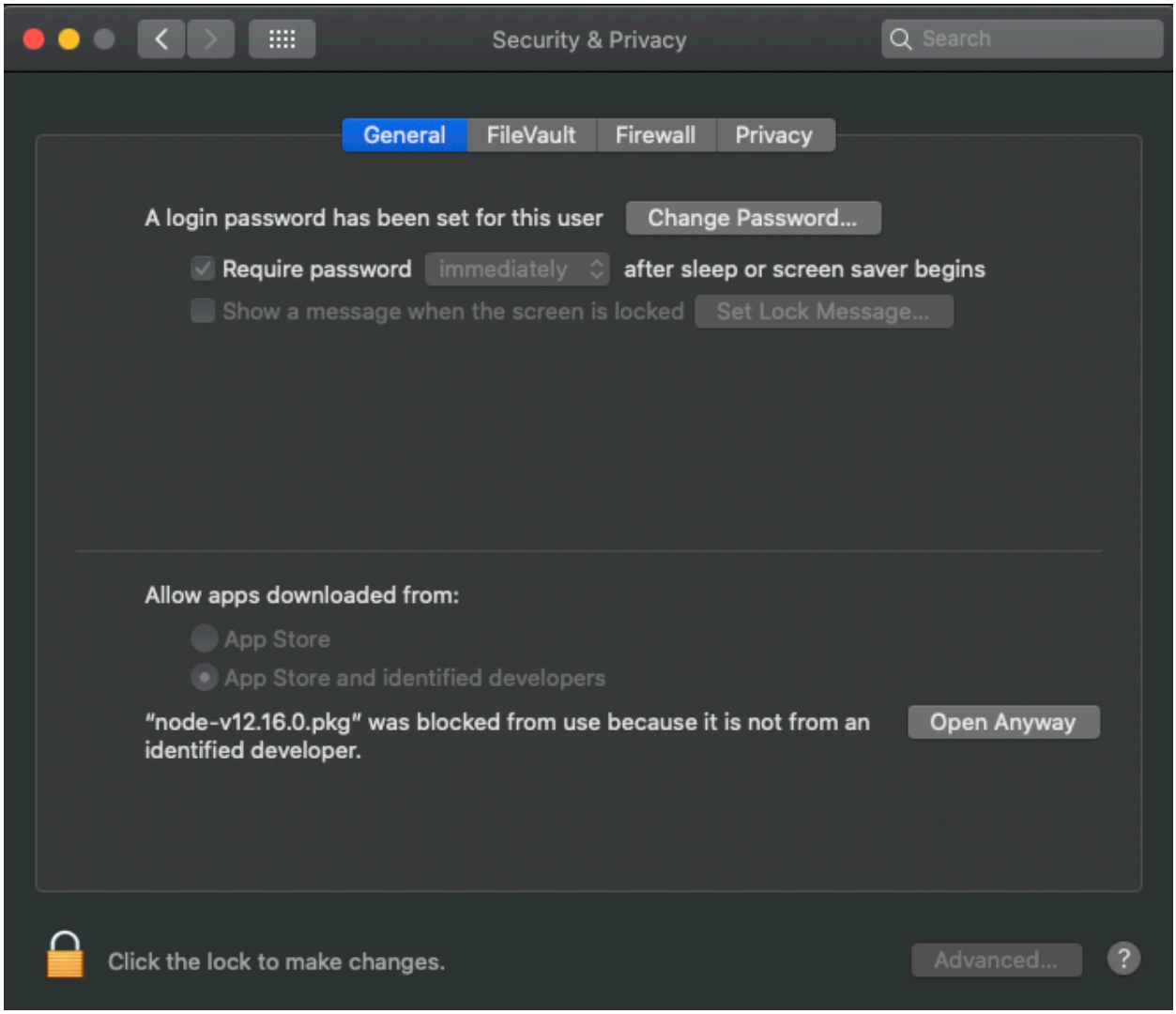
NodeJS and NPM Installation Steps

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
```

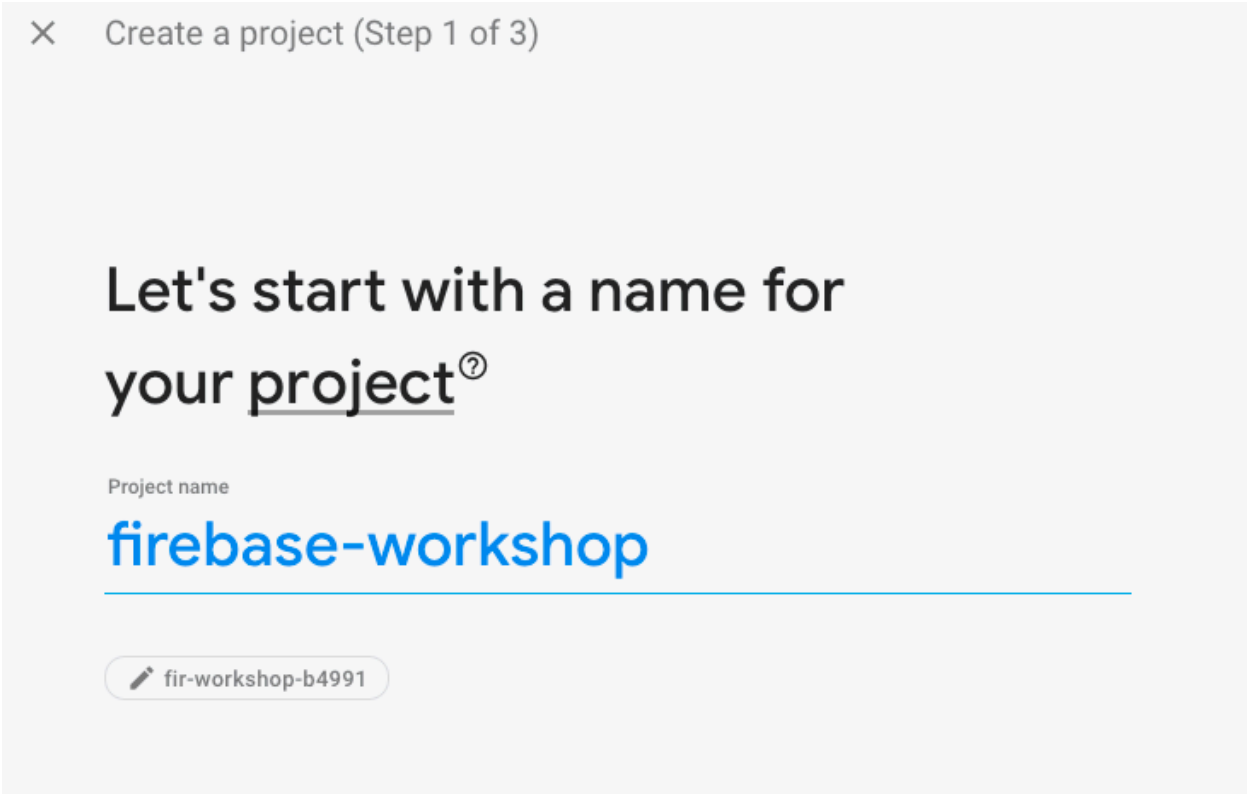
Firebase Console Creation Steps

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

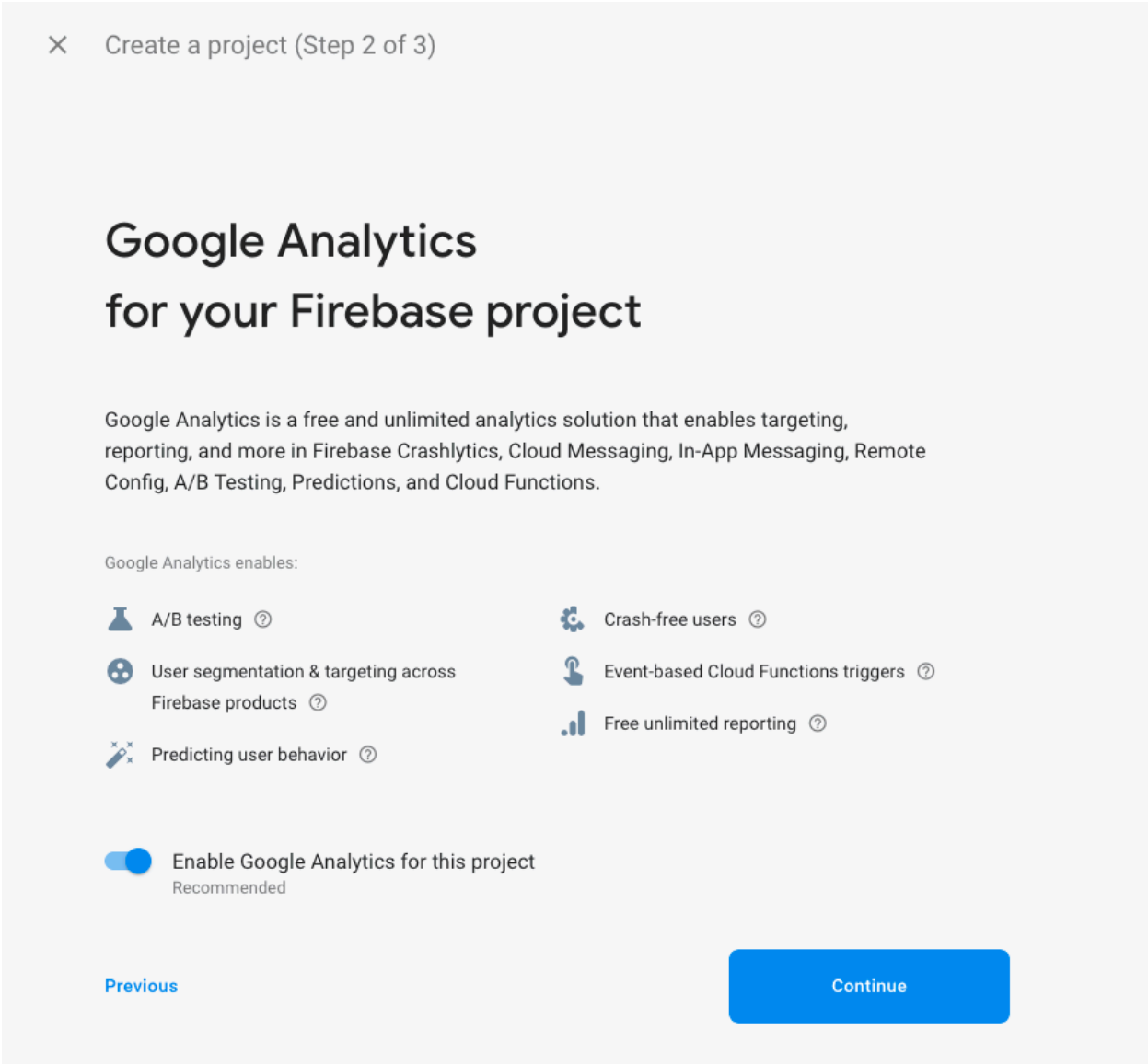
Create a project:

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**



Google Analytics

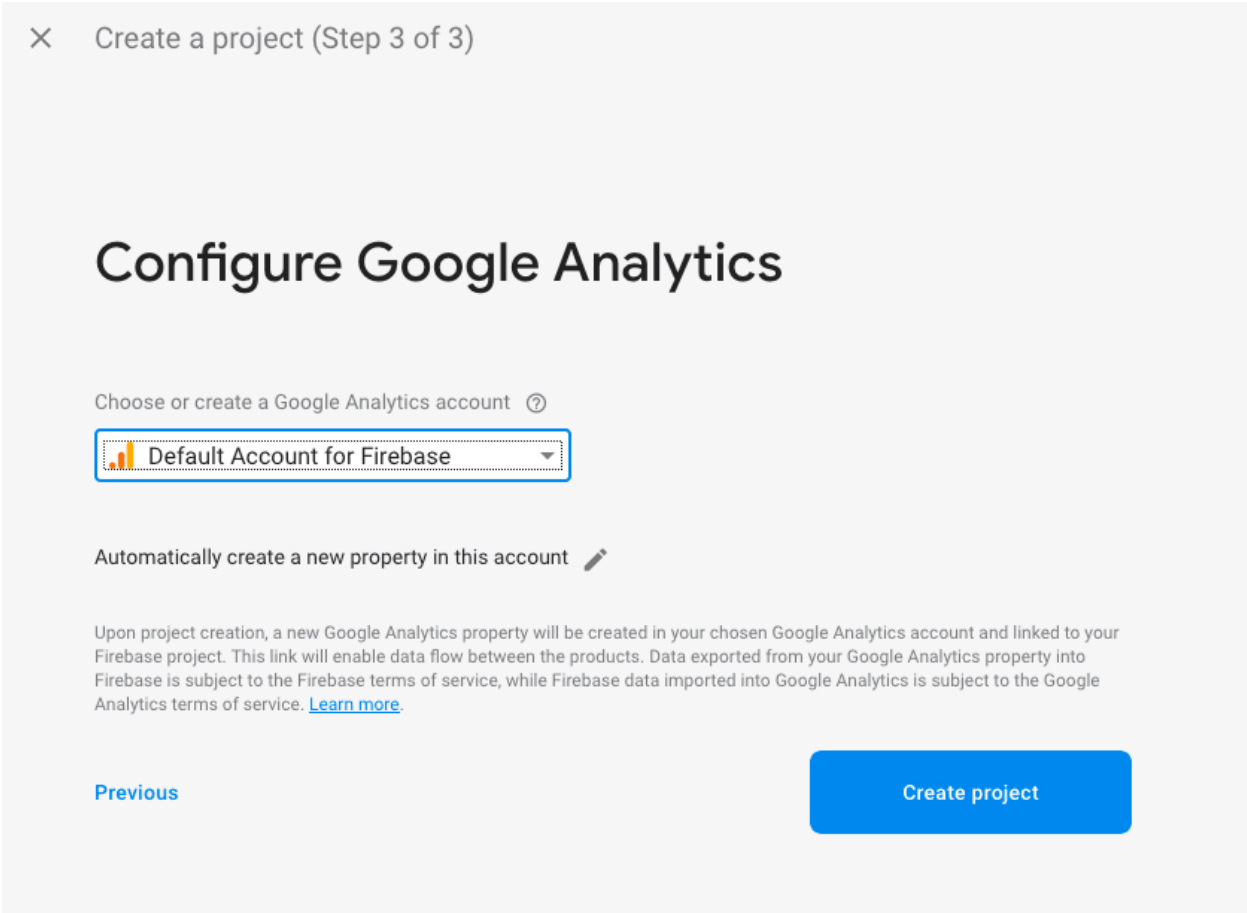
The second step is to allow/deny the use of Google Analytics tools such as reporting, predictions, testing; for this workshop purposes either is ok, if you want to dig deeper later on your own, you can enable it.



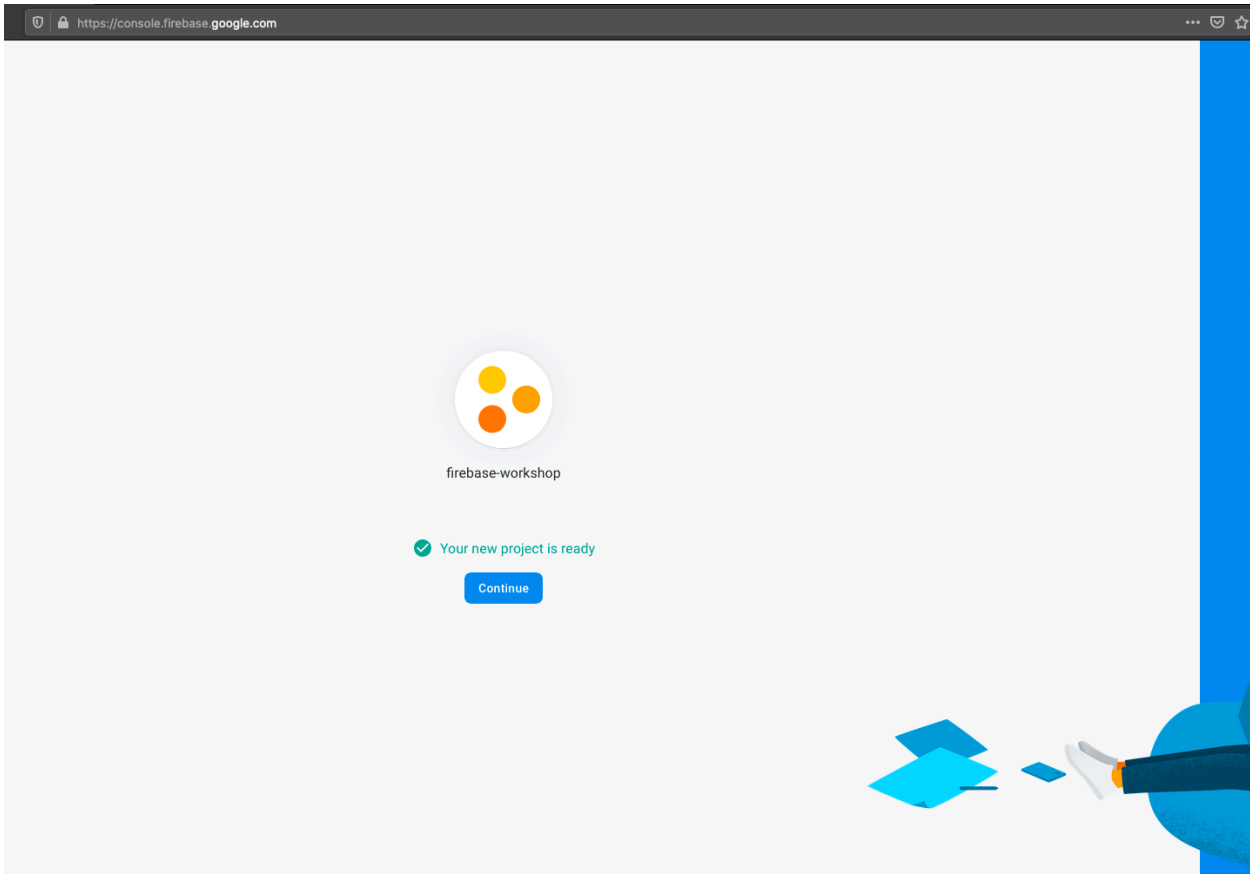
<u>Configure Analytics:</u>

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

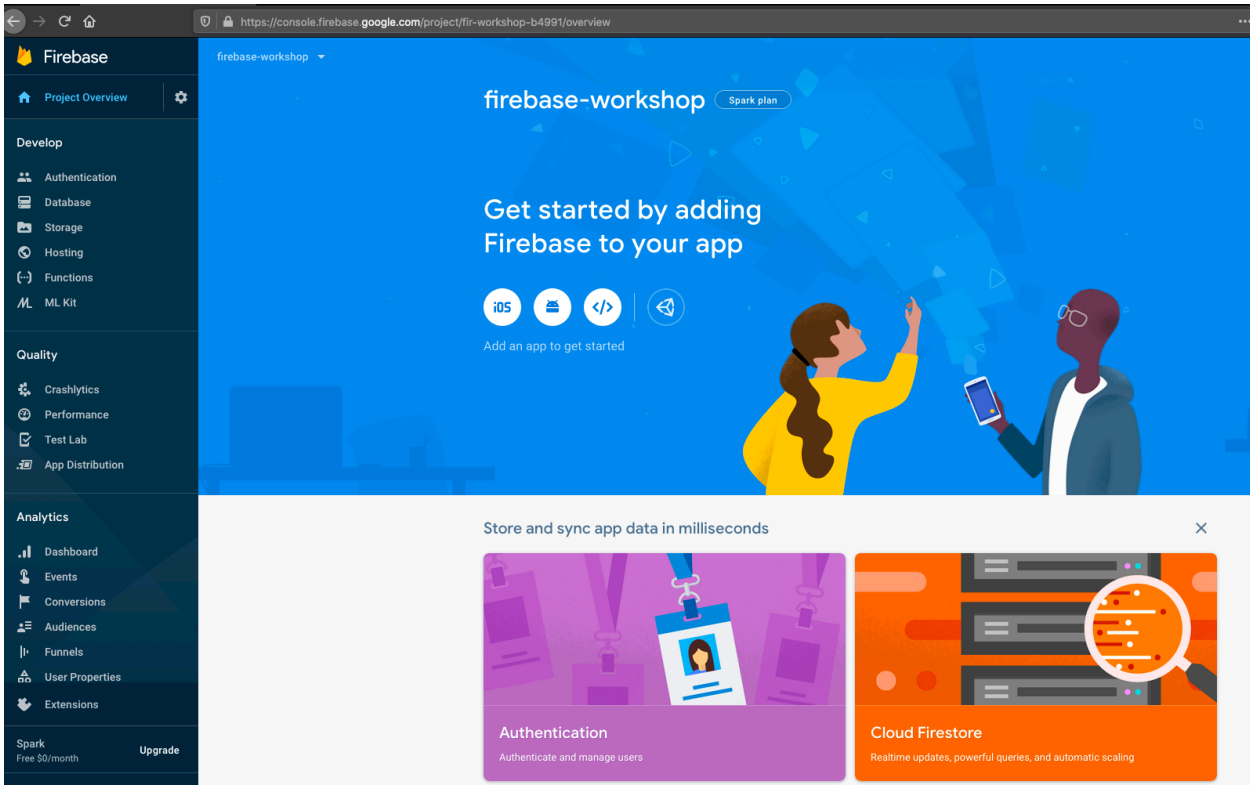
Finally, click on ***Create Project***



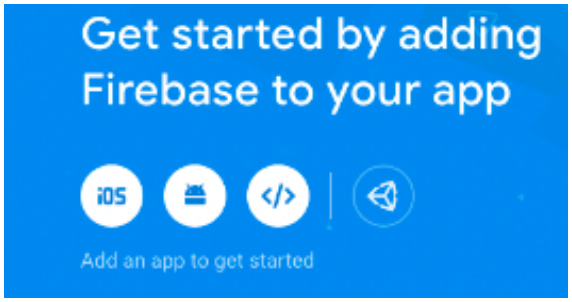
After a few seconds, the project is ready.



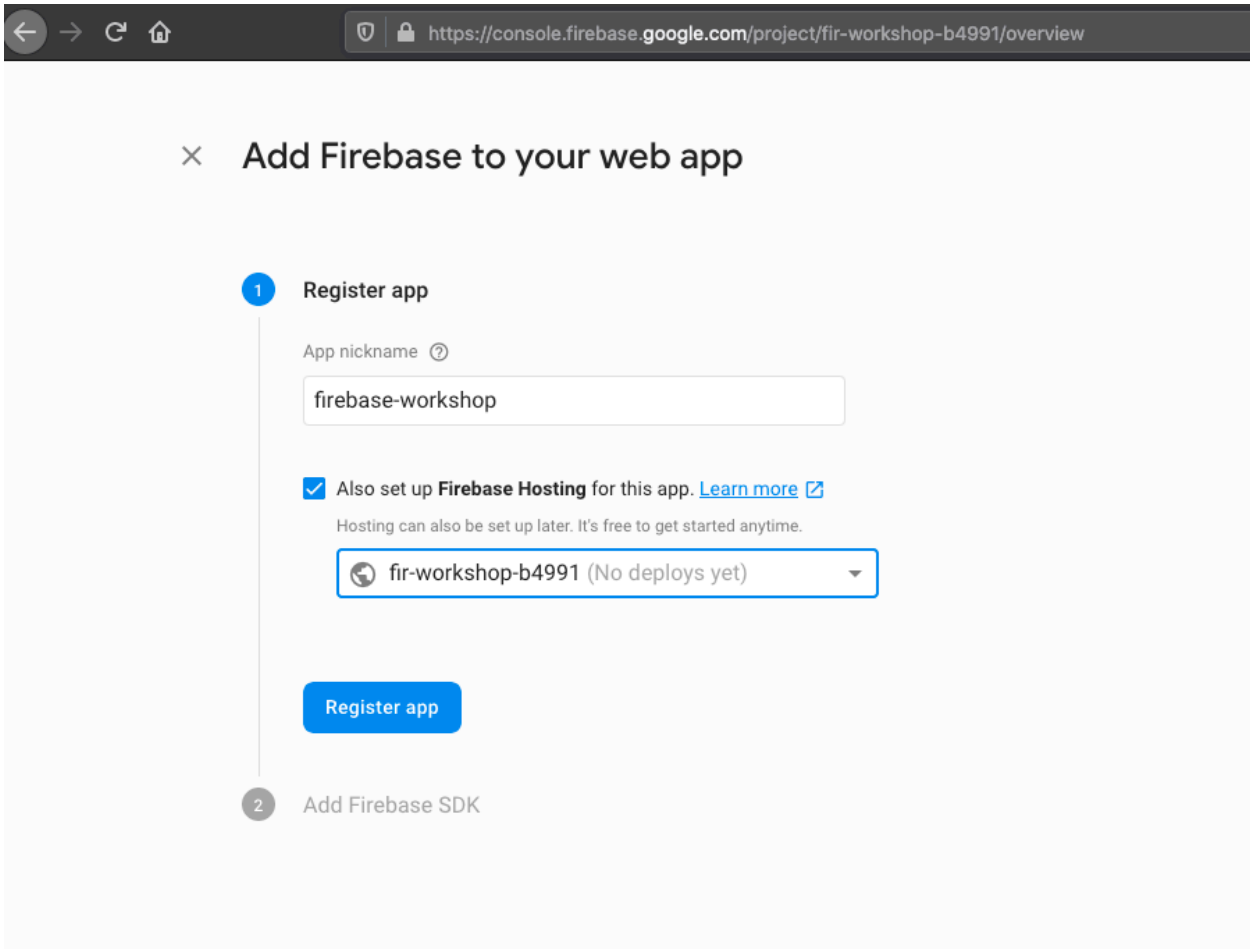
By default, we are using the free plan, it is called “**Spark Plan**”.



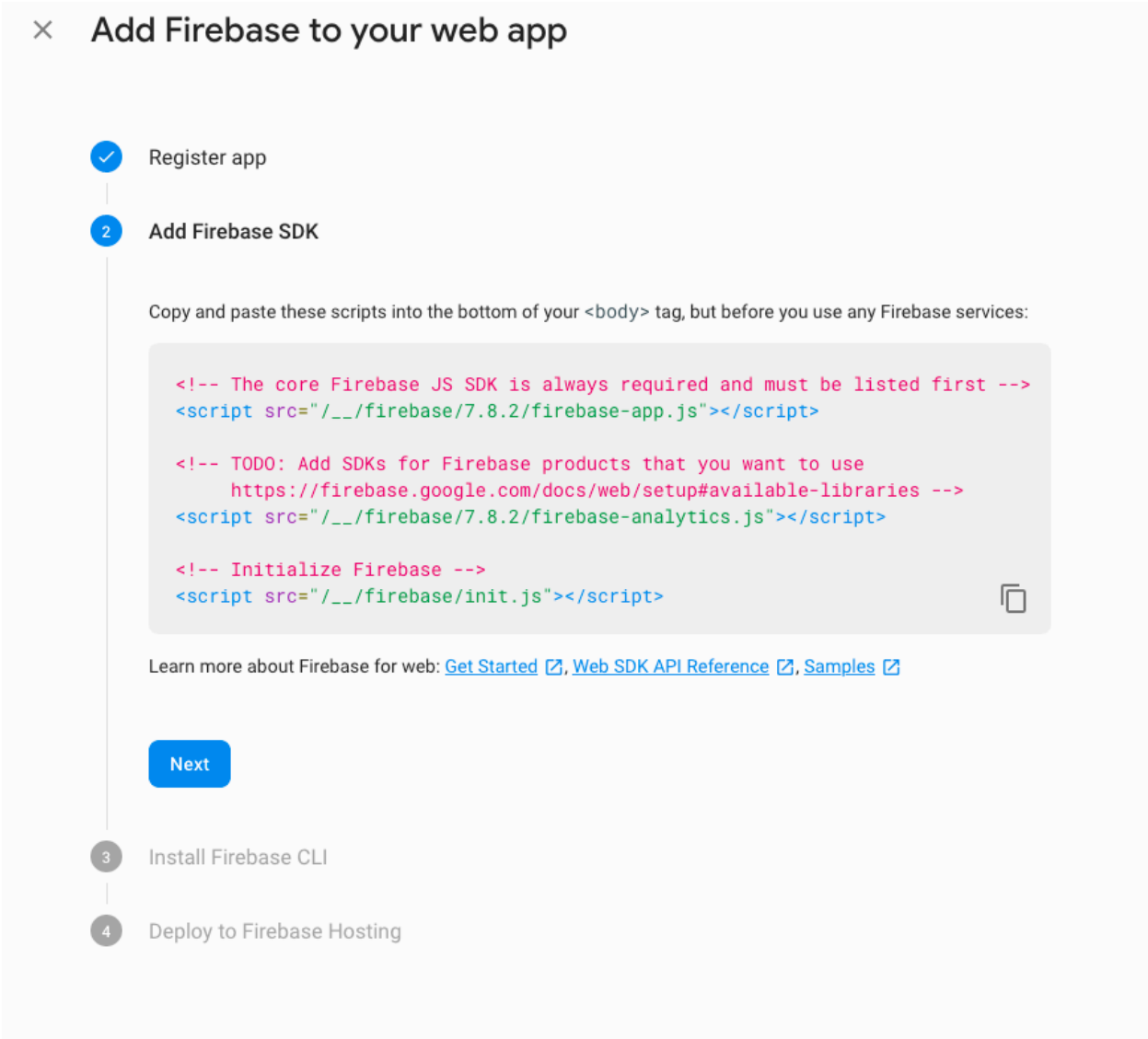
Step 2: It is required to add the Firebase configuration to a target; since it is a multiplatform tool, **iOS**, **Android** and **Web** are available; for this workshop purpose, the web option is the selected one(**</>**)



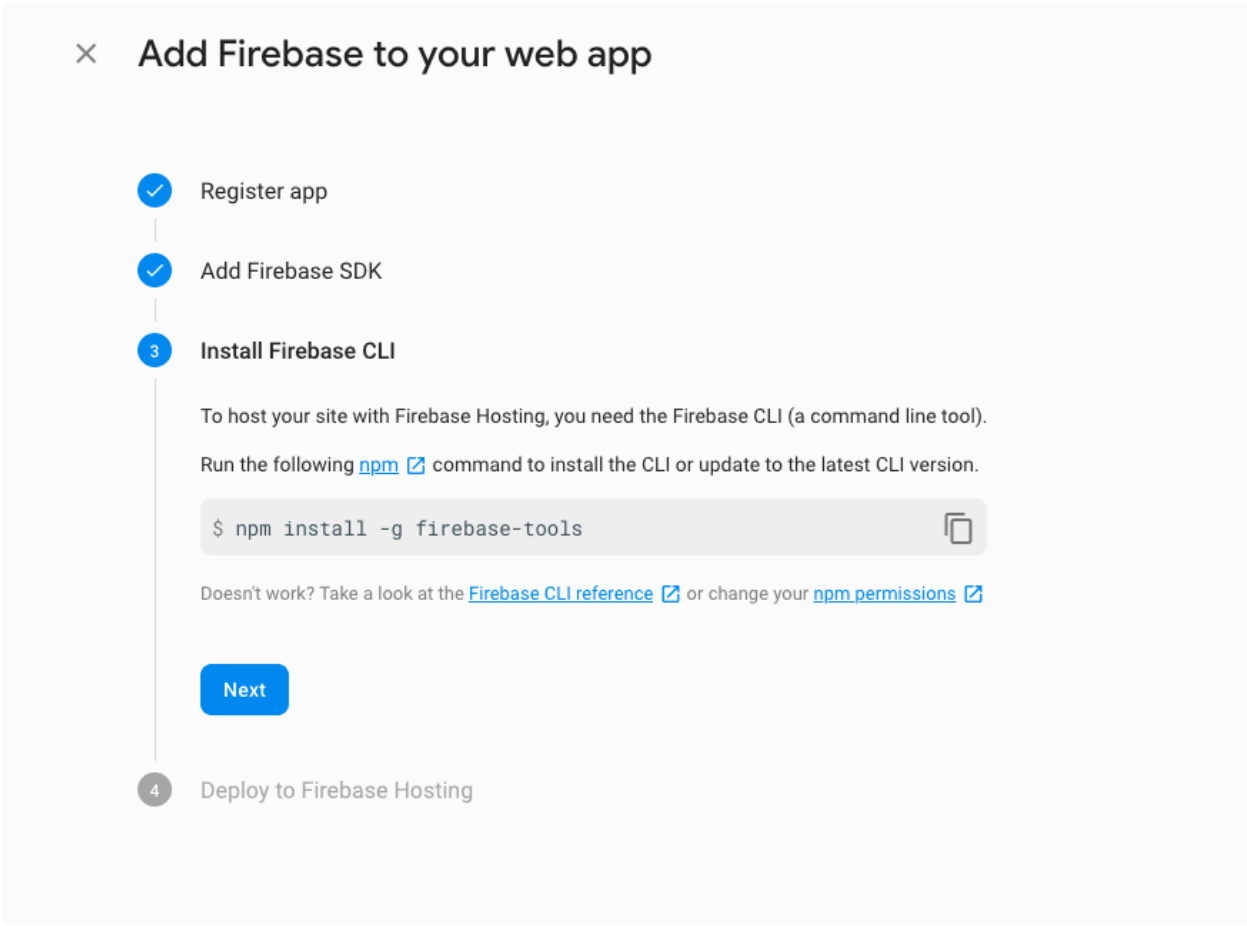
Firstly, an app nickname must me introduced, it can be the previous name of the label; also, check the option of "***Also set up Firebase Hosting for this app***"



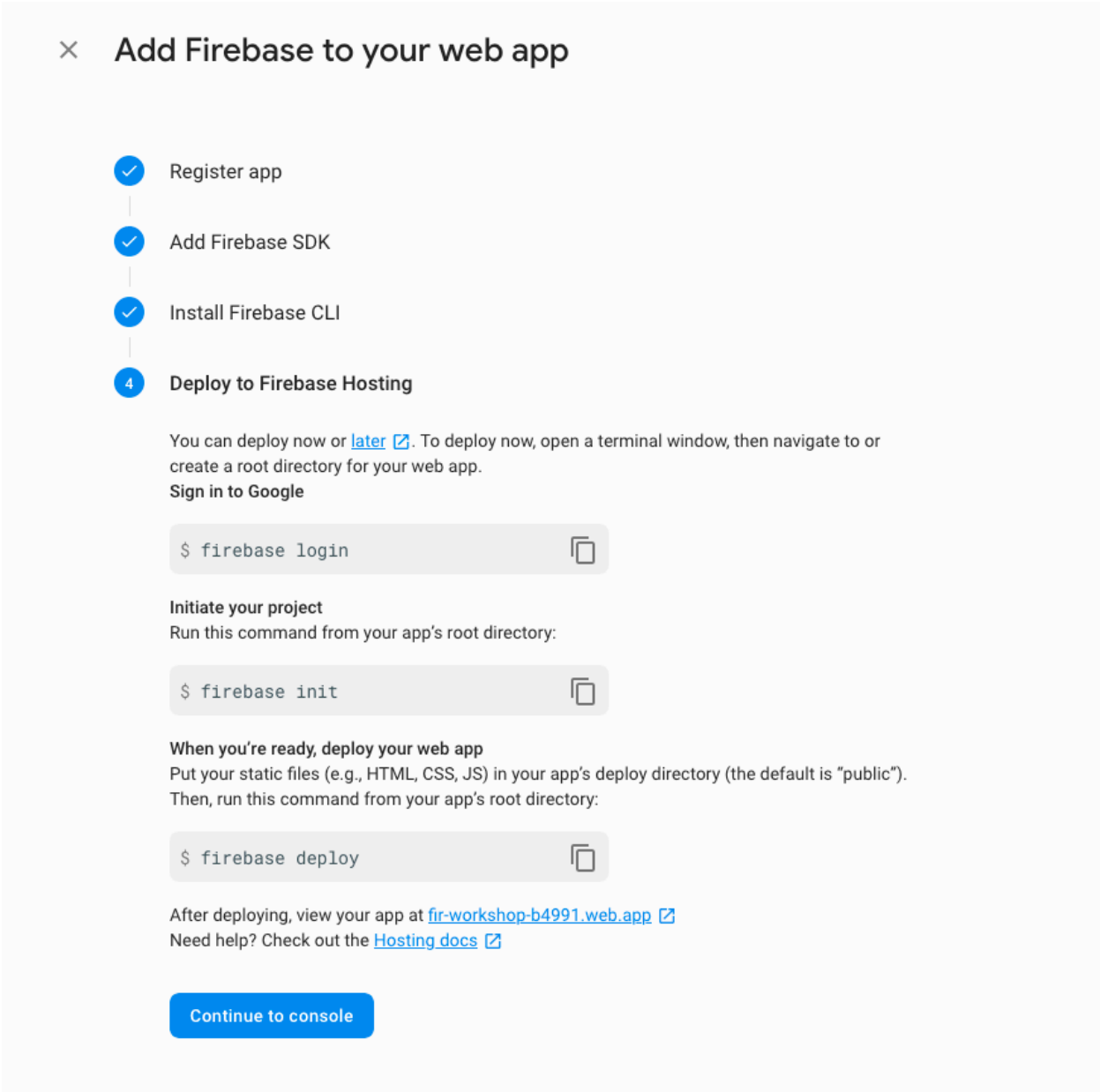
Second, click “**Next**” in the “***Add Firebase SDK***"



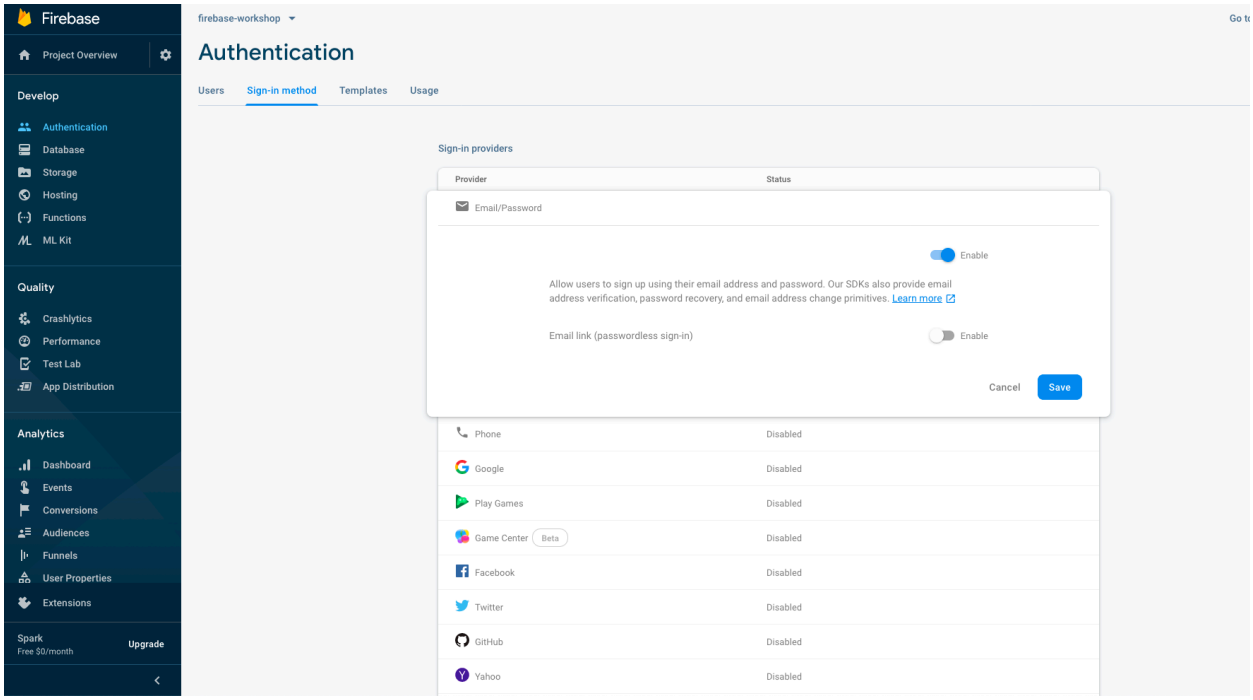
Then, execute “***npm install -g firebase-tools***” and click “**Next"**



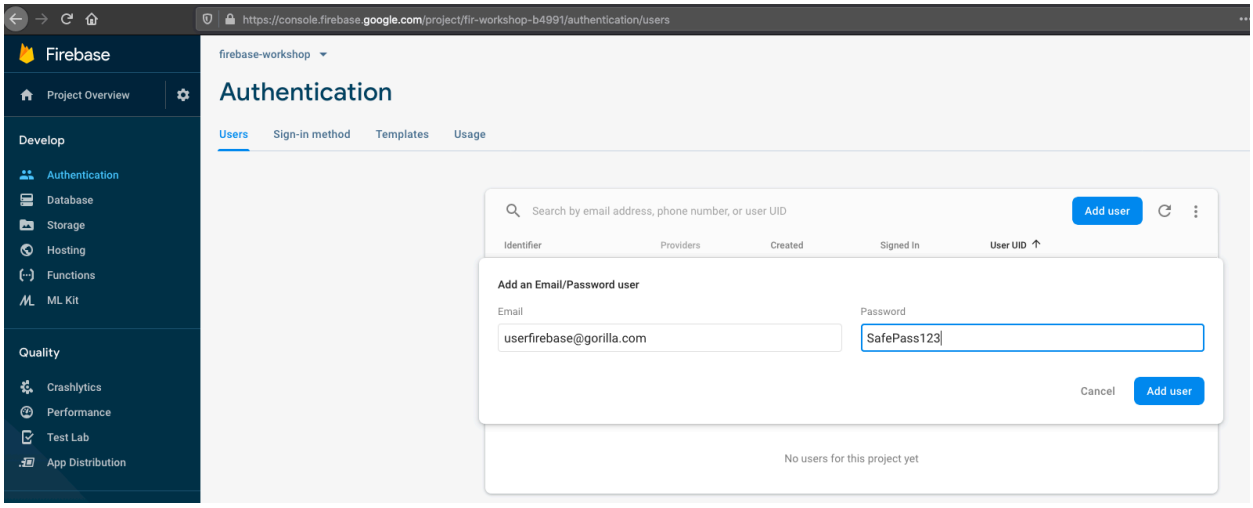
Last, click on "***Continue to console***" button, we are not running the other commands just yet.



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.

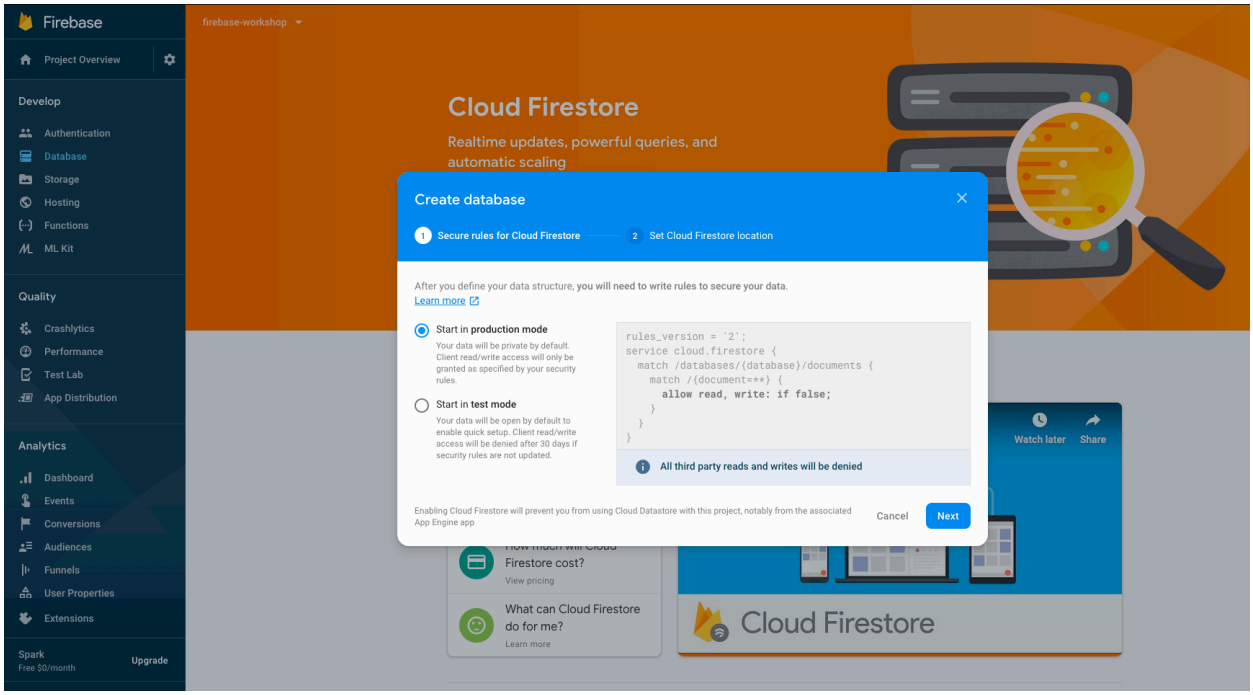


Step 4: Here comes the No-Sql!, **Firebase** enables two types

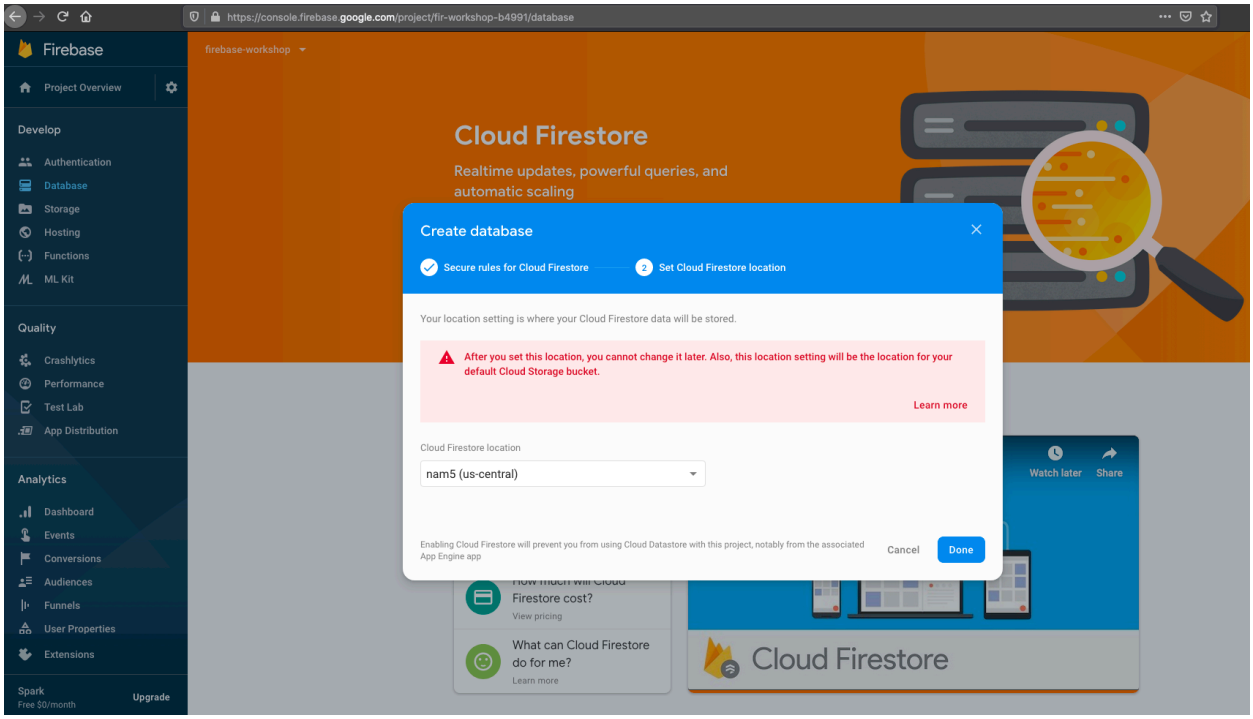
databases:

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

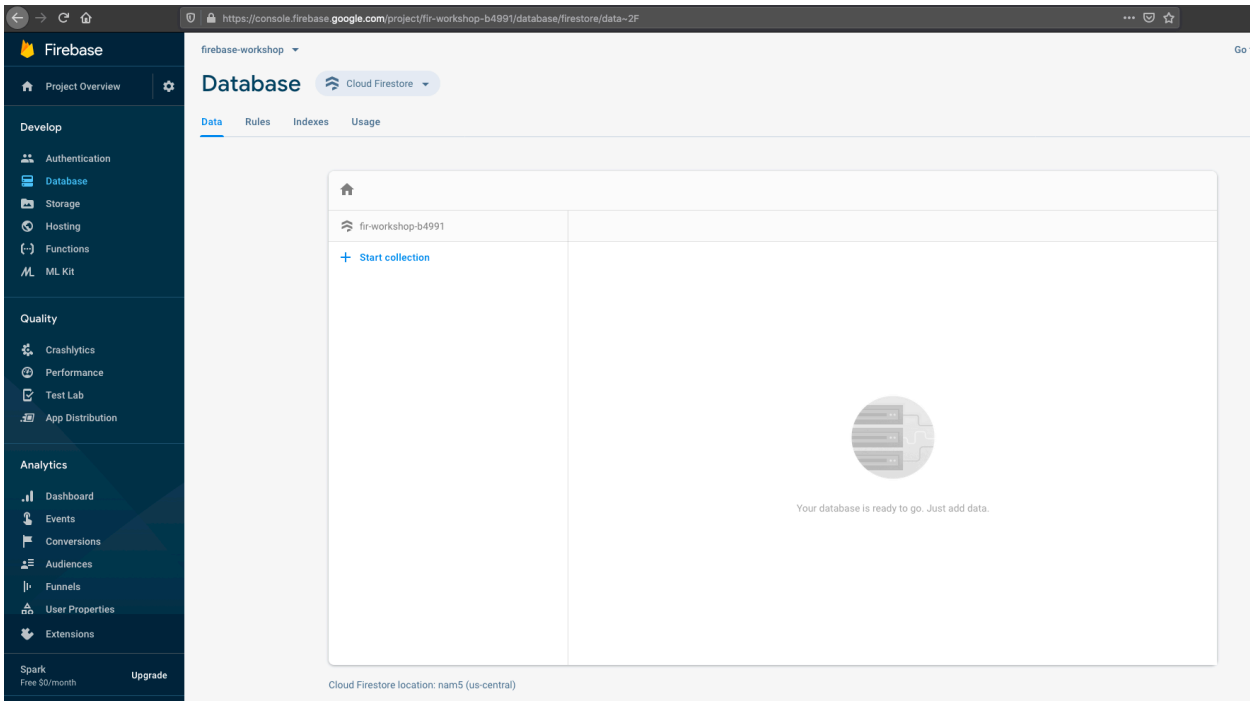
We will create the 2nd one, and start in **Production Mode**



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 lates 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

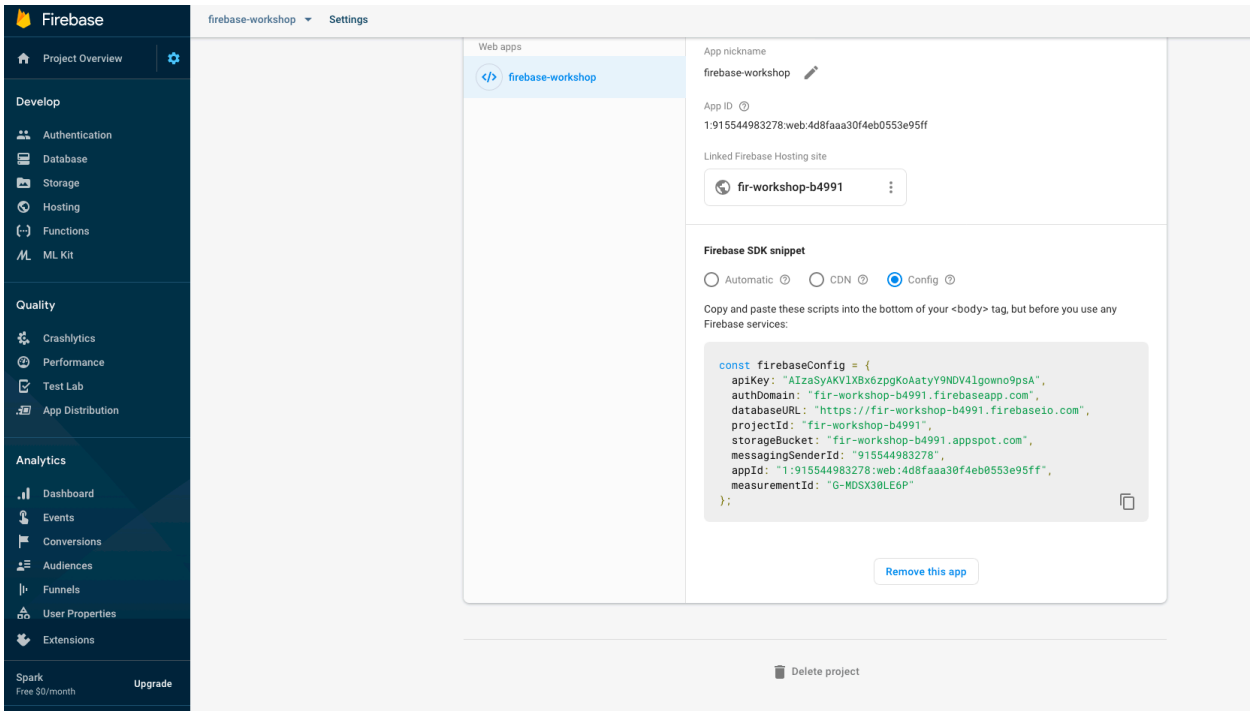
```
$ npm install && npm start
```

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



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,
```

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

export { AngularFirestore };
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, FirebaseModule],
...

})
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 **Firebase Module** created, we must import it also into the **login.module.ts** file

```
...
...

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

```
const modules = [

CommonModule,
LoginRoutingModule,
RouterModule,
HttpClientModule,
MaterialModule,
ReactiveFormsModule,
FirebaseModule,
];

...
...
export class LoginModule {}
```

Step 9: Last but no less, we must override the **login.service.ts** fo using **Firebase** 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 looging
user in'));
    }

    public async logOut(): Promise<void> {
        await this.afAuth.auth
        .signOut()
        .then(_ => console.log('user successfully
loged out'))
        .catch(_ => console.log('error while looging
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
```