Cloud Functions for Firebase

Cloud Functions for Firebase is a server-less framework that lets you automatically run backend code in response to events triggered by Firebase features and HTTPS requests. Your JavaScript or TypeScript code is stored in Google's cloud and runs in a managed environment, meaning there is no need to manage and scale your own servers.

Check the official page for more information.

Setup

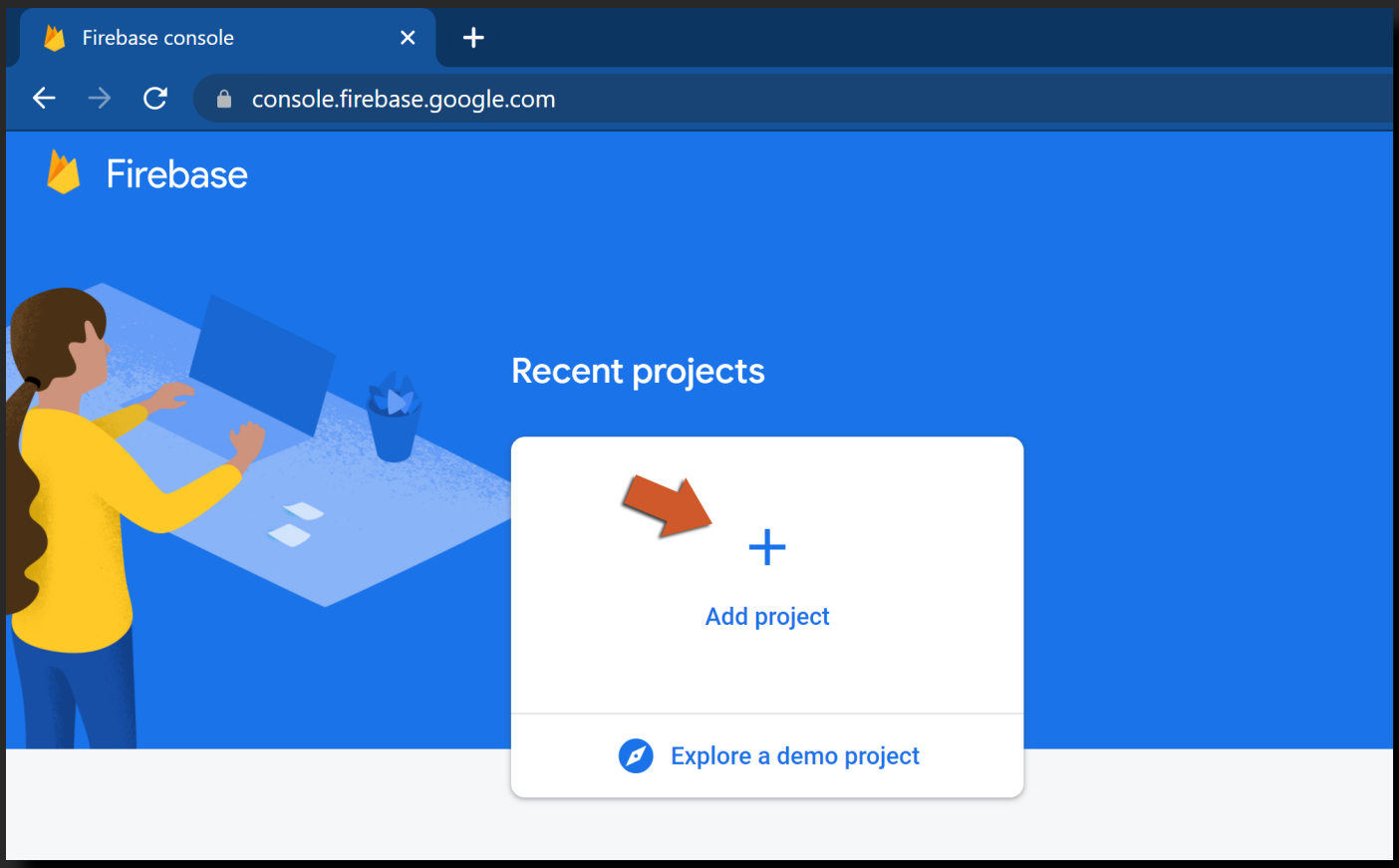
Before starting to use any Firebase extensions, you are required to follow some initial configuration steps. However unlike most of the modules this is purely server-side, as you will be creating JavaScript functions that you can later call using the http_request function from inside your GameMaker Studio project.

- Create Project
- Firebase Console
- Create and Deploy (creating functions with JavaScript and deploying them)

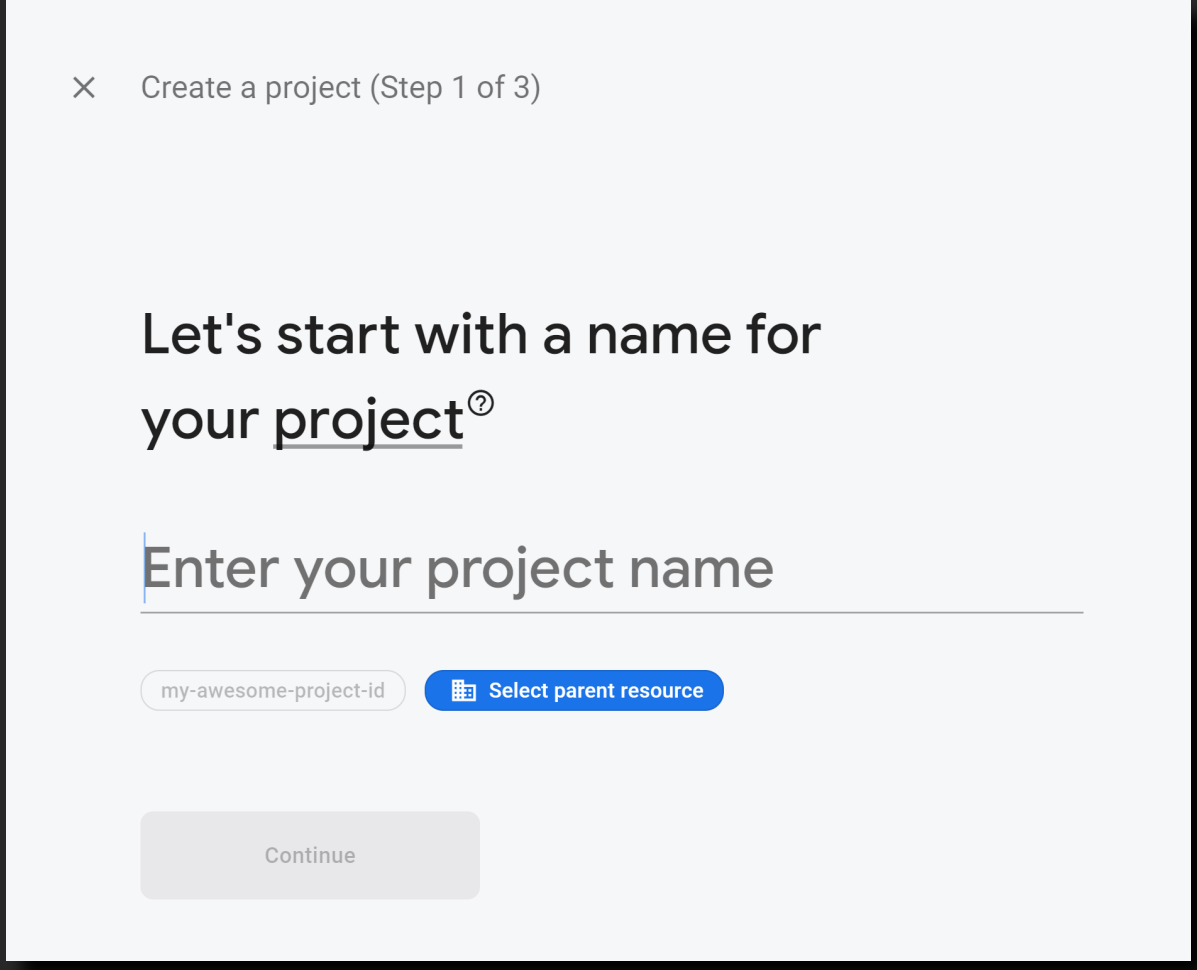
Create Project

Before working with any Firebase functions, you must set up your Firebase project:

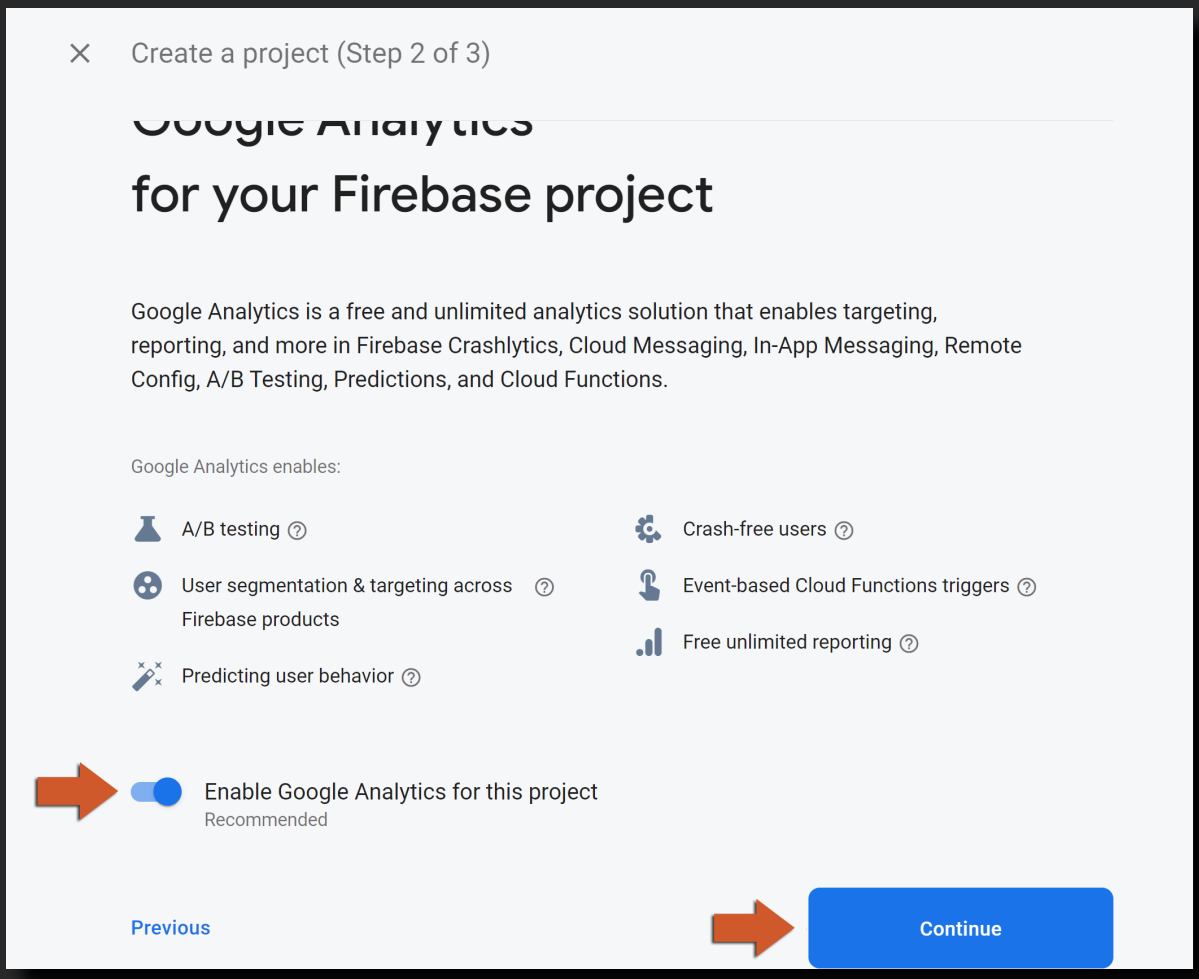
1. Go to the **Firebase Console** web site.
2. Click on **Add Project** to create your new project.



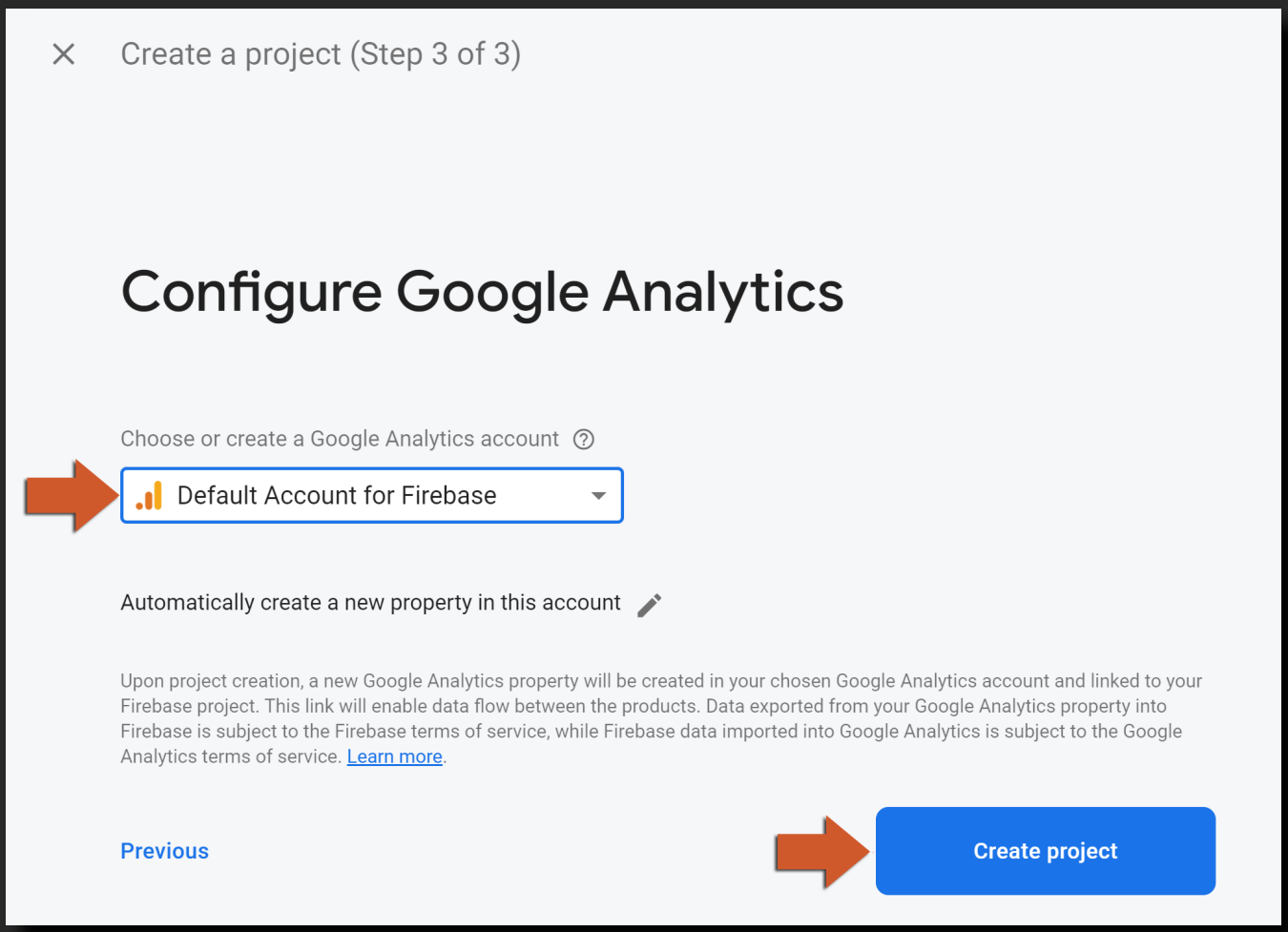
3. Enter a name for your project and click on the **Continue** button.



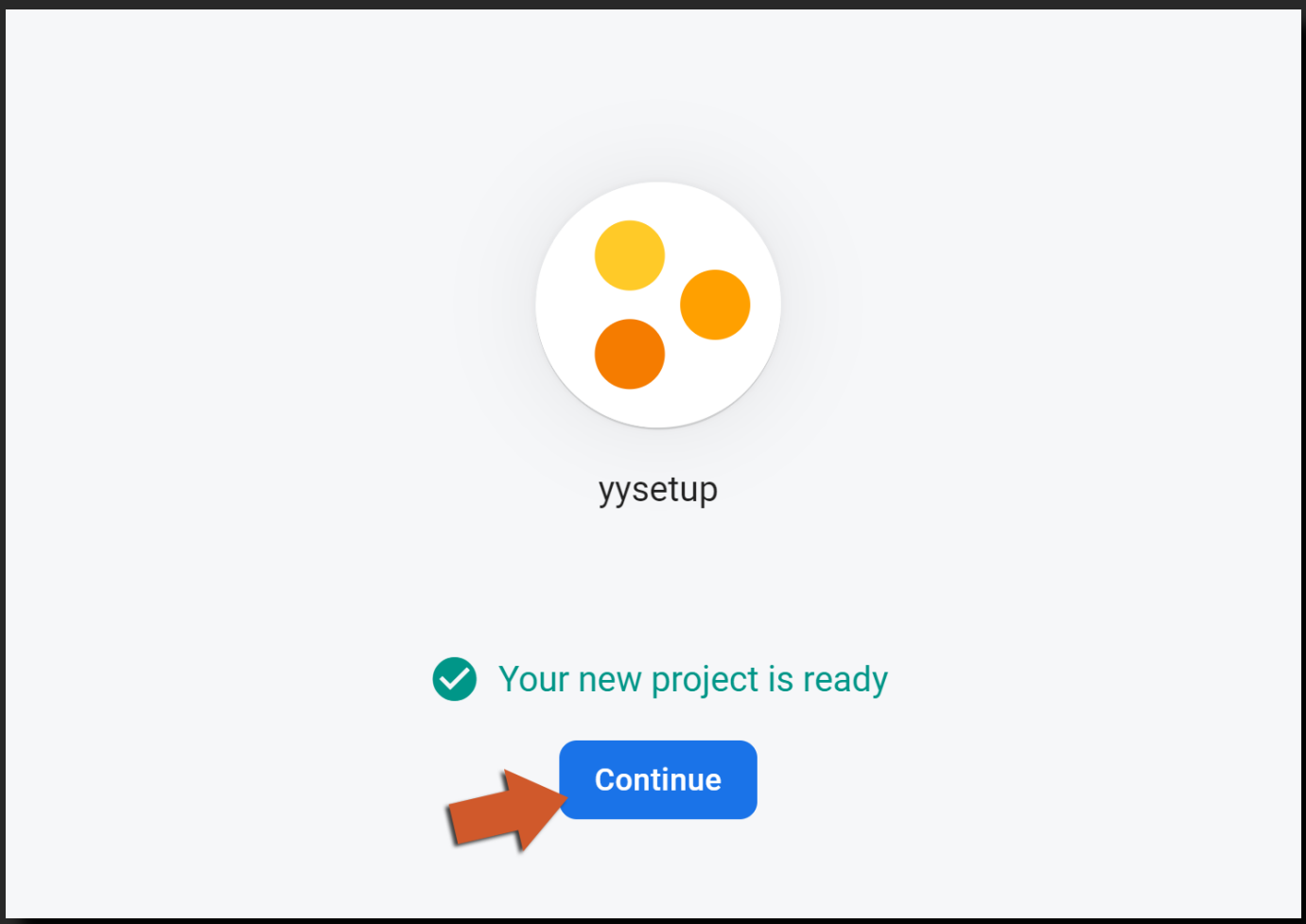
4. On the next page, make sure that **Enable Google Analytics for this project** is enabled and then click the **Continue** button:



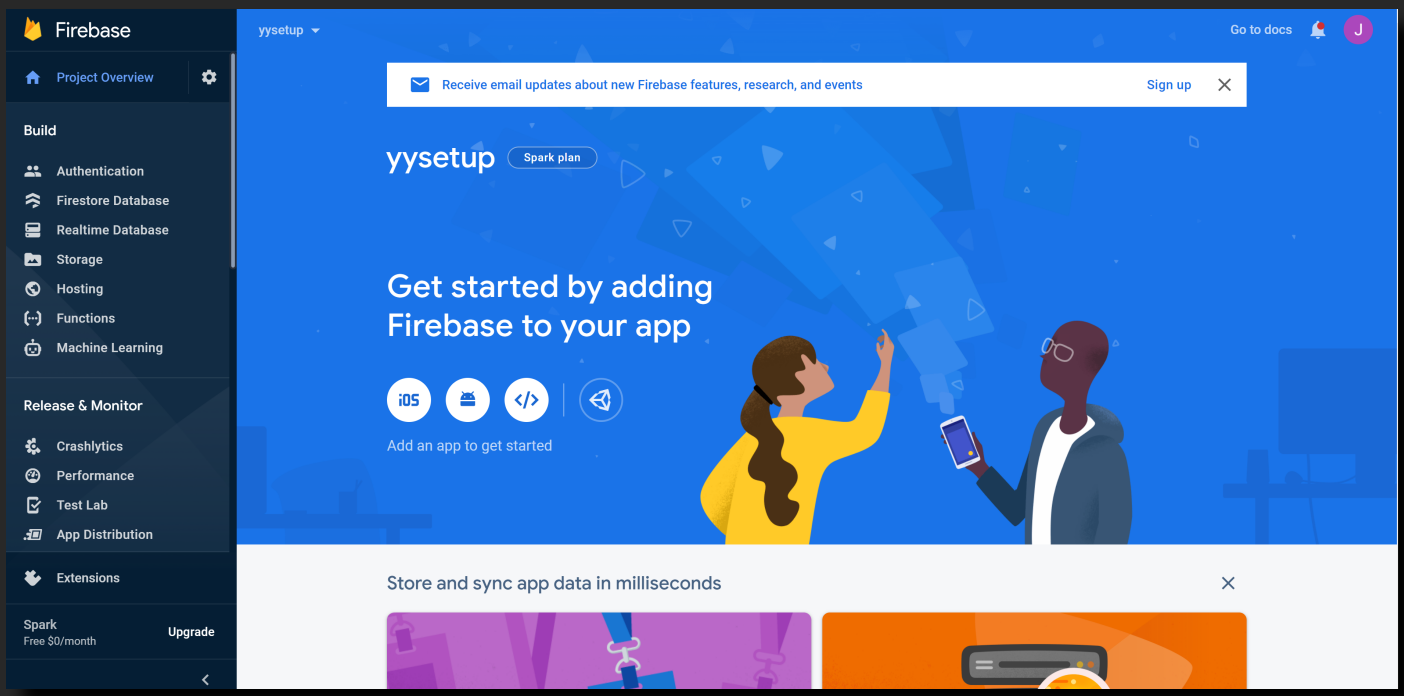
5. Select your account and click the **Create project** button:



6. Wait a moment until you project is created; after a few moments you should see the page shown below:



7. You will now be taken to your new project's home page:

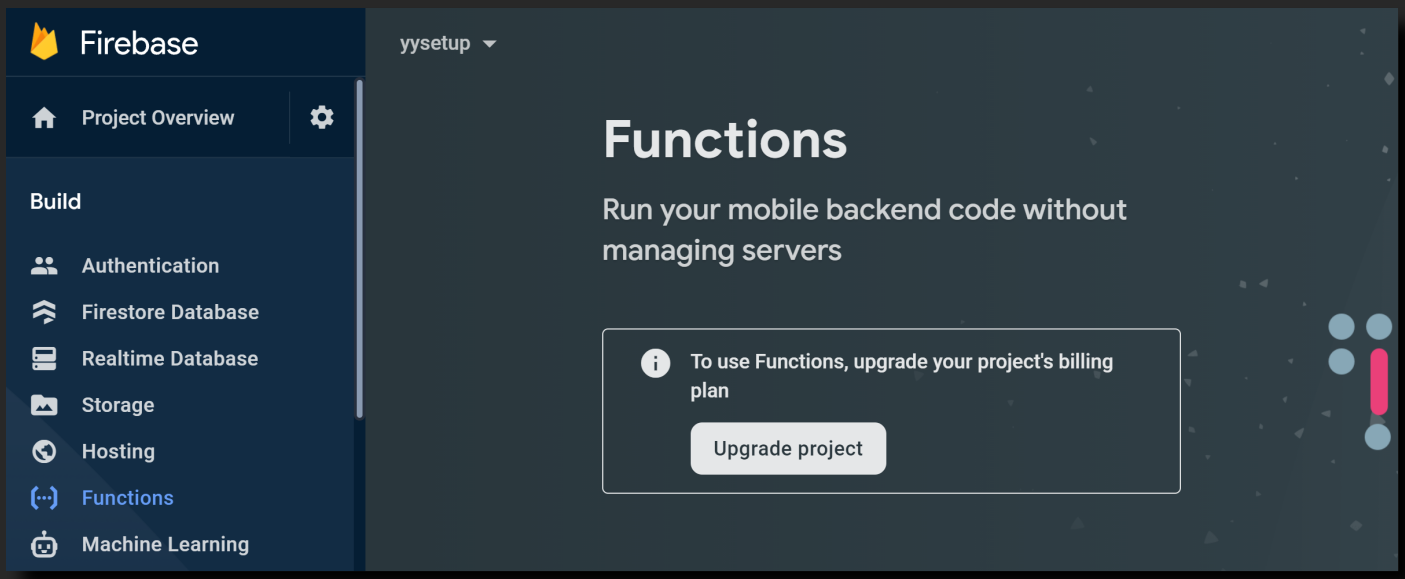


8. Continue your adventure with the Firebase extensions provided for GameMaker!

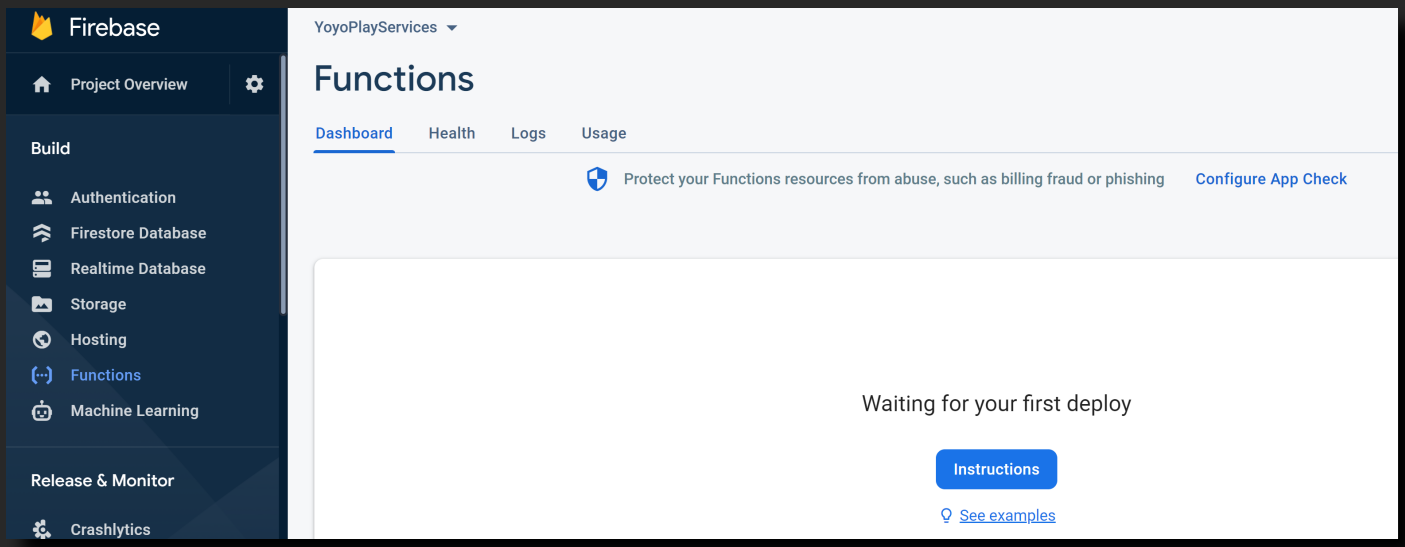
Firebase Console

This guide covers setting up and enabling Cloud Functions from inside the **Firebase Console**.

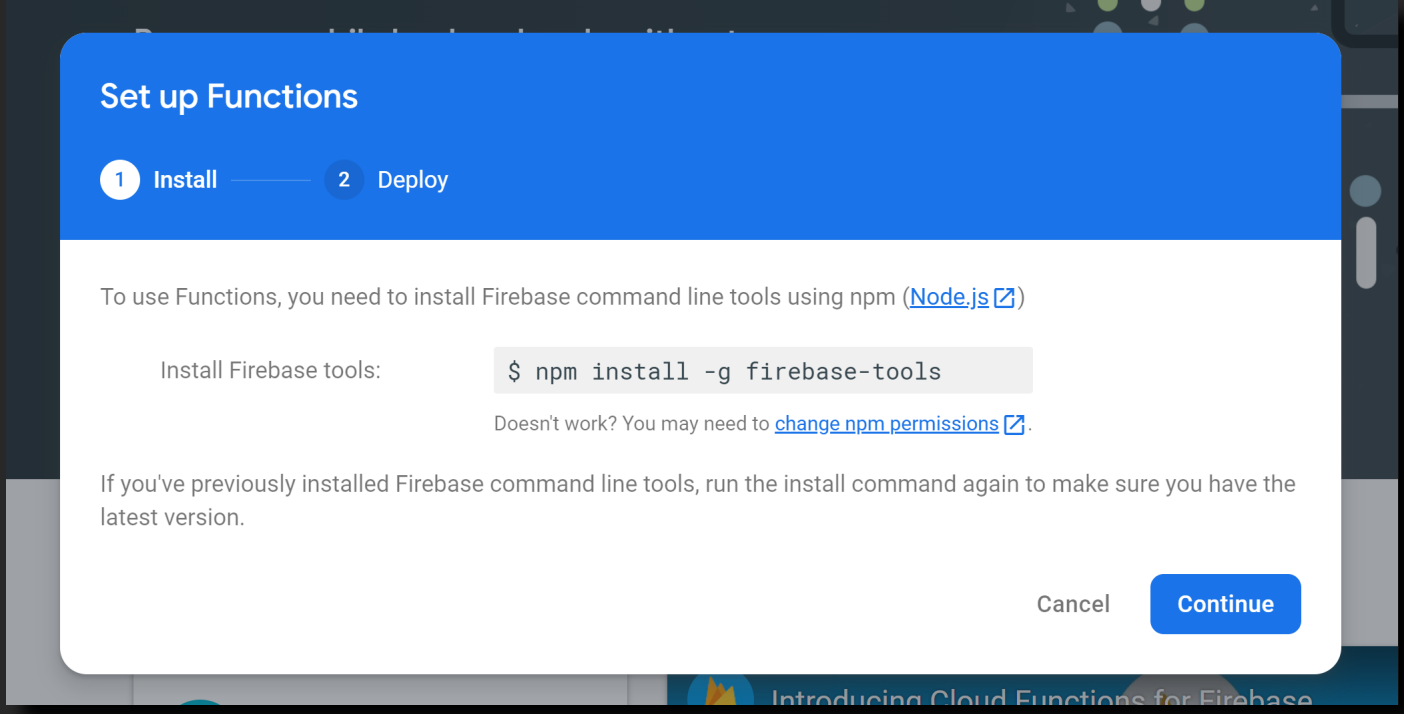
1. Head over to the **Functions** section, however to use this you will need to have upgraded your project to Blaze (pay as you go).



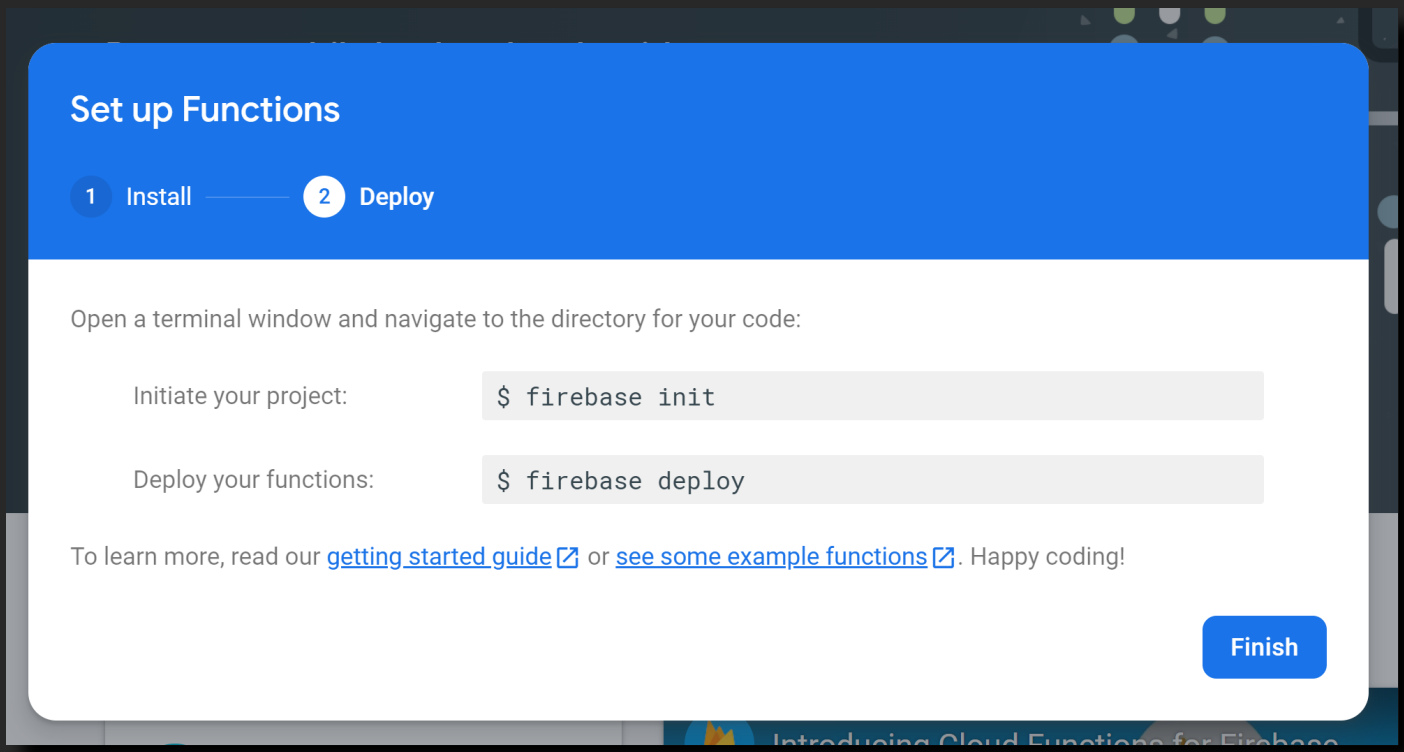
2. After upgrading the project you will see an **Instructions** button that you need to click on:



3. The first instruction is to use `npm install -g firebase-tools` in the command line (you will need to install **NodeJS** for this).



4. After some loading you need to click on ***Continue***, after which you will see the following page:



5. We are done configuring the Firebase Console. We can now continue to **creating and deploying a Cloud Functions project**.

Create and Deploy

Firebase Cloud Functions have very little to do with GML itself and more with programming server-side functions in JavaScript (or TypeScript). This guide will get you going with creating your first function and deploying it to the Firebase Cloud Functions server.

1. We'll move to the folder where we want work using the `cd $Path` command (this will change the path you're working in):

```
Command Prompt
Microsoft Windows [Version 10.0.19041.685]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\[redacted]>cd C:\Users\[redacted]\Desktop\CloudFunctions
```

2. We now need to call the `firebase init` command:

```
\Users\chuyz\Desktop\CloudFunctions>firebase init

   ######## #### ########  ######## ########     ###     ######  ########
   ##        ##  ##     ## ##       ##     ##  ##   ##  ##       ##
   ######    ##  ########  ######   ########  #########  ######  ######
   ##        ##  ##    ##  ##       ##     ## ##     ##       ## ##
   ##       #### ##     ## ######## ########  ##     ##  ######  ########

u're about to initialize a Firebase project in this directory:

C:\Users\chuyz\Desktop\CloudFunctions

Are you ready to proceed? (Y/n)
```

3. You will be asked "**Are you ready to proceed?**", where your response should be **Y** (yes).

4. Now use the arrow keys to navigate the list and select **Functions** with the **Space** key; after that, press the **Enter** key to continue.

```
? Which Firebase features do you want to set up for this directory? Press Space to select features, then Enter to confir
m your choices.
 ( ) Realtime Database: Configure a security rules file for Realtime Database and (optionally) provision default instanc
e
 ( ) Firestore: Configure security rules and indexes files for Firestore
>(*) Functions: Configure a Cloud Functions directory and its files
 ( ) Hosting: Configure files for Firebase Hosting and (optionally) set up GitHub Action deploys
 ( ) Hosting: Set up GitHub Action deploys
 ( ) Storage: Configure a security rules file for Cloud Storage
(Move up and down to reveal more choices)
```

5. At this point we recommend using an existing Firebase project (as this will make configuration easier).

```
? Please select an option: (Use arrow keys)
> Use an existing project
  Create a new project
  Add Firebase to an existing Google Cloud Platform project
  Don't set up a default project
```

6. Now enter your **Project ID**, which you can get under **Project Settings** in the **Firebase Console** for your project.

```
? Please select an option: Use an existing project
? Please input the project ID you would like to use:
```

7. Choose **JavaScript** (for the purpose of this tutorial).

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

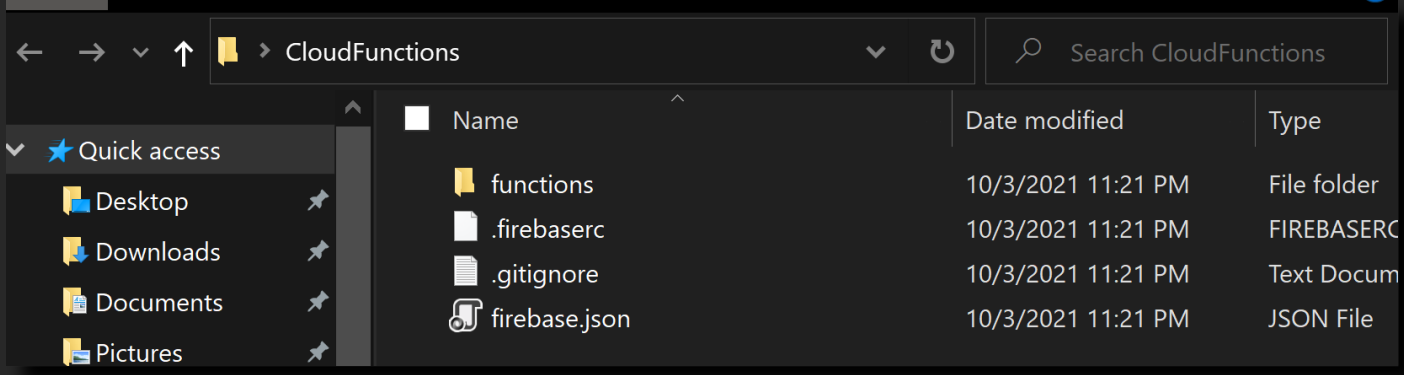
8. Choose **N** (unless you are experienced with JS and wish to use ESLint).

```
? What language would you like to use to write Cloud Functions? JavaScript
? Do you want to use ESLint to catch probable bugs and enforce style? (y/N)
```

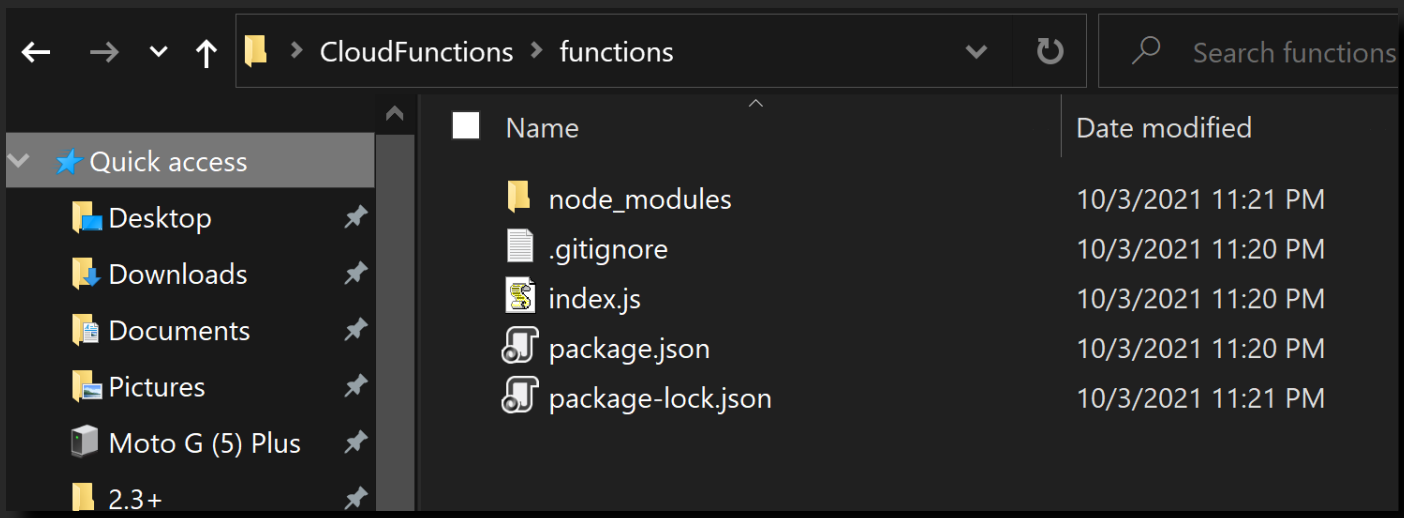
9. Finally select **Y** (for installing dependencies).

```
+  wrote functions/.gitignore
? Do you want to install dependencies with npm now? (Y/n)
```

10. After the process is over, you can see that your folder now has the following structure:



11. Inside the **functions** folder we can see a `index.js` file, which is where we will be declaring our functions.



12. In the `index.js` file you will see the following code, which creates a function called **helloWorld** and initialises its code.

```
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) => {
  functions.logger.info("Hello logs!", {structuredData: true});
  response.send("Hello from Firebase!");
});
```

> **NOTE** We can see that we are exporting a `helloWorld` function that registers to an `onRequest` event with the arguments `request` and `response`; these correspond to the "message sent by the client" (GameMaker Studio) and the "message we will send back", respectively.

13. Deploy it with `firebase deploy` command.

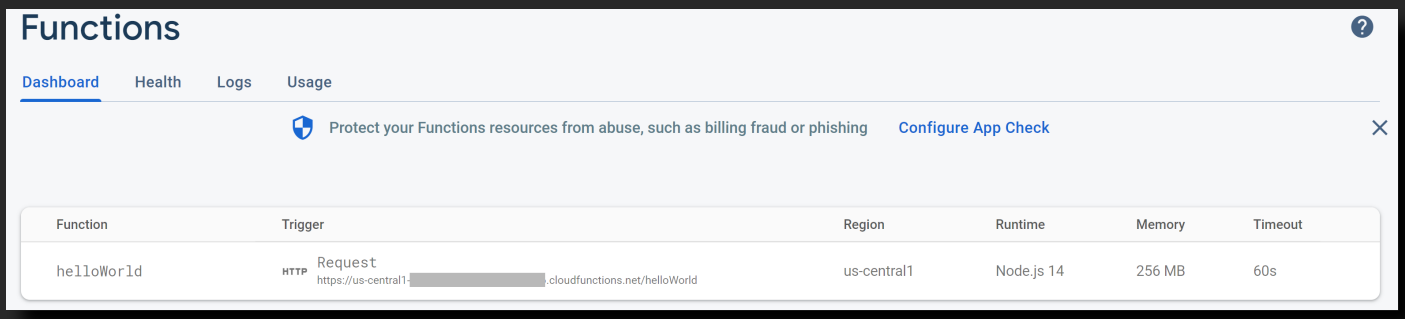
14. When the deploy process is done, you will see the "**Deploy complete**" message.

```
=== Deploying to '[redacted]'...

i  deploying functions
i  functions: ensuring required API cloudfunctions.googleapis.com is enabled...
i  functions: ensuring required API cloudbuild.googleapis.com is enabled...
+  functions: required API cloudfunctions.googleapis.com is enabled
+  functions: required API cloudbuild.googleapis.com is enabled
i  functions: preparing functions directory for uploading...
i  functions: packaged functions (24.57 KB) for uploading
+  functions: functions folder uploaded successfully
i  functions: creating Node.js 14 function helloWorld(us-central1)...
+  functions[helloWorld(us-central1)]: Successful create operation.
i  functions: cleaning up build files...
Function URL (helloWorld(us-central1)): https://us-central1-[redacted].cloudfunctions.net/helloWorld

+  Deploy complete!
```

15. After deploying has finished, you will see the function on the Firebase Console in the **Functions** tab:



16. This way you are able to publish your own functions to the Firebase Cloud Functions.