

Firestore: Getting started

Firestore is a service that works on top of [Google Cloud](#) to bring the benefits of serverless architecture to the end-user, so that they don't have to worry about managing or scaling servers and can focus on the problem at hand.

Table of contents

- [Prerequisites](#)
 - [Firestore Registration](#)
 - [Firestore project creation](#)
 - [Firestore services configuration](#)
 - [Firestore Storage](#)
 - [Firestore Authentication](#)
 - [Firestore Database](#)
 - [Firestore Functions](#)
 - [Firestore Hosting](#)
 - [Web application configuration](#)
- [CLI tools installation](#)
 - [Firestore files](#)
- [Firestore functions development](#)
- [Firestore deploy](#)

Prerequisites

Firestore Registration

In order to get firestore working, we first need to register an account with them. This can be done from the following address:

<https://firebase.google.com/>

Firestore project creation

- Once an account has been created, we need to go to the *console* by clicking on the **GO TO CONSOLE**:



Search

[GO TO CONSOLE](#)

ful, realtime crash reporting. [Learn more](#)



1S

- Once in the console, we proceed to create a new project:

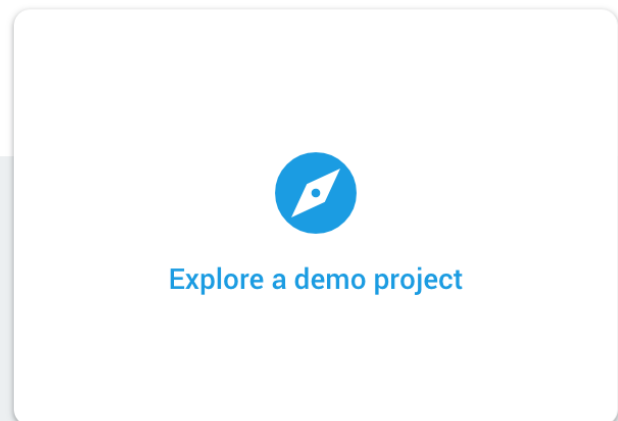
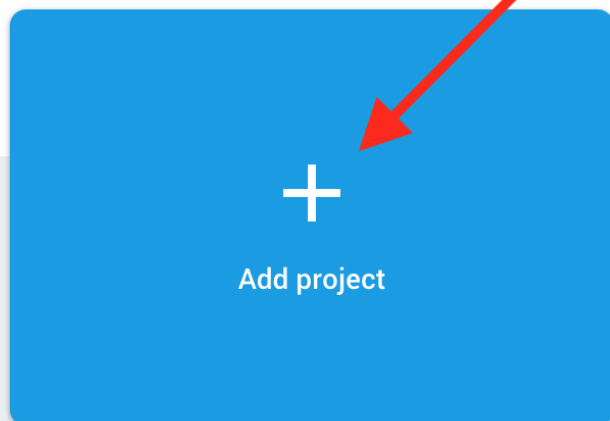
Welcome to Firebase!

Tools from Google for developing great apps, engaging with your users, and earning more through mobile ads.

[🔗 Learn more](#) [☰ Documentation](#) [💬 Support](#)



Your projects using Firebase



- Choose a project name (don't worry about duplicates since Firebase can handle them) and choose the Country/region that you reside in (this has nothing to do with where Google will host the services).

Firebase services configuration

Once a project has been created, you'll be presented with a dashboard that contains all of the services that Firebase offers, along with a few useful links to help you setup your project.

A screenshot of the Firebase console dashboard. The left sidebar is dark grey and contains a 'DEVELOP' section with icons and links for Authentication, Database, Storage, Hosting, Functions, and ML Kit. Below this is a 'STABILITY' section with links for Crashlytics, Performance, and Test Lab. The main content area has a blue background with the text 'Welcome to Firebase!' and 'Get started here.' in white. Below this text are three white circular icons: 'iOS', the Android robot, and a code symbol '</>'. Under each icon is the text 'Add Firebase to your [platform] app'.

For the sake of this project, we'll show how to configure the web client for Firebase, along with the console tools used to work and deploy Firebase features.

Firebase Storage

Upon selecting **Storage** in the sidebar, we are presented with an initial screen that lets us read about what this service offers, access to the documentation and a *Get started* link.



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

[Learn more](#)

[View the docs](#)

GET STARTED

Currently, pressing the *Get started* link, gives an alert showing that by default, **Storage** is configured to only allow authorised users to read or write to our **Storage** instance.

Note: For our use-case, we decided that this was too restrictive, since we wanted anonymous users to have read access to the files, so we ended up modifying this default.

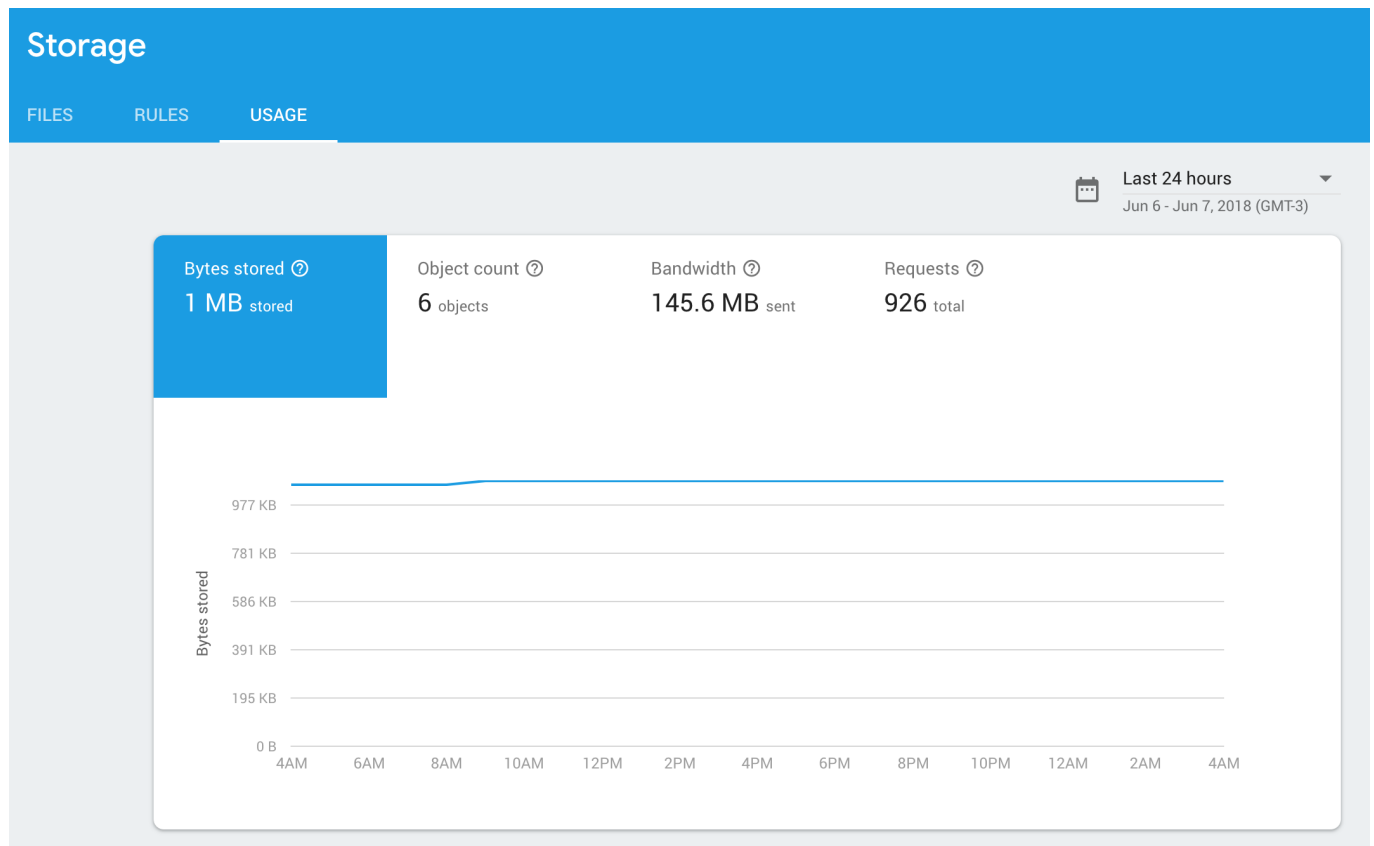
Firebase presents us with a screen that shows the state of our storage **bucket**. Here, we can:

- Upload files
- Download files
- Delete files
- Create directories

gs://serverless-itba.appspot.com	lefwA9Yaw...r0ZzZpDB52	UPLOAD FILE		
<input type="checkbox"/>	Name	Size	Type	Last modified
<input checked="" type="checkbox"/>	9781929570010.jpg	27.99 KB	image/jpeg	Jun 7, 2018
<input type="checkbox"/>	thumb_9781929570010.jpg	7.54 KB	image/jpeg	Jun 7, 2018
<input type="checkbox"/>	thumb_WhatsApp Image 2018-04-07 at 22.34.18.jpeg	14.58 KB	image/jpeg	Jun 6, 2018
<input type="checkbox"/>	WhatsApp Image 2018-04-07 at 22.34.18.jpeg	164.81 KB	image/jpeg	Jun 6, 2018

Also here we can access the **Rules** for our Storage service has configured and usage information, such as how much bandwidth was used, how many requests we had and how many files were created for any given period of

time.



Firestore Authentication

The **Authentication Service** provides a similar Getting started screen as the **Storage Service**, but upon entering this service, we can see that what we are presented is quite different.

From here, we'll be able to:




- Choose what types of Authentication mechanisms we'll want to enable for our application
- Customize certain aspects of:
 - user registration
 - password reset
 - email server settings
 - etc.
- See every user that has registered through Firebase
- Enable or disable registered users

Authentication

WEB SETUP

USERS SIGN-IN METHOD TEMPLATES USAGE

ADD USER ↺ ⋮

Identifier	Providers	Created	Signed In	User UID ↑
████████@gmail.com		May 30, 2018	Jun 2, 2018	████████████████████
████████@itba.edu.ar		May 30, 2018	Jun 7, 2018	████████████████████
████████@gmail.com		Jun 2, 2018	Jun 2, 2018	████████████████████

Rows per page: 50 1-3 of 3 < >

Firebase Database

Firebase offers two database services: *Realtime Database* and *Cloud Firestore*. For this project we'll focus on the latter.

As with *Storage*, once we press the *Get started* button, we are prompted about the default security rules to use with our database. There are two defaults presented, one is specifically focused for development purposes by allowing read and writes no matter their authentication status and the other one is **strict**, meaning that reads and writes are disabled. This last option is intended to be built upon so that you only allow reads or writes to specific resources for specific users.

Security rules for Cloud Firestore



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

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write;
    }
  }
}
```



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

Enabling Cloud Firestore Beta will preclude you from using Cloud Datastore with this project, notably from the associated App Engine app.

CANCEL

ENABLE

Once we get passed the security prompt, we are greeted by the *Firestore* dashboard:

The screenshot shows the Firebase Database console interface. At the top, there's a blue header with 'Database' and 'Cloud Firestore BETA'. Below the header are tabs for 'DATA', 'RULES', 'INDEXES', and 'USAGE'. The main content area shows a breadcrumb path: 'images > 06RkchdV22A5YQSHnN2'. Below this, there are three panels. The left panel shows the 'serverless-itba' database with a '+ ADD COLLECTION' button and a list of collections, including 'images'. The middle panel shows the 'images' collection with a '+ ADD DOCUMENT' button and a list of documents, including '06RkchdV22A5YQSHnN2'. The right panel shows the details of the selected document, including fields like 'author', 'name', 'pic', 'uid', 'imagePath', 'thumbPath', and 'uploadTime'.

Here we can:

- View the current state of our document database
- Create/Edit/Delete database records
- Manage our security rules
- View database usage
- Create/Edit/Delete database indexes


Firestore Functions



In the *Functions* section, we can get all of the information relevant to each of the cloud functions defined such as:

- List of enabled functions along with the specified trigger
- Health (amount of errors, performance issues, etc)
- Logs (anything logged from within our functions get displayed here)
- Usage (function invocation for any given period of time)

Functions BETA ?

DASHBOARD HEALTH LOGS USAGE

 **Current billing period** ▼
Jun 1 - Jun 30, 2018 (GMT-3)

Function	Event	Executions	Median run time
generateThu...	 google.storage.object.finalize serverless-itba.appspot.com	446 	57.40ms

Firestore Hosting

Firestore hosting sections handles everything related to the static-website hosting service. In here you can see a list of past releases as well, manage the domain that your website will use and see the bandwidth usage for a certain period of time.

Hosting ?





DASHBOARD USAGE

Domains

CONNECT DOMAIN

Domain	Status
serverless-itba.firebaseio.com Default	

Deployment history

Status	Time	Deploy	Files
 Current	Jun 7, 2018 2:58 AM	 [REDACTED]@gmail.com WUPC2z	34
 Deployed	Jun 7, 2018 2:54 AM	 [REDACTED]@gmail.com NaaAem	34

To be able to use each of the services configured previously, we need to retrieve the configuration block for our project. This block will be used to initialize the Firebase web client so that it correctly uses all of our services.

For us to be able to get this file, we'll go to the *Firebase Project Overview* and click the *Add Firebase to your web app* button.

Upon clicking the button, we are presented with a screen that shows us the configuration block required, as well as presents us with some links that will help us configuring the Firebase client:

```
<script src="https://www.gstatic.com/firebasejs/5.0.4/firebase.js"/>
<script>
  // Initialize Firebase
  var config = {
    apiKey: "...",
    authDomain: "serverless-itba.firebaseio.com",
    databaseURL: "https://serverless-itba.firebaseio.com",
    projectId: "serverless-itba",
    storageBucket: "serverless-itba.appspot.com",
    messagingSenderId: "..."
  };
  firebase.initializeApp(config);
</script>
```

That snippet is the general way to add Firebase to a web page, but since we are using React, which supports ES6 Javascript, we'll do it a bit differently:

```
import firebase from "firebase/app";
/* We need the following imports since they have side-effects
 * on the global firebase object and add the required functionality to it
 */
import "firebase/auth";
import "firebase/firestore";
import "firebase/storage";

const config = {
  apiKey: "...",
  authDomain: "serverless-itba.firebaseio.com",
  databaseURL: "https://serverless-itba.firebaseio.com",
  projectId: "serverless-itba",
  storageBucket: "serverless-itba.appspot.com",
  messagingSenderId: "..."
};

firebase.initializeApp(config);
```

Once that code is run, we'll be able to access each of the imported services without the need for specialized code:

```
firebase.firestore().ref().child("images")...;  
// ...  
firebase.auth().signInWithRedirect(...)  
  .then(result => {...})  
  .catch(error => {...});
```

CLI tools installation

For us to be able to deploy Firebase security rules, services configuration, static website or functions, we'll need to install the `firebase-tools` javascript package.

This requires either *Yarn* or *NPM* to be installed.

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

Once the tools are installed, we need to login to Firebase so that we have deploy permissions:

```
firebase login
```

After we are logged in, it's time to initialize Firebase so that it can create every directory and file it needs for the services we have configured. The following command will prompt us with a series of questions in order to better suit our specific project:

```
firebase init
```

After initializing Firebase, we need to specify which project we'll use, to do this, we run the following command with the project id as an argument:

```
firebase use serverless-itba
```

Now we are ready to develop and deploy

Firebase files

Firebase will create the following files:

- `storage.rules` Contains Storage security rules
- `firestore.rules` Contains Firestore security rules
- `database.rules.json` Contains database security rules
- `firestore.indexes.json` Contains Firestore index definitions
- `public/` Files under this directory will be deployed to Firebase Hosting
- `functions/` Files under this directory will be deployed to Firebase Functions

- All of our cloud functions will be defined inside here
- `firebase.json` Contains settings for each of the services, such as pre and post deploy scripts and configuration (rules/indexes) mapping for each service.

Firebase functions development

For this project, we chose to develop Firebase functions using *Typescript*. Since no browser runs *Typescript*, there are some extra steps that the Firebase CLI tools do automatically for us to transpile the Typescript code to Javascript.

Function code is implemented inside the `functions/src/` folder.

In the `index.ts` file we add an include guard that only imports the `generateThumbnail` function if it wasn't imported previously.

In the `generateThumbnail.ts` file is where the cloud function is defined.

Here we import the required Firebase libraries:

```
import * as functions from "firebase-functions";
import * as admin from "firebase-admin";
// Utility imports to do image and file path manipulation
import * as path from "path";
import * as sharp from "sharp";
```

Once we have `functions` imported, we can start defining each of our functions using any of the available triggers (HTTP, Storage, etc.)

An example of this is:

```
/**
 * Trigger for our storage service that is called
 * whenever a file finished uploading, passing to our
 * callback function the object that was added along
 * with metadata such as bucket that received the file,
 * file name, file content type, etc.
 */
functions.storage.object().onFinalize(object => {...})
```

You can see the whole implementation in [generateThumbnail.ts](#)

Firebase deploy

Once we have finished creating cloud functions, defining security rules, database indexes, etc., we are ready to do our first deploy.

The basic way to deploy to firebase is by running the following command:

```
firebase deploy
```

That function will deploy the configuration files along with the required files for all of our configured services. Meaning that it will:

- Deploy our static website residing inside the **public** directory
- Deploy all of our functions inside the **functions** directory
- Update security rules for the database and storage

If you don't want to deploy everything, **firebase deploy** allows you to specify which services you want to deploy:

```
firebase deploy --only hosting
```