Table of Contents

Requirements:

- Nodejs & npm (Node package manager)
- Angular CLI command line interface, which allows you to create the angular project and facilitates the generation of components.
- VS Code or any other IDE or Text Editor

Introduction:

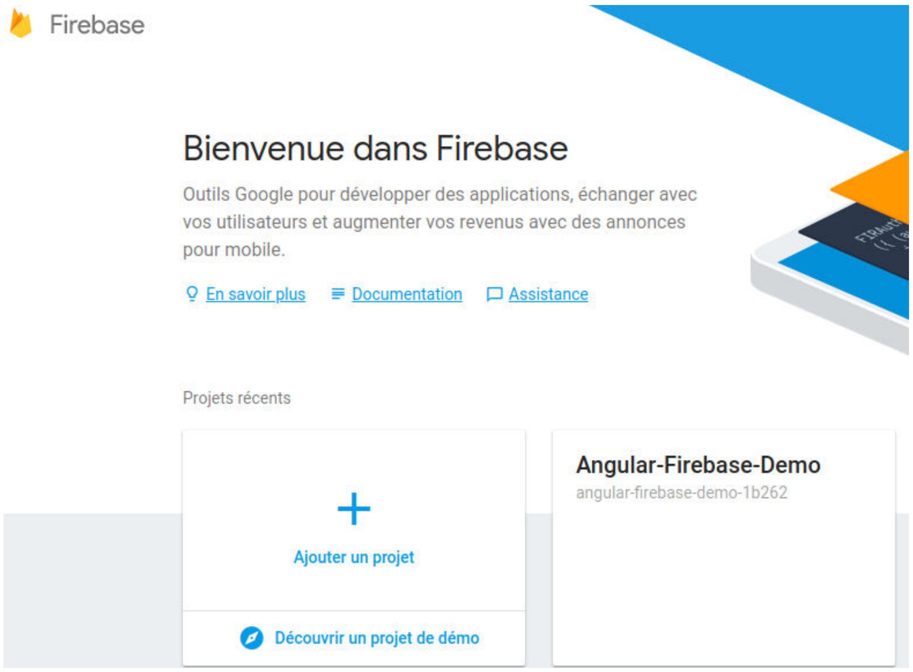
In this tutorial, we'll use Firebase as a Backend as a Service (BaaS) that provides us with a real-time database and an authentication service.

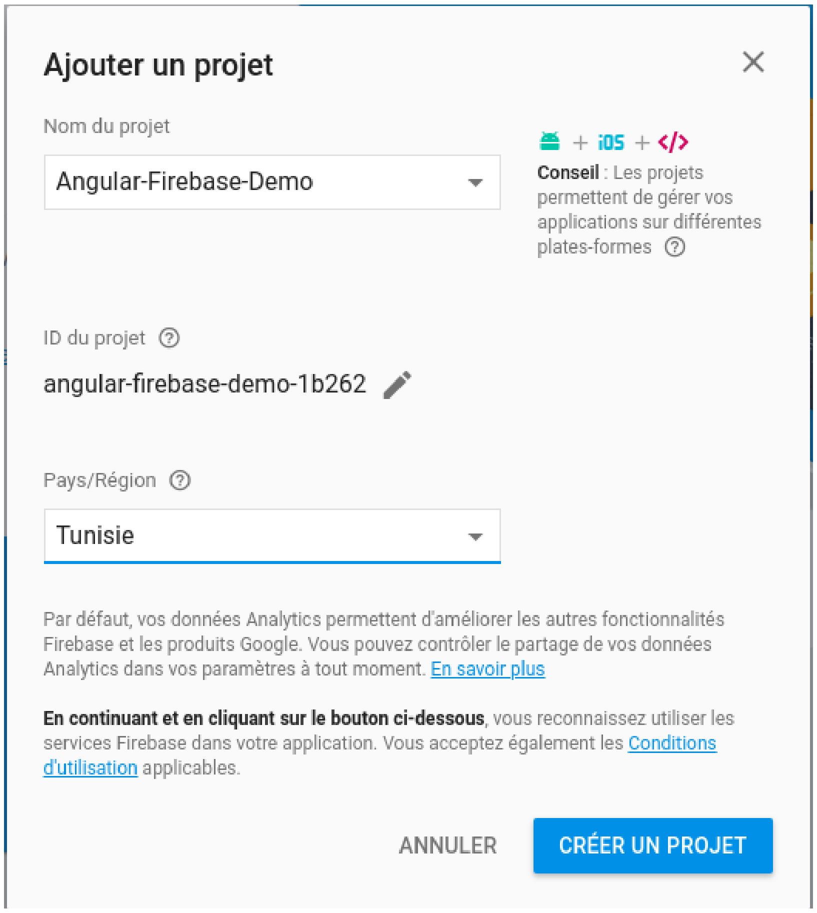
Then, we'll see how we can use Function as a Service (FaaS) to write server-side logic for our application.

I) Connecting Firebase to an Angular project:

During all of the following demos we will proceed as follows in order to create a Firebase project and bind it with an Angular 5 project

Step 1: Creation of a Firebase project (https://console.firebase.google.com):


Firebase
Bienvenue dans Firebase
Outils Google pour développer des applications, échanger avec vos utilisateurs et augmenter vos revenus avec des annonces pour mobile.
En savoir plus
Documentation
Assistance
Projets récents
Ajouter un projet
Découvrir un projet de démo
Angular-Firebase-Demo
angular-firebase-demo-1b262



Ajouter un projet

Nom du projet

Angular-Firebase-Demo

Conseil : Les projets permettent de gérer vos applications sur différentes plates-formes

ID du projet

angular-firebase-demo-1b262

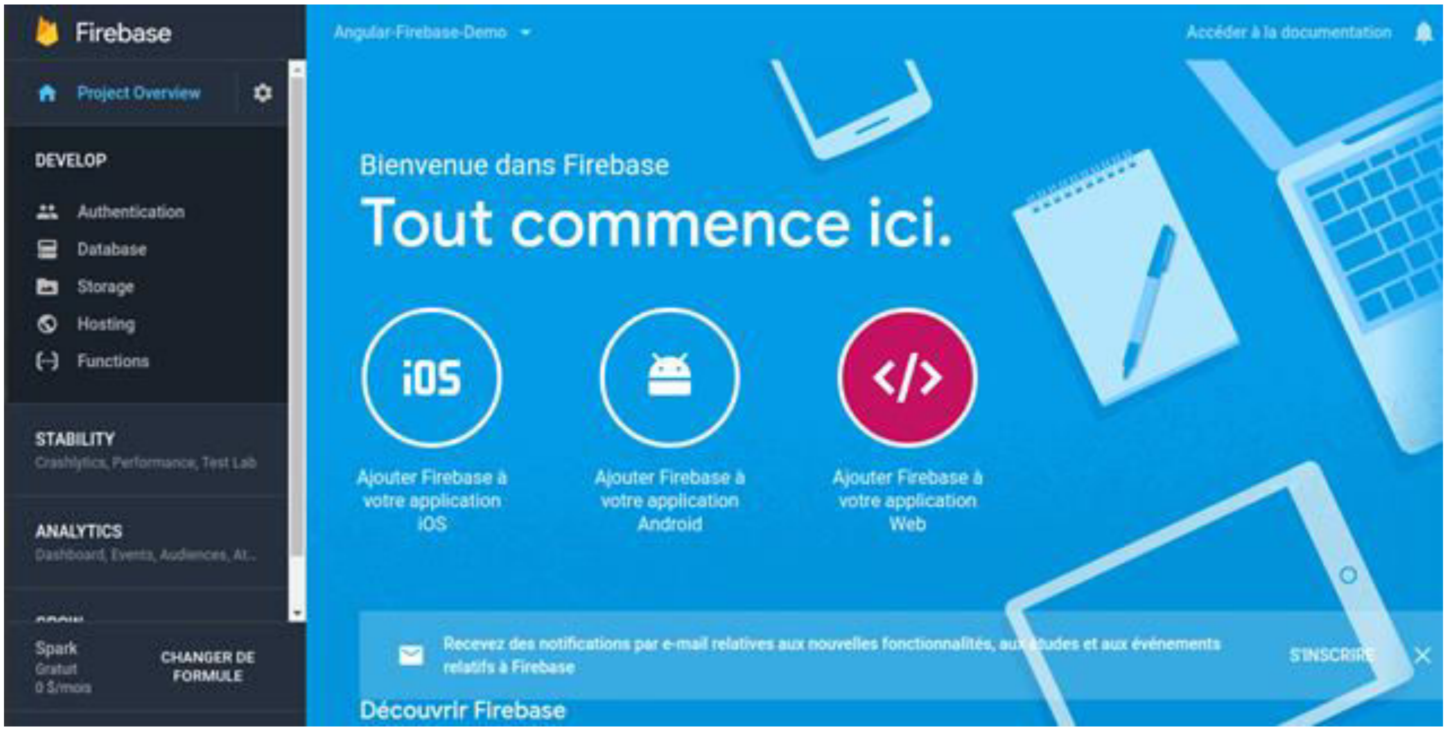
Pays/Région

Tunisie

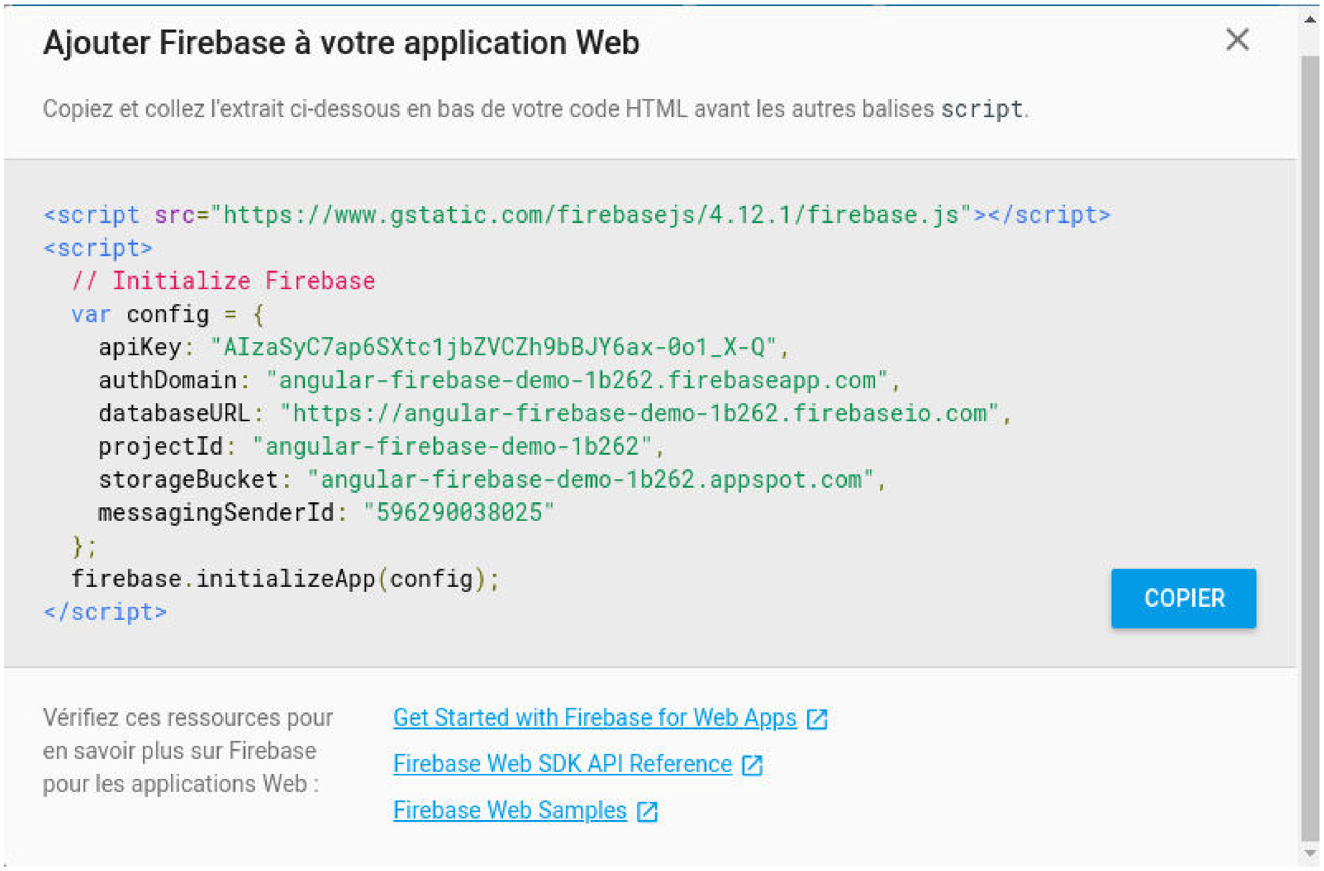
Par défaut, vos données Analytics permettent d'améliorer les autres fonctionnalités Firebase et les produits Google. Vous pouvez contrôler le partage de vos données Analytics dans vos paramètres à tout moment. En savoir plus

En continuant et en cliquant sur le bouton ci-dessous, vous reconnaissez utiliser les services Firebase dans votre application. Vous acceptez également les Conditions d'utilisation applicables.

Step 2: Getting the Firebase application credentials:



Firebase
Angular-Firebase-Demo
Accéder à la documentation
Project Overview
DEVELOP
Authentication
Database
Storage
Hosting
Functions
STABILITY
Crashlytics, Performance, Test Lab
ANALYTICS
Dashboard, Events, Audiences, At...
Spark
Gratuit
0 $/mois
CHANGER DE FORMULE
Bienvenue dans Firebase
Tout commence ici.
iOS
Ajouter Firebase à votre application iOS
Ajouter Firebase à votre application Android
Ajouter Firebase à votre application Web
Recevez des notifications par e-mail relatives aux nouvelles fonctionnalités, aux études et aux événements relatifs à Firebase
S'INSCRIRE
Découvrir Firebase



Copiez et collez l'extrait ci-dessous en bas de votre code HTML avant les autres balises `script`.

```
<script src="https://www.gstatic.com/firebasejs/4.12.1/firebase.js"></script>
<script>
  // Initialize Firebase
  var config = {
    apiKey: "AIzaSyC7ap6SXtc1jbZVCZh9bBJY6ax-0o1_X-Q",
    authDomain: "angular-firebase-demo-1b262.firebaseapp.com",
    databaseURL: "https://angular-firebase-demo-1b262.firebaseio.com",
    projectId: "angular-firebase-demo-1b262",
    storageBucket: "angular-firebase-demo-1b262.appspot.com",
    messagingSenderId: "596290038025"
  };
  firebase.initializeApp(config);
</script>
```

Vérifiez ces ressources pour en savoir plus sur Firebase pour les applications Web :

Get Started with Firebase for Web Apps
Firebase Web SDK API Reference
Firebase Web Samples

Step 3: Creation of a new Angular project:

```
ng new Angular-Firebase-Demo
```

Step 4: Adding required packages for firebase:

```
npm install firebase angularfire2 --save
```

Step 5: Connecting Angular project with the Firebase project:

- We start by creating a variable in environment.ts containing the firebase project's credentials:

```
export const environment = {
        production: false,

        firebase: {
                apiKey: "AIzaSyC7ap6SXtc1jbZVCZh9bBJY6ax-0o1_X-Q",
                authDomain: "angular-firebase-demo-1b262.firebaseapp.com",
                databaseURL: "https://angular-firebase-demo-1b262.firebaseio.com",
                projectId: "angular-firebase-demo-1b262",
                storageBucket: "angular-firebase-demo-1b262.appspot.com",
                messagingSenderId: "596290038025"
                }
        };
```

- Then, we import these modules to app.modules.ts :

```
import { AngularFireModule } from 'angularfire2';
import { AngularFirestoreModule } from 'angularfire2/firestore';
import { AngularFireAuthModule } from 'angularfire2/auth';
```

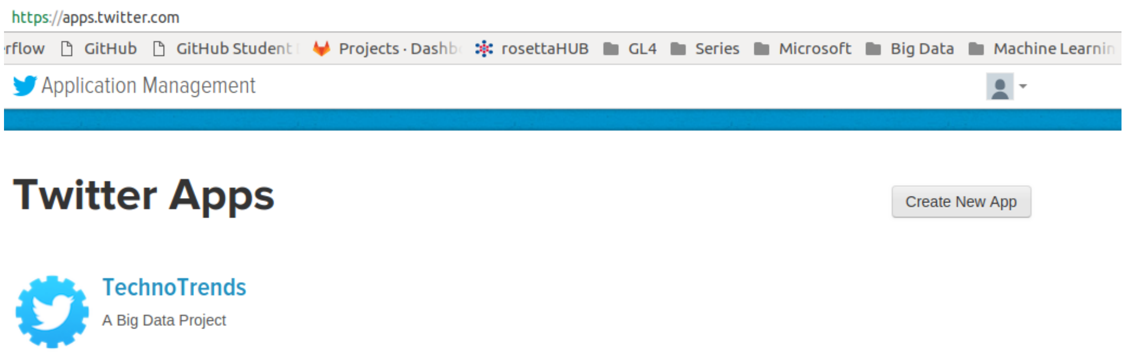
- And we add them to the imports array:

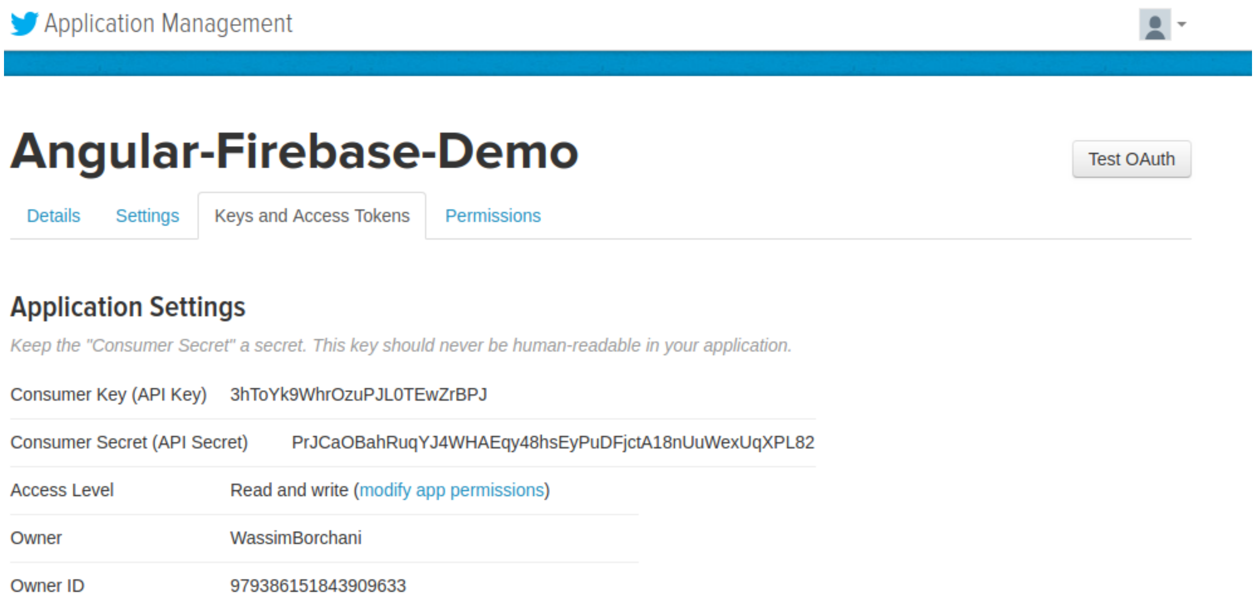
```
imports: [
        AngularFireModule.initializeApp(environment.firebase),
        AngularFirestoreModule, // imports firebase/firestore, only needed for database features
        AngularFireAuthModule, // imports firebase/auth, only needed for auth features
],
```

II) Authentication

i) Logging in with a Social Login Providers (Example with Twitter):

<u>**Step 1:** Creation of a Twitter app with apps.twitter.com in order to generate access tokens:</u>

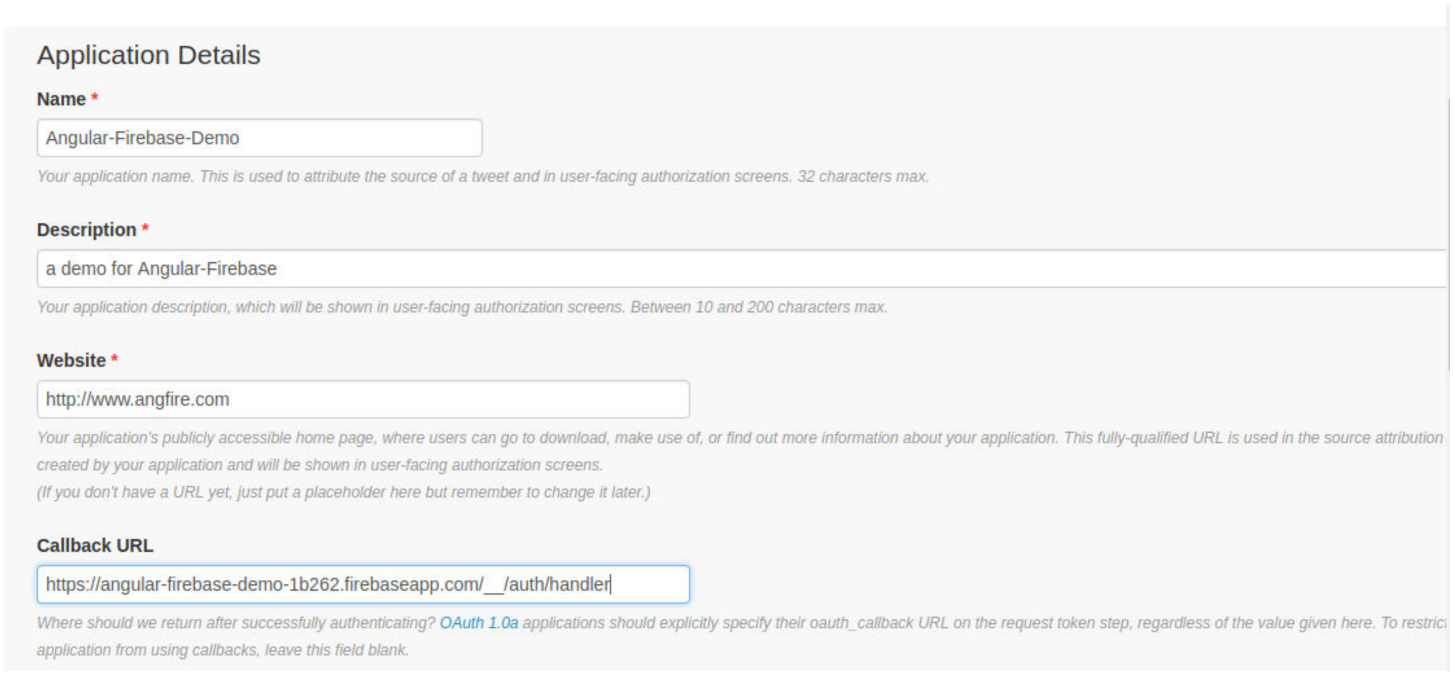




Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key)	3hToYk9WhrOzuPJL0TEwZrBPJ
Consumer Secret (API Secret)	PrJCaOBahRuqYJ4WHAEqy48hsEyPuDFjctA18nUuWexUqXPL82
Access Level	Read and write (modify app permissions)
Owner	WassimBorchani
Owner ID	979386151843909633



Application Details

Name *

Angular-Firebase-Demo

Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max.

Description *

a demo for Angular-Firebase

Your application description, which will be shown in user-facing authorization screens. Between 10 and 200 characters max.

Website *

http://www.angfire.com

Your application's publicly accessible home page, where users can go to download, make use of, or find out more information about your application. This fully-qualified URL is used in the source attribution created by your application and will be shown in user-facing authorization screens.
(If you don't have a URL yet, just put a placeholder here but remember to change it later.)

Callback URL

https://angular-firebase-demo-1b262.firebaseapp.com/__/auth/handler

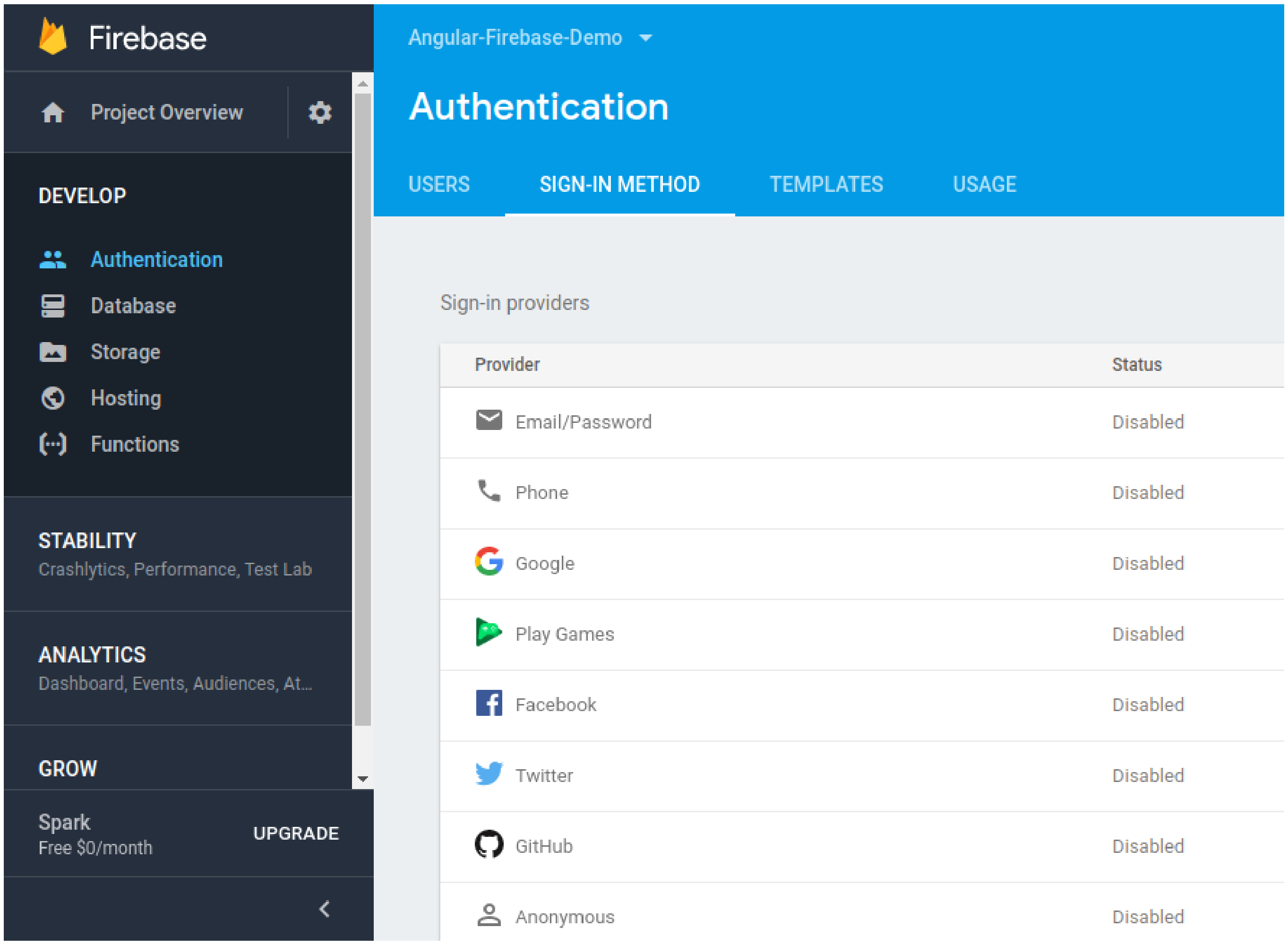
Where should we return after successfully authenticating? OAuth 1.0a applications should explicitly specify their oauth_callback URL on the request token step, regardless of the value given here. To restric application from using callbacks, leave this field blank.

<u>**Step 2:** Setting uo Firebase Social Login Providers (https://console.firebase.google.com):</u>

On this step we are going to integrate to our Angular app the social login providers of our choice. In this demo we will enable Twitter as the sole social sign-in option for our Angular 5 example app.

To select the authentication methods we want to integrate on our Angular app, we go to our Firebase project, under Firebase console, then we go to Develop => Authentication and then click the Sign-in method tab

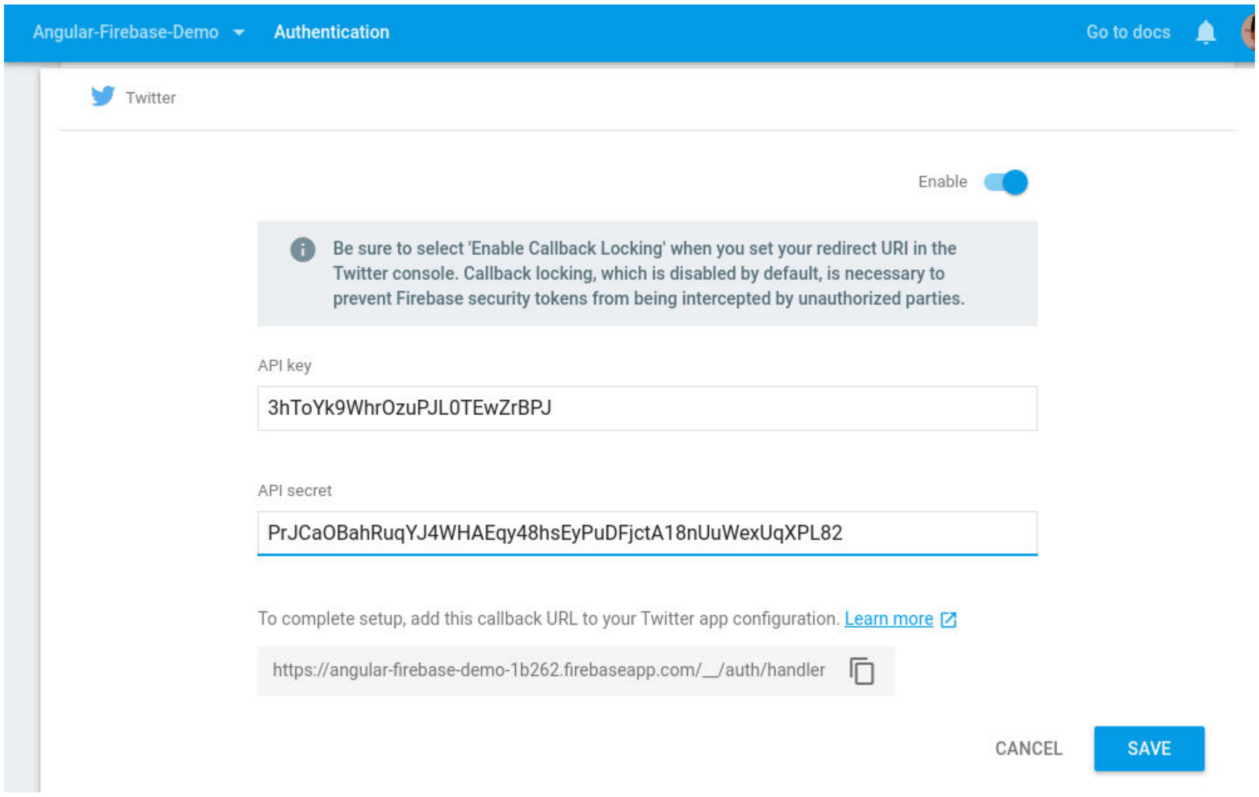
Firebase authentication providers such as Facebook and Twitter, require us to provide a Client API ID and a Client API Secret Key and also use the provided OAuth URI as the redirect URI from our Facebook or Twitter App.



Authentication

Sign-in providers

Provider	Status
Email/Password	Disabled
Phone	Disabled
Google	Disabled
Play Games	Disabled
Facebook	Disabled
Twitter	Disabled
GitHub	Disabled
Anonymous	Disabled



Be sure to select 'Enable Callback Locking' when you set your redirect URI in the Twitter console. Callback locking, which is disabled by default, is necessary to prevent Firebase security tokens from being intercepted by unauthorized parties.

API key
3hToYk9WhrOzuPJL0TEwZrBPJ

API secret
PrJCaOBahRuqYJ4WHAEqy48hsEyPuDFjctA18nUuWexUqXPL82

To complete setup, add this callback URL to your Twitter app configuration. Learn more

https://angular-firebase-demo-1b262.firebaseapp.com/__/auth/handler

Step 3: Creation of Angular services:

- **user.service.ts**

ng g s User

```
import { Injectable } from"@angular/core";
import'rxjs/add/operator/toPromise';
import { AngularFirestore } from'angularfire2/firestore';
import { AngularFireAuth } from'angularfire2/auth';
import*as firebase from'firebase/app';

@Injectable()
exportclass UserService {
        constructor(
                        public db: AngularFirestore,
                        public afAuth: AngularFireAuth
                        ){ }
        getCurrentUser(){
                return new Promise<any>((resolve, reject) => {
                        var user = firebase.auth().onAuthStateChanged(function(user){
                                if (user) {
                                        resolve(user);
                                }
                                else {
                                        reject('No user logged in');
                                }
                        })
                })
        }

        updateCurrentUser(value){
                return new Promise((resolve, reject) => {
                        var user = firebase.auth().currentUser;
                        user.updateProfile({
                                displayName: value.name,
                                photoURL: user.photoURL
                        }).then(res => {
                                resolve(res)
                        }, err => reject(err))
                })
        }
}
```

- **auth.service.ts:**

ng g s Auth

```
import { AngularFireAuthModule } from'angularfire2/auth';
import { Injectable } from'@angular/core';
import { AngularFireAuth } from'angularfire2/auth';
import * as firebase from'firebase/app';

@Injectable()
export class AuthService {

	constructor(private afAuth: AngularFireAuth) { }


	loginWithTwitter(){
		return new Promise<any>((resolve, reject) => {
			let provider = new firebase.auth.TwitterAuthProvider();
			this.afAuth.auth
			.signInWithPopup(provider)
			.then(res => {
				resolve(res);
			}, err => {
				console.log(err);
				reject(err);
			})
		})
	}
}
```

Step 4: Creation of some components:

- **LoginComponent -** This will feature our social logins and will also provide the possibility to login with email and password.

ng g c Login

Content of Login Component:

```
 import { Component } from'@angular/core';
import { AuthService } from'../services/auth.service'
import { Router, Params } from'@angular/router';
import { FormBuilder, FormGroup, Validators } from'@angular/forms';

@Component({
        selector: 'page-login',
        templateUrl: 'login.component.html',
        styleUrls: ['login.component.css']
        })

export class LoginComponent {

        loginForm: FormGroup;
        errorMessage: string = '';

        constructor( public authService: AuthService,
                         private router: Router,
                         private fb: FormBuilder
                        ) {
                this.createForm();
        }

createForm() {
        this.loginForm = this.fb.group({
                        email: ['', Validators.required ],
                        password: ['',Validators.required]
                });
        }
tryTwitterLogin(){
        this.authService.loginWithTwitter()
        .then(res => {
                this.router.navigate(['/user']);
        })
}

tryLogin(value){
        this.authService.doLogin(value)
        .then(res => {
                        this.router.navigate(['/user']);
        }, err => {
                console.log(err);
                this.errorMessage = err.message;
        })
        }
}
```

- **RegisterComponent -** This will feature our social logins and will also provide the possibility to create a new account with email and password.

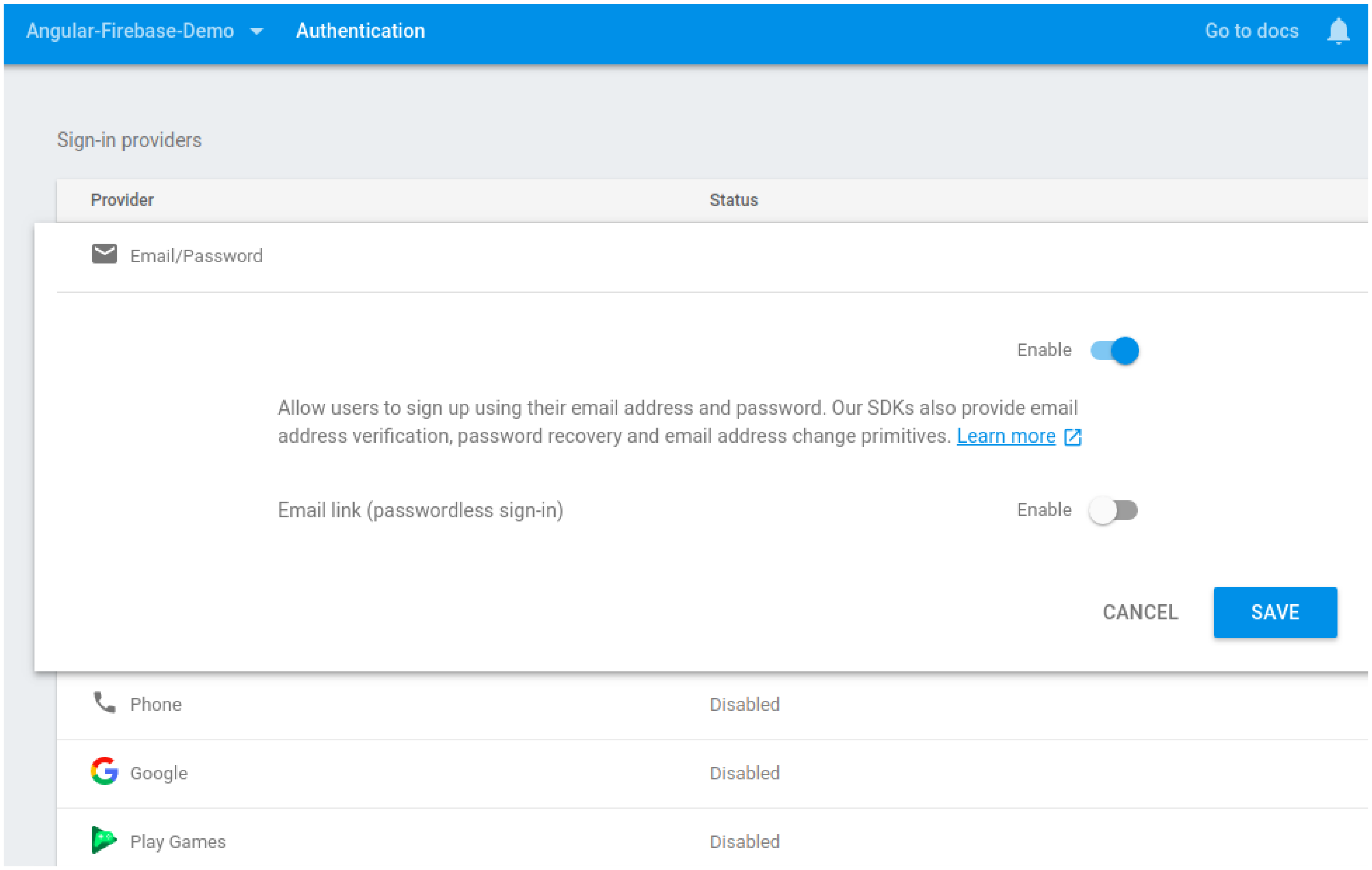
```
ng g c Register
```

- **UserComponent -** This will serve as the protected area that authenticated users will have access to.

```
ng g c User
```

ii) Login/Register with Email/Password:

Step 1: In the same way we did with social providers, we have to enable email/password sign-in method in the Firebase console.



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

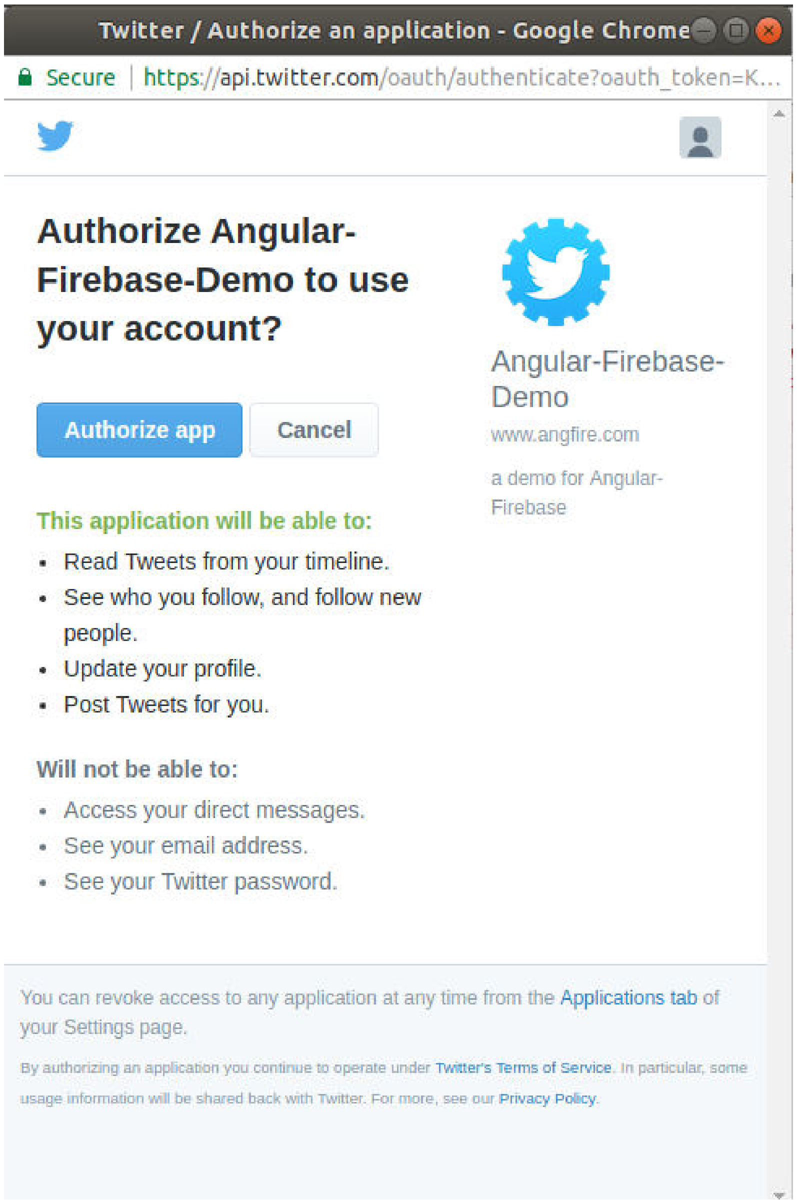
Provider	Status
Phone	Disabled
Google	Disabled
Play Games	Disabled

Step 2:

- We add the following code to our previously created auth.service:

```
doRegister(value){
        return new Promise<any>((resolve, reject) => {
                firebase.auth().createUserWithEmailAndPassword(value.email,
        value.password)
                .then(res => {
                        resolve(res);
                }, err => reject(err))
        })
}

doLogin(value){
        return new Promise<any>((resolve, reject) => {
                firebase.auth().signInWithEmailAndPassword(value.email, value.password)
                .then(res => {
                        resolve(res);
                }, err => reject(err))
        })
}

doLogout(){
        return new Promise((resolve, reject) => {
                if(firebase.auth().currentUser){
                        this.afAuth.a
                        uth.signOut()
                        resolve();
                }
                else{
                        reject();
                }
        });
}
```

Preview:


Twitter / Authorize an application - Google Chrome
Secure
https://api.twitter.com/oauth/authenticate?oauth_token=K...

Authorize Angular-Firebase-Demo to use your account?

Authorize app | Cancel

Angular-Firebase-Demo
www.angfire.com
a demo for Angular-Firebase

This application will be able to:

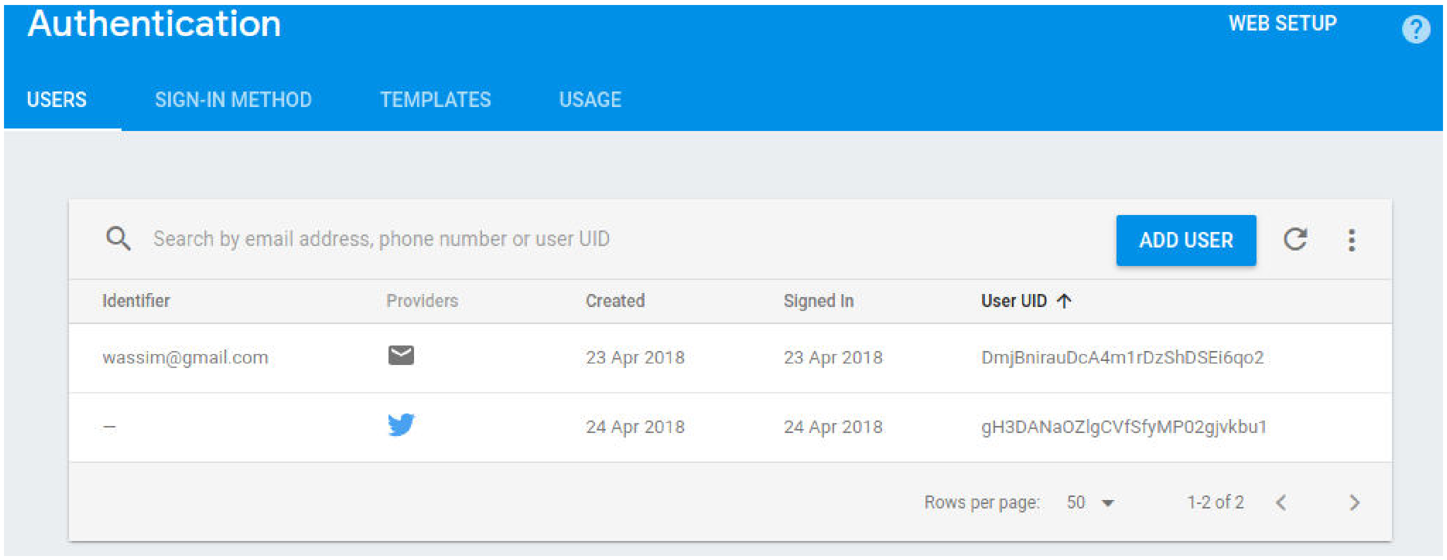
- Read Tweets from your timeline.
- See who you follow, and follow new people.
- Update your profile.
- Post Tweets for you.

Will not be able to:

- Access your direct messages.
- See your email address.
- See your Twitter password.

You can revoke access to any application at any time from the Applications tab of your Settings page.

By authorizing an application you continue to operate under Twitter's Terms of Service. In particular, some usage information will be shared back with Twitter. For more, see our Privacy Policy.



Identifier	Providers	Created	Signed In	User UID ↑
wassim@gmail.com	✉	23 Apr 2018	23 Apr 2018	DmjBnirauDcA4m1rDzShDSEi6qo2
–	🐦	24 Apr 2018	24 Apr 2018	gH3DANaOZlgCVfSfyMP02gjvkbu1

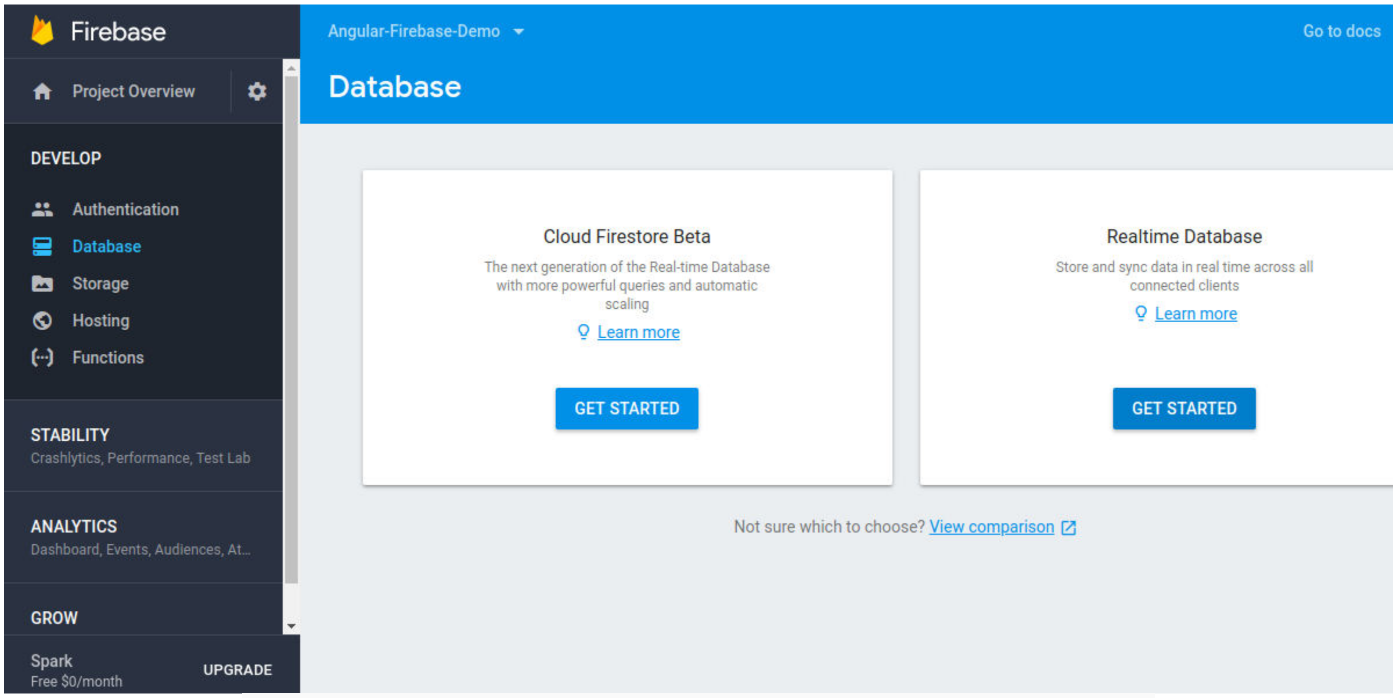
Rows per page: 50 1-2 of 2

III) Databases

Step 1: Changing FirebaseDB Permissions (https://console.firebase.google.com):

- Go to Database > GET STARTED > Rules Tab
- Set both read & write permission to true

Note: this is not recommended for a real world application but we've done this just for the sake of this tutorial to make things easier



Cloud Firestore Beta

The next generation of the Real-time Database with more powerful queries and automatic scaling

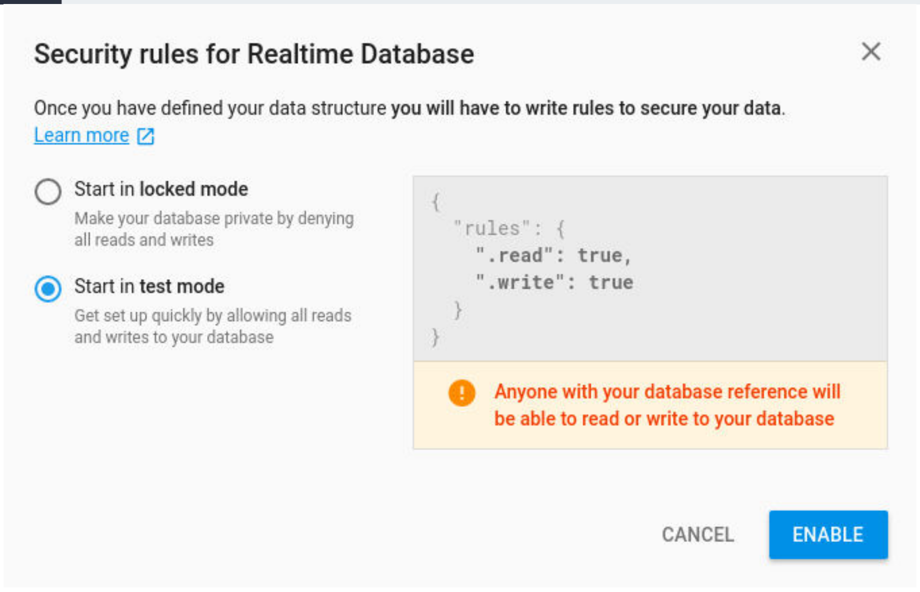
Learn more

Realtime Database

Store and sync data in real time across all connected clients

Learn more

Not sure which to choose? View comparison



Security rules for Realtime Database

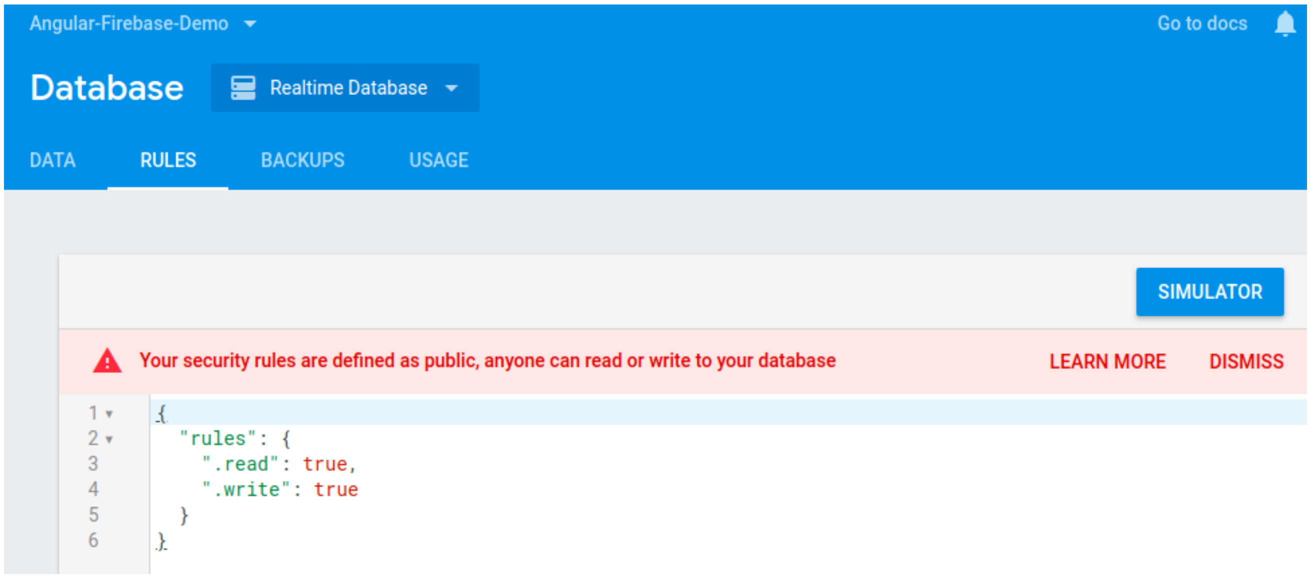
Once you have defined your data structure **you will have to write rules to secure your data.**
Learn more

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": true,
    ".write": true
  }
}
```

Anyone with your database reference will be able to read or write to your database



Your security rules are defined as public, anyone can read or write to your database

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

Step 2: Creation of Employee Model in Angular:

ng g class employee –type=model

Content of Employee Model:

```
export class Employee {
        $key: string;
        name: string;
        position: string;
        office: string;
        salary: number;
}
```

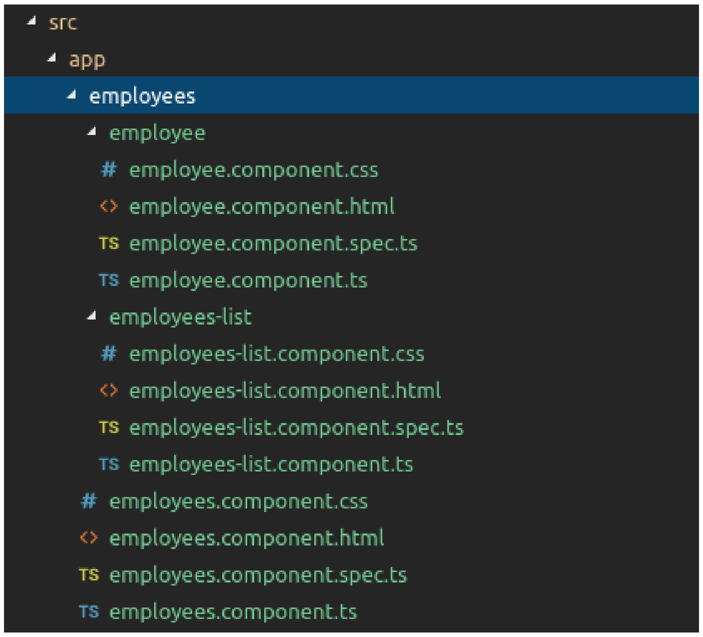
Note: $key is used to store unique key automatically generated by firebase DB when we insert a new record.

Step 3: Implementation of Employee Services:

Inside the service file we will implement Firebase CRUD Operations:

```
import { Injectable } from'@angular/core';
import { AngularFireDatabase, AngularFireList } from'angularfire2/database'
import { Employee} from'../models/employee.model';

@Injectable()
export class EmployeeService {
	employeeList: AngularFireList<any>;
	selectedEmployee: Employee = new Employee();
	constructor(private firebase :AngularFireDatabase ) { }

	getData(){
		this.employeeList = this.firebase.list('employees');
		returnthis.employeeList;
	}

	insertEmployee(employee : Employee)
	{
		this.employeeList.push({
			name: employee.name,
			position: employee.position,
			office: employee.office,
			salary: employee.salary
		});
	}

	updateEmployee(employee : Employee){
		this.employeeList.update(employee.$key,
			{
				name: employee.name,
				position: employee.position,
				office: employee.office,
				salary: employee.salary
			});
	}

	deleteEmployee($key : string){
		this.employeeList.remove($key);
	}
}
```

Step 4: Creation of Angular Components with the following hierarchies:



```
src
  app
    employees
      employee
        employee.component.css
        employee.component.html
        employee.component.spec.ts
        employee.component.ts
      employees-list
        employees-list.component.css
        employees-list.component.html
        employees-list.component.spec.ts
        employees-list.component.ts
      employees.component.css
      employees.component.html
      employees.component.spec.ts
      employees.component.ts
```

- **Content of EmployeesListComponent:**

```
import { Component, OnInit } from'@angular/core';
import { EmployeeService } from'../../services/employee.service';
import { Employee } from'../../models/employee.model';

@Component({
        selector: 'app-employees-list',
        templateUrl: './employees-list.component.html',
        styleUrls: ['./employees-list.component.css']
})

export class EmployeesListComponent implements OnInit {
        employeesList: Employee[];
        constructor(private employeeService: EmployeeService) { }

        ngOnInit() {
                var x = this.employeeService.getData();
                x.snapshotChanges().subscribe(item => {
                        this.employeesList = [];
                        item.forEach(element => {
                                var y = element.payload.toJSON();
                                y["$key"] = element.key;
                                this.employeesList.push(y as Employee);
                        });
```

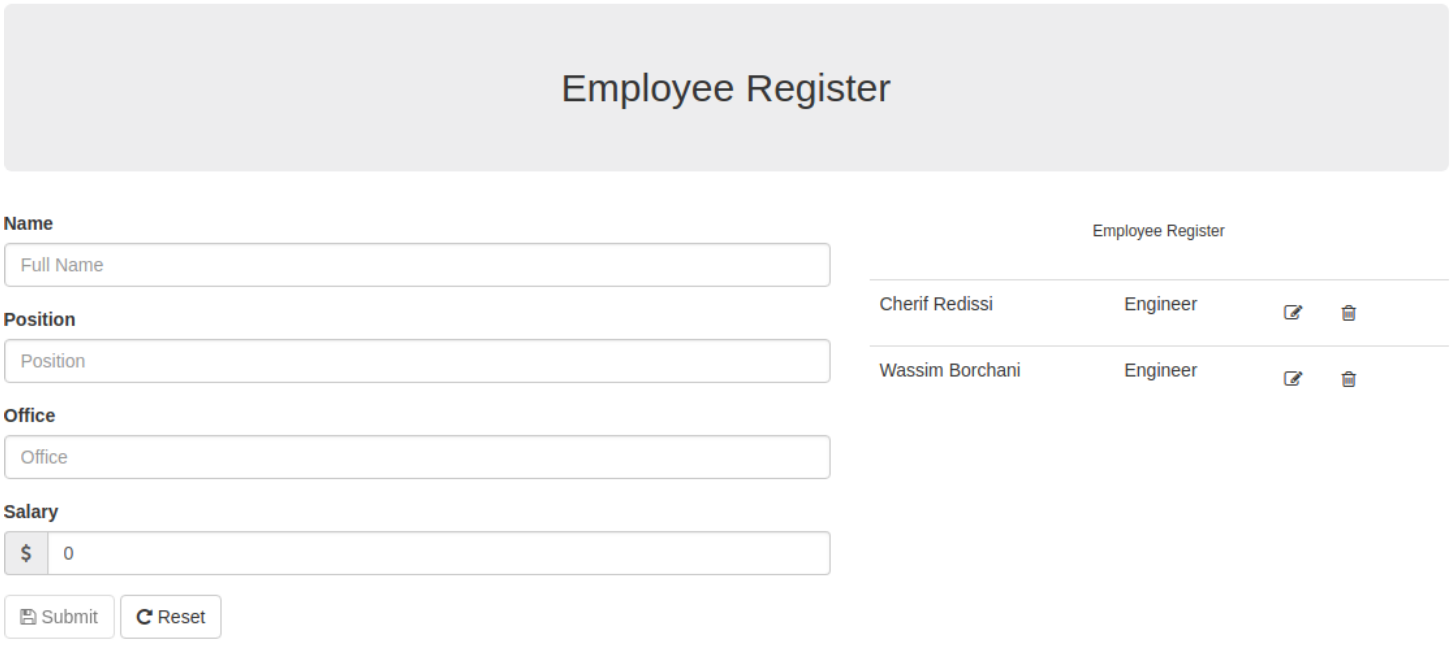
```
			});
		}

		onEdit(emp: Employee) {
			this.employeeService.selectedEmployee = Object.assign({}, emp);
		}

		onDelete(key: string) {
			if (confirm('Are you sure to delete this record ?') == true) {
				this.employeeService.deleteEmployee(key);
				console.log("Deleted Successfully", "Employee register");
			}
		}

}
```

- **Content of EmployeeComponent**

```
import { Component, OnInit } from'@angular/core';
import { NgForm } from'@angular/forms'
import { EmployeeService } from'../../services/employee.service';

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

	constructor(private employeeService: EmployeeService) { }

	ngOnInit() {
		this.resetForm();
	}

	onSubmit(employeeForm: NgForm) {
		if (employeeForm.value.$key == null)
			this.employeeService.insertEmployee(employeeForm.value);
		else
			this.employeeService.updateEmployee(employeeForm.value);
		this.resetForm(employeeForm);
```

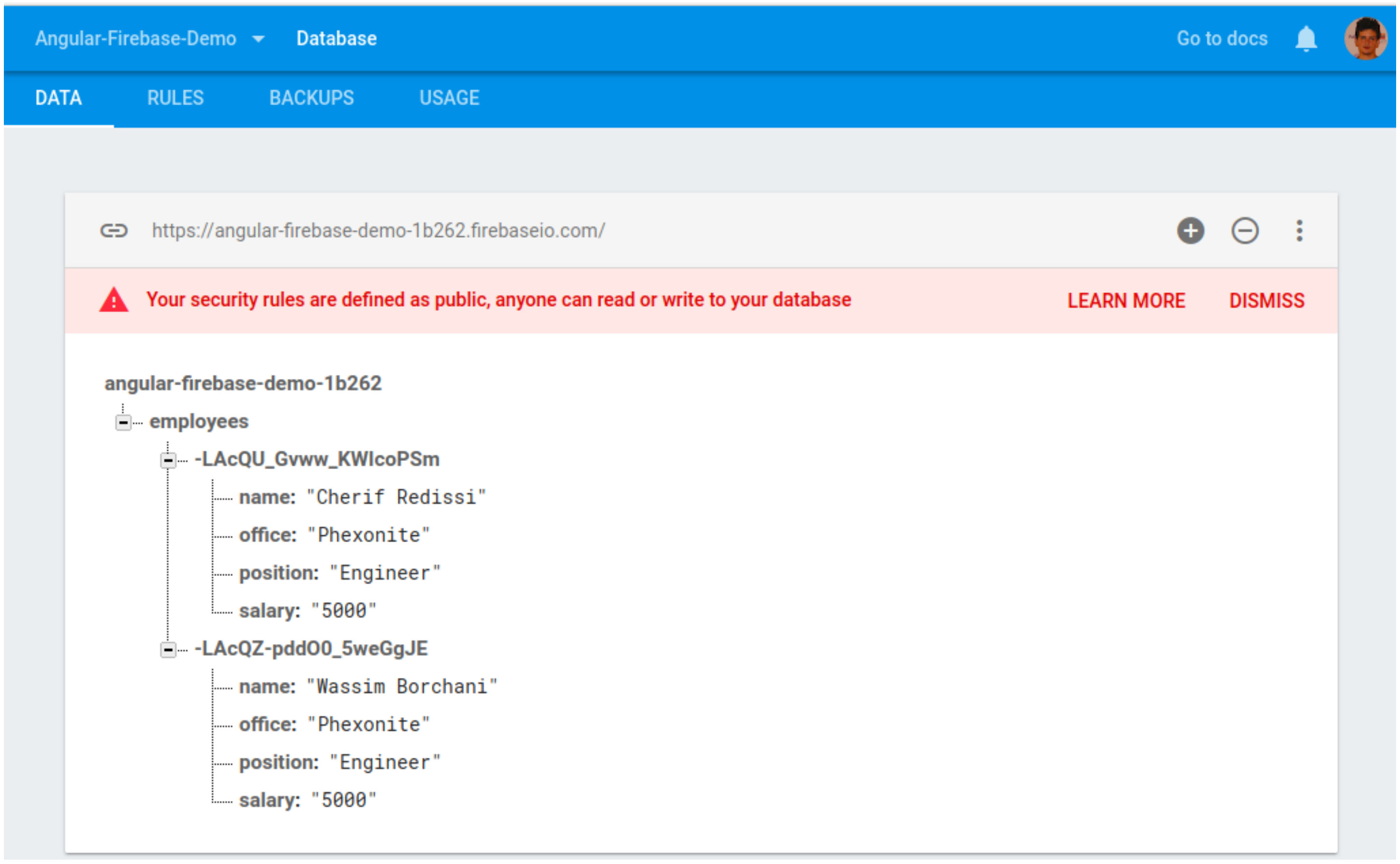
```
            console.log('Submitted Succcessfully', 'Employee Register');
        }

        resetForm(employeeForm?: NgForm) {
            if (employeeForm != null)
                employeeForm.reset();
            this.employeeService.selectedEmployee = {
                $key: null,
                name: '',
                position: '',
                office: '',
                salary: 0,
            }
        }
}
```

Preview:



Employee Register

Cherif Redissi	Engineer	
Wassim Borchani	Engineer	



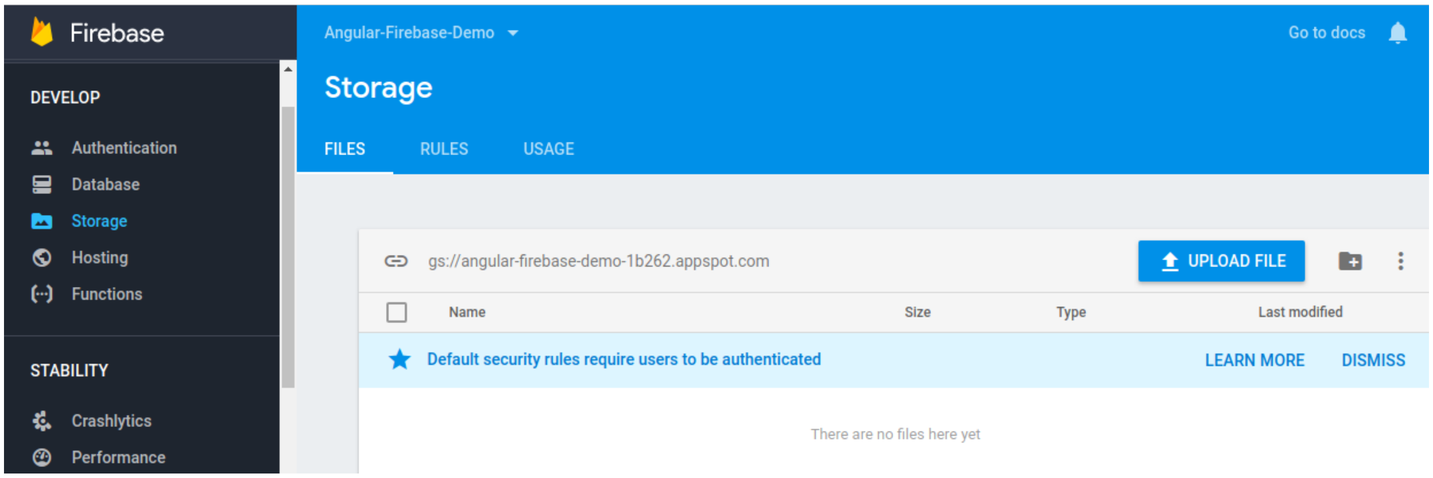
Your security rules are defined as public, anyone can read or write to your database

```
angular-firebase-demo-1b262
  employees
    -LAcQU_Gvww_KWIcoPSm
      name: "Cherif Redissi"
      office: "Phexonite"
      position: "Engineer"
      salary: "5000"
    -LAcQZ-pddO0_5weGgJE
      name: "Wassim Borchani"
      office: "Phexonite"
      position: "Engineer"
      salary: "5000"
```

IV) Storage

Step 1: Changing Storage Rules (https://console.firebase.google.com):

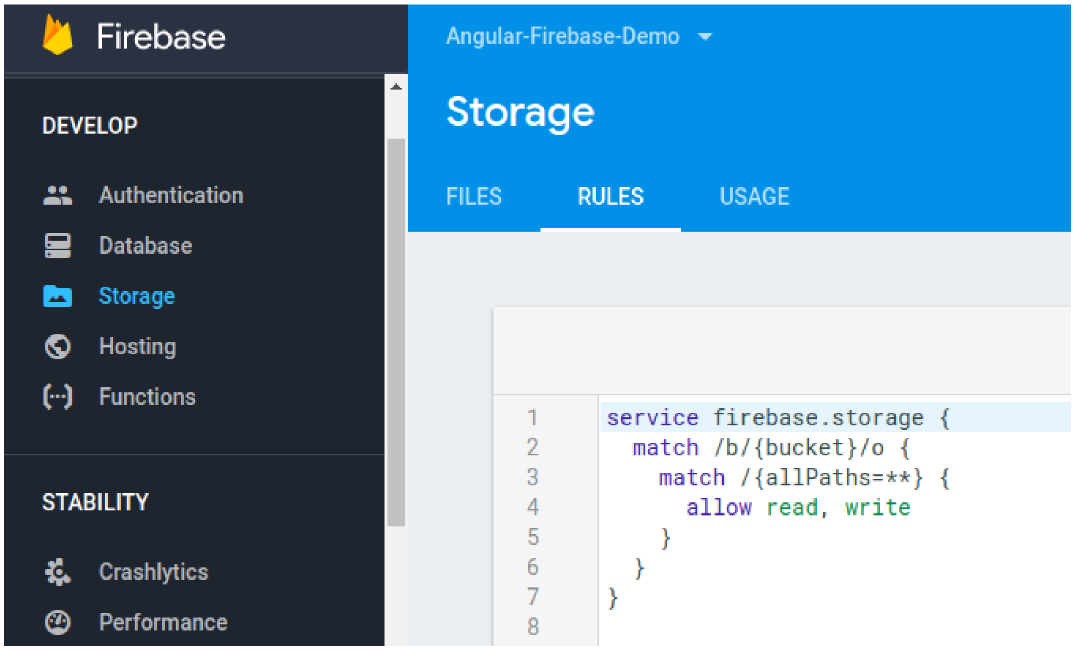
This step is optional and it was made just for the sake of this demo to make things quiet easier



gs://angular-firebase-demo-1b262.appspot.com

Default security rules require users to be authenticated

There are no files here yet



```
service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      allow read, write
    }
  }
}
```

Step 2: Creation of Angular components and services:

```
ng g c upload-form
ng g c upload-detail
ng g c uploads-list

ng g s upload
```

Step 3: Defining our Upload class:

```
ng g class upload
```

```
export class Upload {

	$key: string;
	file:File;
	name:string;
	url:string;
	progress:number;
	createdAt: Date = new Date();
	constructor(file:File) {
		this.file = file;
	}
}
```

This class will be used in the service layer.
Notice it has a constructor for file attribute, which has a type of File. This will allows us to initialize new uploads.

<u>Step 4: Implementing Upload Service:</u>

This Service accomplishes the following tasks:

- Establish a reference to the firebase storage bucket.
- Define the uploadTask as a promise to put the file in storage.
- Monitor the uploadTask event using the .on function.
- Handle the events of in progress, success, and error.

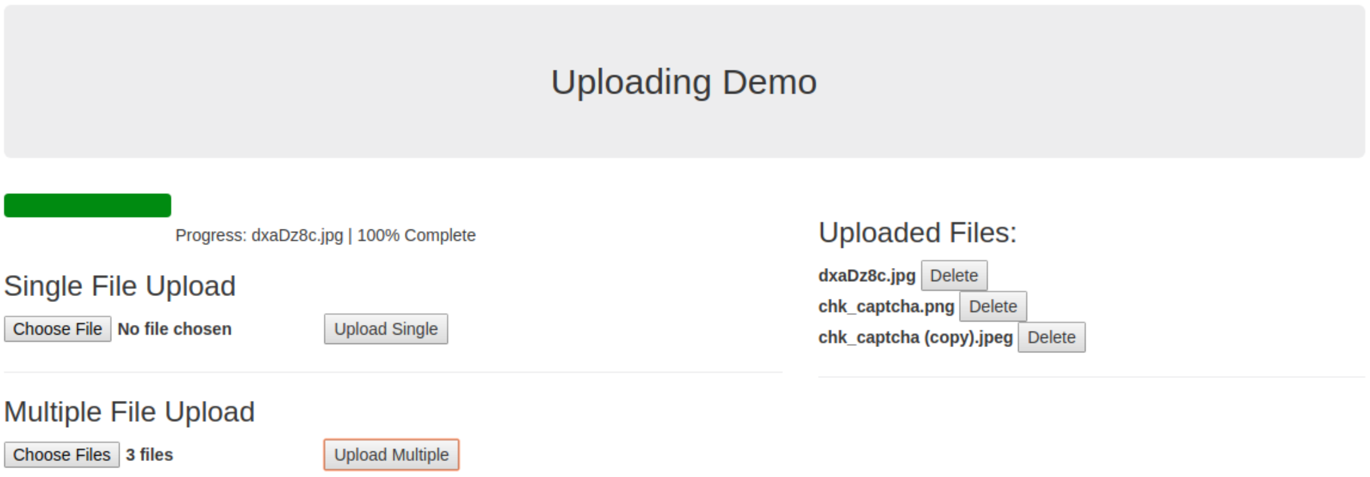
We can reuse this upload process for both single and multiple file uploads from the component.

```
import { Injectable } from'@angular/core';
import { AngularFireDatabase, AngularFireList } from'angularfire2/database';
import*as firebase from'firebase';
import { Observable } from'rxjs/Observable';
import { Upload } from'../utils/upload';

@Injectable()
exportclass UploadService {

        basePath = 'uploads';
        uploadsRef: AngularFireList<Upload>;
        uploads: Observable<Upload[]>;

        constructor(private db: AngularFireDatabase) { }

        getUploads() {
                this.uploads = this.db.list(this.basePath).snapshotChanges().map((actions) => {
                        return actions.map((a) => {
                                const data = a.payload.val();
                                const $key = a.payload.key;
                                return { $key, ...data };
                        });
                });
        return this.uploads;
        }

        deleteUpload(upload: Upload) {
                this.deleteFileData(upload.$key)
```

```
          .then( () => {
                  this.deleteFileStorage(upload.name);
          })
          .catch((error) => console.log(error));
  }

  // Executes the file uploading to firebase
  //https://firebase.google.com/docs/storage/web/upload-files
  pushUpload(upload: Upload) {
          const storageRef = firebase.storage().ref();
          const uploadTask = storageRef.child(`${this.basePath}/${upload.file.name}`).put(upload.file);

                  uploadTask.on(firebase.storage.TaskEvent.STATE_CHANGED,
                  (snapshot: firebase.storage.UploadTaskSnapshot) => {
                  // upload in progress
                          const snap = snapshot;
                          upload.progress = (snap.bytesTransferred / snap.totalBytes) * 100
                  }, (error) => {
                  // upload failed
                          console.log(error);
                  },
                  () => {
                  // upload success
                  if (uploadTask.snapshot.downloadURL) {
                          upload.url = uploadTask.snapshot.downloadURL;
                          upload.name = upload.file.name;
                          this.saveFileData(upload);
                          return;
                  } else {
                          console.error('No download URL!');
                  }
          },
  );
}

  // Writes the file details to the realtime db
  private saveFileData(upload: Upload) {
          this.db.list(`${this.basePath}/`).push(upload);
  }

  // Writes the file details to the realtime db
  private deleteFileData(key: string) {
          returnthis.db.list(`${this.basePath}/`).remove(key);
```

```
        }

        // Firebase files must have unique names in their respective storage dir
        // So the name serves as a unique key
        private deleteFileStorage(name: string) {
                const storageRef = firebase.storage().ref();
                storageRef.child(`${this.basePath}/${name}`).delete()
        }
}
```

Preview:



Uploading Demo

Progress: dxaDz8c.jpg | 100% Complete

Single File Upload

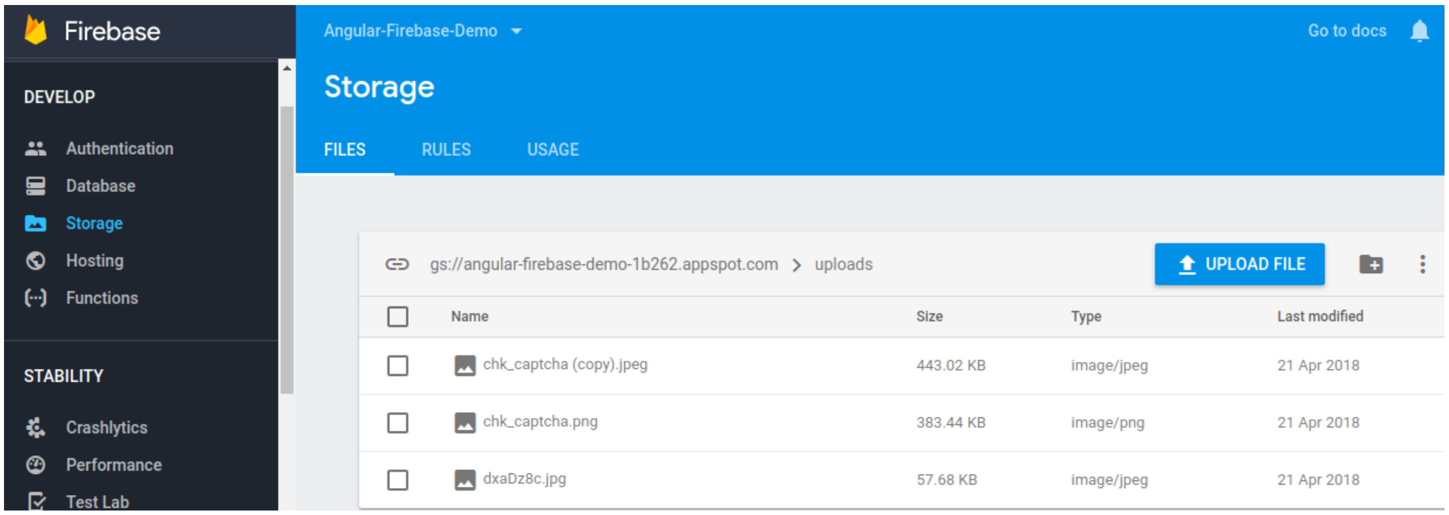
Choose File No file chosen Upload Single

Multiple File Upload

Choose Files 3 files Upload Multiple

Uploaded Files:

dxaDz8c.jpg Delete
chk_captcha.png Delete
chk_captcha (copy).jpeg Delete



gs://angular-firebase-demo-1b262.appspot.com > uploads

Name	Size	Type	Last modified
chk_captcha (copy).jpeg	443.02 KB	image/jpeg	21 Apr 2018
chk_captcha.png	383.44 KB	image/png	21 Apr 2018
dxaDz8c.jpg	57.68 KB	image/jpeg	21 Apr 2018

Demo 4: Firebase Functions

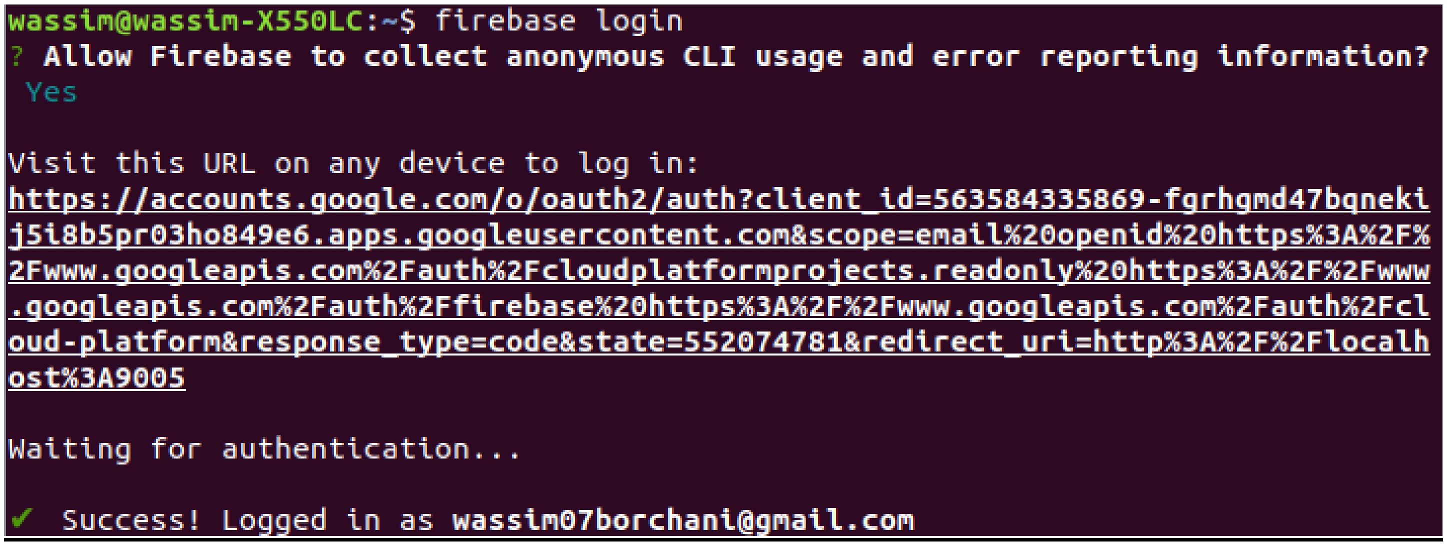
In this part, we'll see how we can use Function as a Service (FaaS) to write server-side logic for our application.

Step 1: In order to get started with firebase functions we need to install the firebase tools:

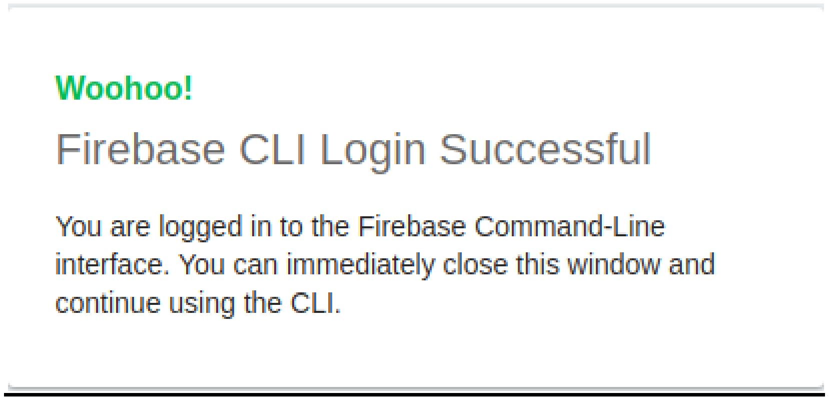
```
npm install-g firebase-tools
```

Step 2: authenticate to our Google account:

```
firebase login
```



```
wassim@wassim-X550LC:~$ firebase login
? Allow Firebase to collect anonymous CLI usage and error reporting information?
 Yes

Visit this URL on any device to log in:
https://accounts.google.com/o/oauth2/auth?client_id=563584335869-fgrhgmd47bqneki
j5i8b5pr03ho849e6.apps.googleusercontent.com&scope=email%20openid%20https%3A%2F%
2Fwww.googleapis.com%2Fauth%2Fcloudplatformprojects.readonly%20https%3A%2F%2Fwww
.googleapis.com%2Fauth%2Ffirebase%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcl
oud-platform&response_type=code&state=552074781&redirect_uri=http%3A%2F%2Flocalh
ost%3A9005

Waiting for authentication...

✔  Success! Logged in as wassim07borchani@gmail.com
```



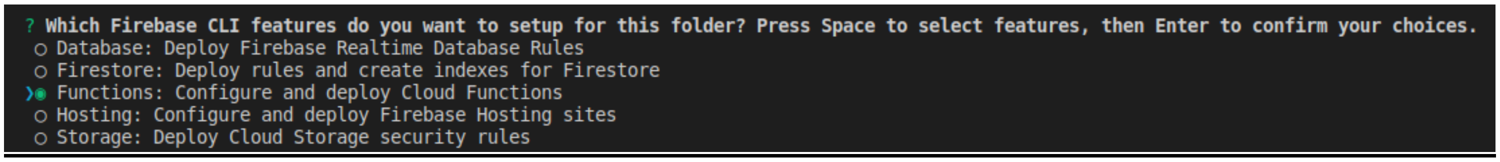
Woohoo!

Firebase CLI Login Successful

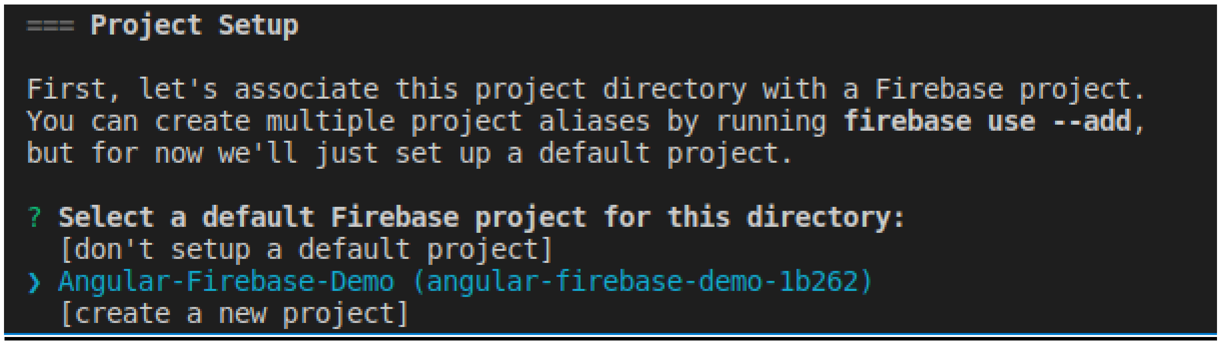
You are logged in to the Firebase Command-Line interface. You can immediately close this window and continue using the CLI.

Step 3: enable functions for our project:

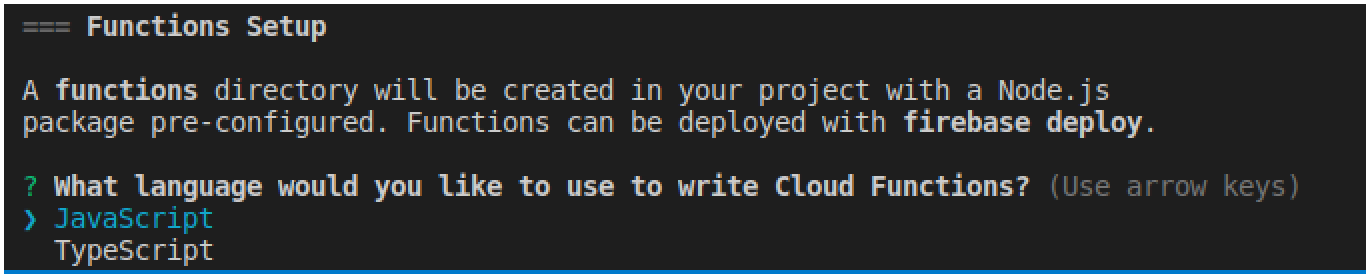
firebase init functions



```
? Which Firebase CLI features do you want to setup for this folder? Press Space to select features, then Enter to confirm your choices.
 ○ Database: Deploy Firebase Realtime Database Rules
 ○ Firestore: Deploy rules and create indexes for Firestore
❯◉ Functions: Configure and deploy Cloud Functions
 ○ Hosting: Configure and deploy Firebase Hosting sites
 ○ Storage: Deploy Cloud Storage security rules
```



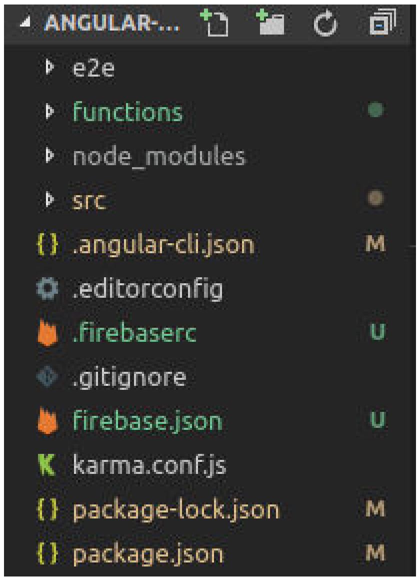
```
=== Project Setup

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.

? Select a default Firebase project for this directory:
  [don't setup a default project]
❯ Angular-Firebase-Demo (angular-firebase-demo-1b262)
  [create a new project]
```

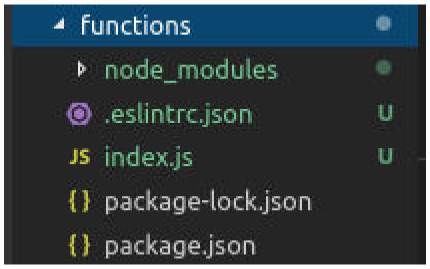


```
=== Functions Setup

A functions directory will be created in your project with a Node.js
package pre-configured. Functions can be deployed with firebase deploy.

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

Once we follow thru the prompts, it'll create a new folder named "functions" in your project root

By having a look at functions folder we'll find an index.js and a package.json file. The package.json has 2 dependencies "firebase-admin" & "firebase-functions":





- **package.json's content:**

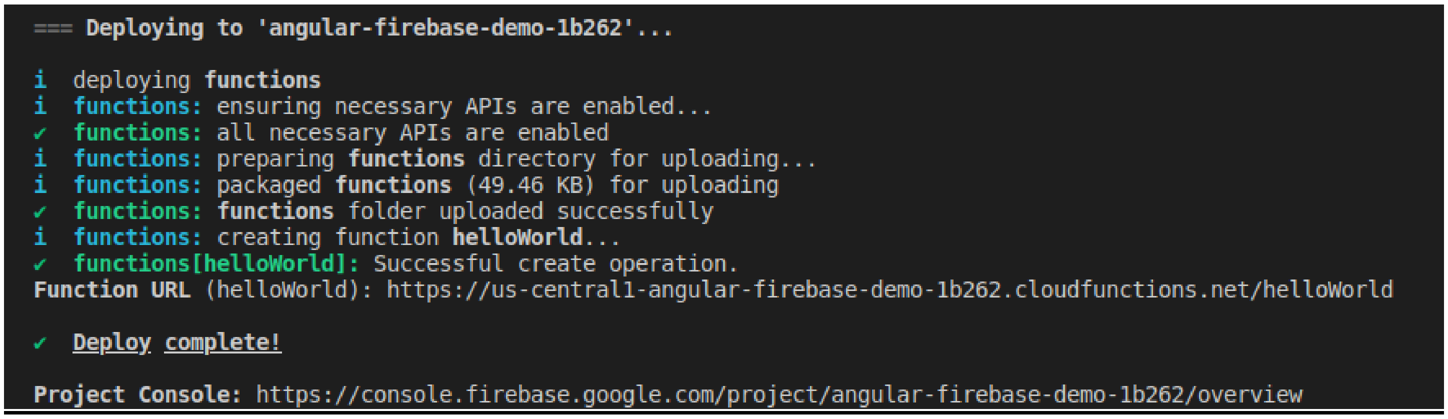
```
{
        "name": "functions",
        "description": "Cloud Functions for Firebase",

        "dependencies": {
                "firebase-admin": "~5.12.0",
                "firebase-functions": "^1.0.1",
                "express":"4.15.3",
                "cookie-parser":"1.4.3",
                "cors":"2.8.3"
        },
        "private": true,
        "devDependencies": {
                "@types/node": "^7.0.21"
        }
}
```

Step 4: uncomment index.js content:

```
const functions = require('firebase-functions');

// Create and Deploy Your First Cloud Functions
// https://firebase.google.com/docs/functions/write-firebase-functions

exports.helloWorld = functions.https.onRequest((request, response) => {
        response.send("Hello from Firebase!");
});
```

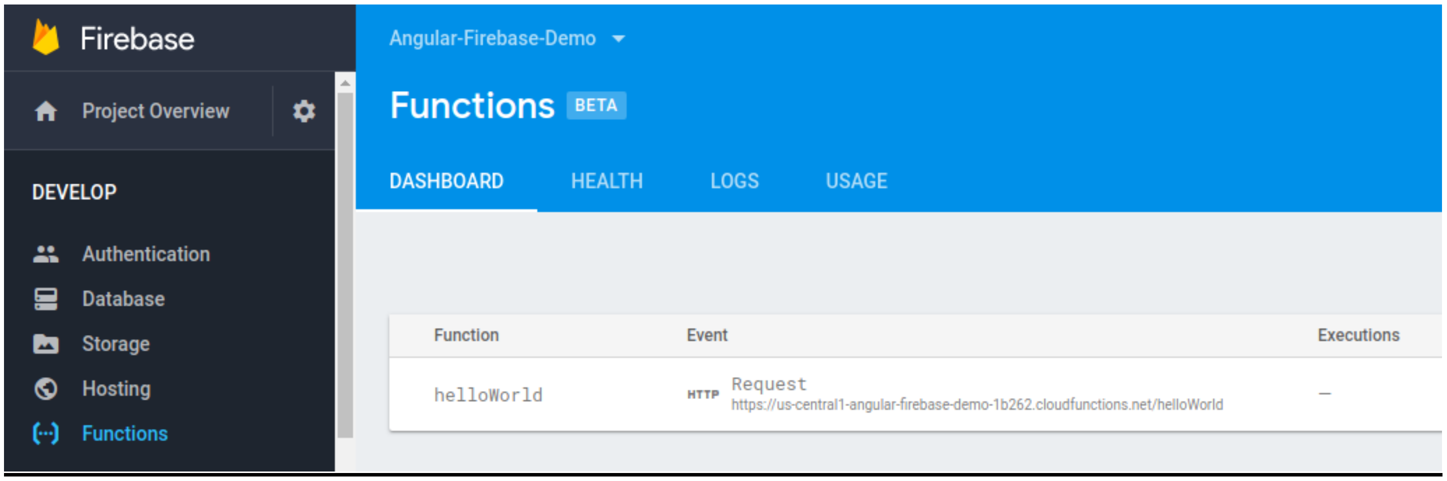
Step 5: deploy functions:

firebase deploy –only functions

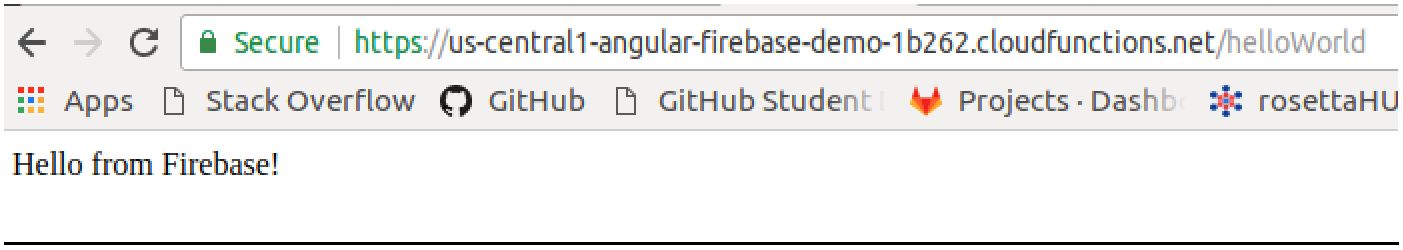


```
=== Deploying to 'angular-firebase-demo-1b262'...

i  deploying functions
i  functions: ensuring necessary APIs are enabled...
✓  functions: all necessary APIs are enabled
i  functions: preparing functions directory for uploading...
i  functions: packaged functions (49.46 KB) for uploading
✓  functions: functions folder uploaded successfully
i  functions: creating function helloWorld...
✓  functions[helloWorld]: Successful create operation.
Function URL (helloWorld): https://us-central1-angular-firebase-demo-1b262.cloudfunctions.net/helloWorld

✓  Deploy complete!

Project Console: https://console.firebase.google.com/project/angular-firebase-demo-1b262/overview
```

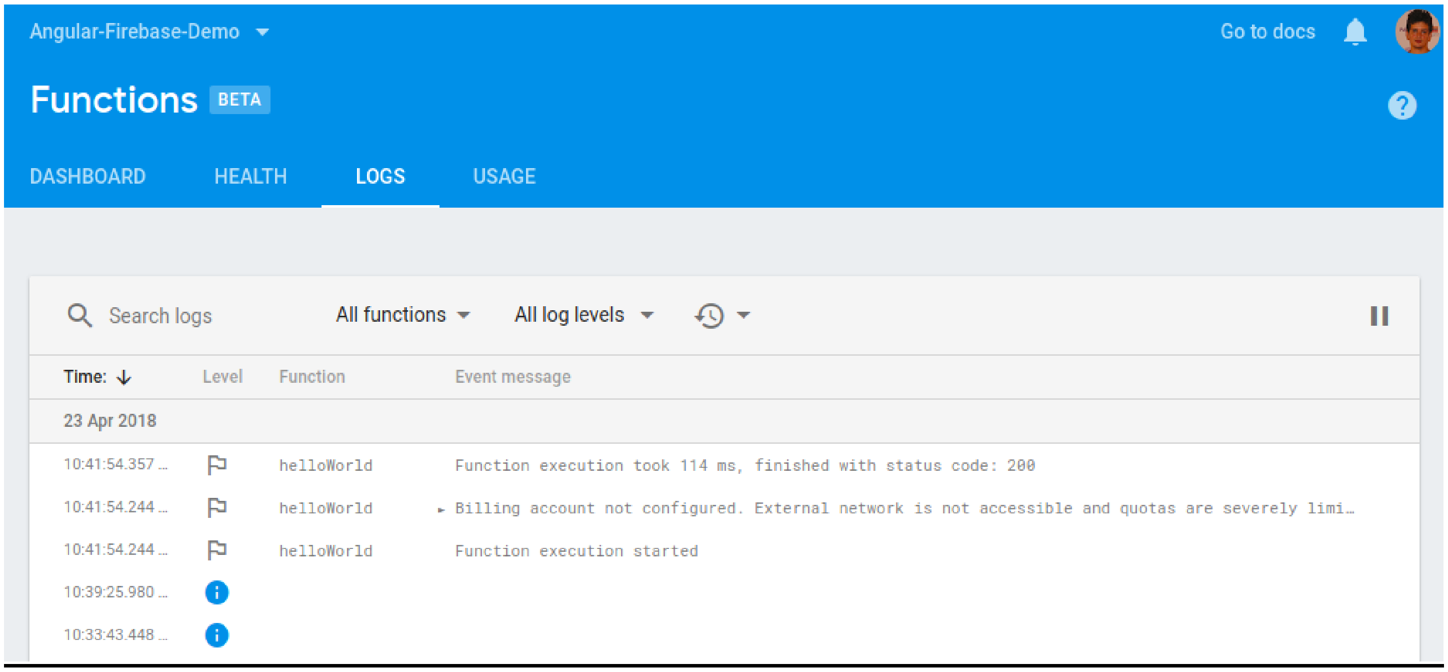
Preview:



Function	Event	Executions
helloWorld	HTTP Request https://us-central1-angular-firebase-demo-1b262.cloudfunctions.net/helloWorld	–


Secure
https://us-central1-angular-firebase-demo-1b262.cloudfunctions.net/helloWorld
Apps
Stack Overflow
GitHub
GitHub Student
Projects · Dashb
rosettaHU

Hello from Firebase!



```
Time: ↓        Level  Function     Event message
23 Apr 2018
10:41:54.357 …        helloWorld   Function execution took 114 ms, finished with status code: 200
10:41:54.244 …        helloWorld   ▸ Billing account not configured. External network is not accessible and quotas are severely limi…
10:41:54.244 …        helloWorld   Function execution started
10:39:25.980 …
10:33:43.448 …
```