Diploma in
Information Technology

AY2018/2019
Semester 1 (Stage 3A)

Proposed Project: Directie

Project Deployment
Guide

Group 1

Members:

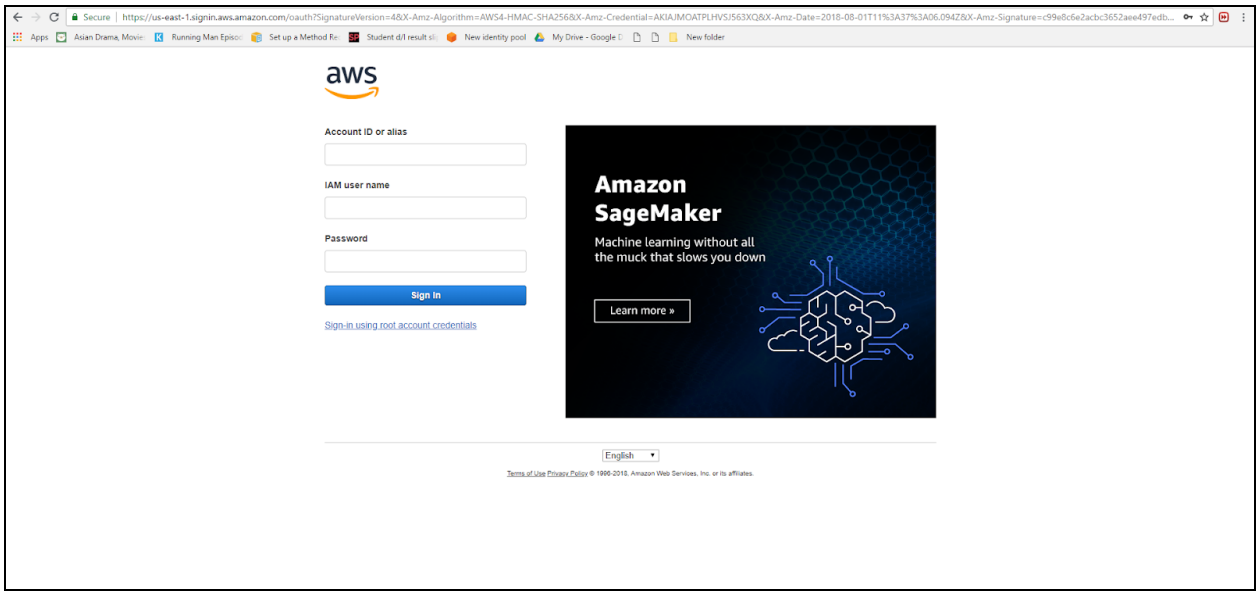
Ng Hong Yao (1625893)
Chia Kwee Cheng (1626120)
Tay Wei Sern (1626203)
Garick Chong (1626555)
Chua Wei (1639290)

Supervisor: Ms Lin Zhao

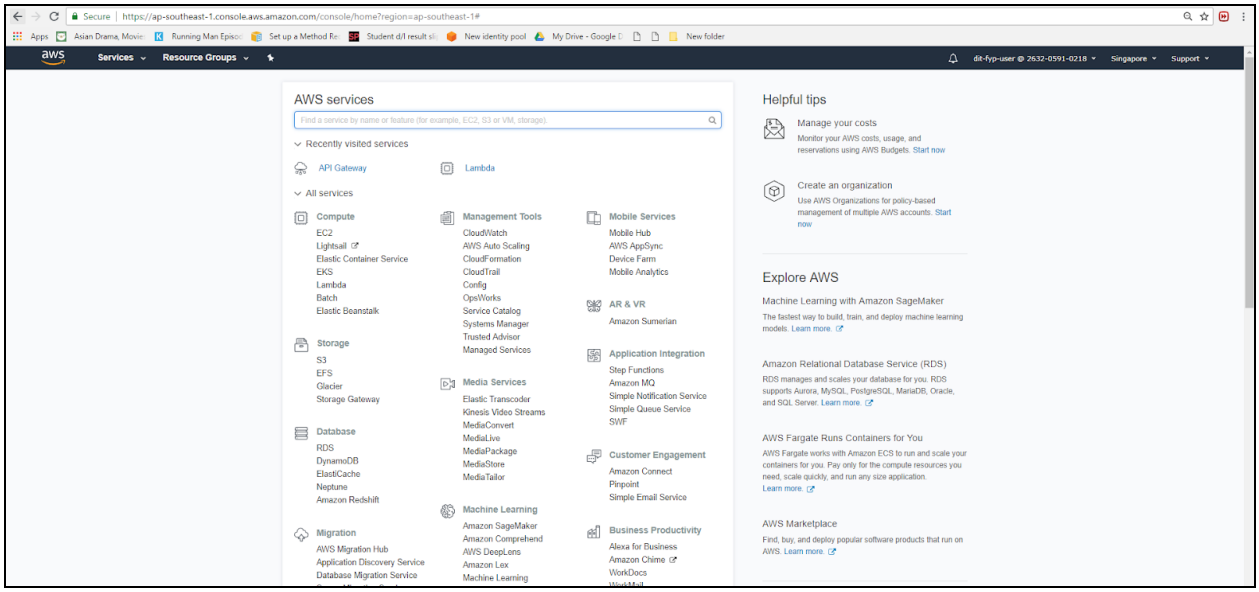
1.0 Setting up AWS Database

Before setting up AWS, you must sign up for an AWS Account if you do not have one.

Step 1 - Login to AWS Server

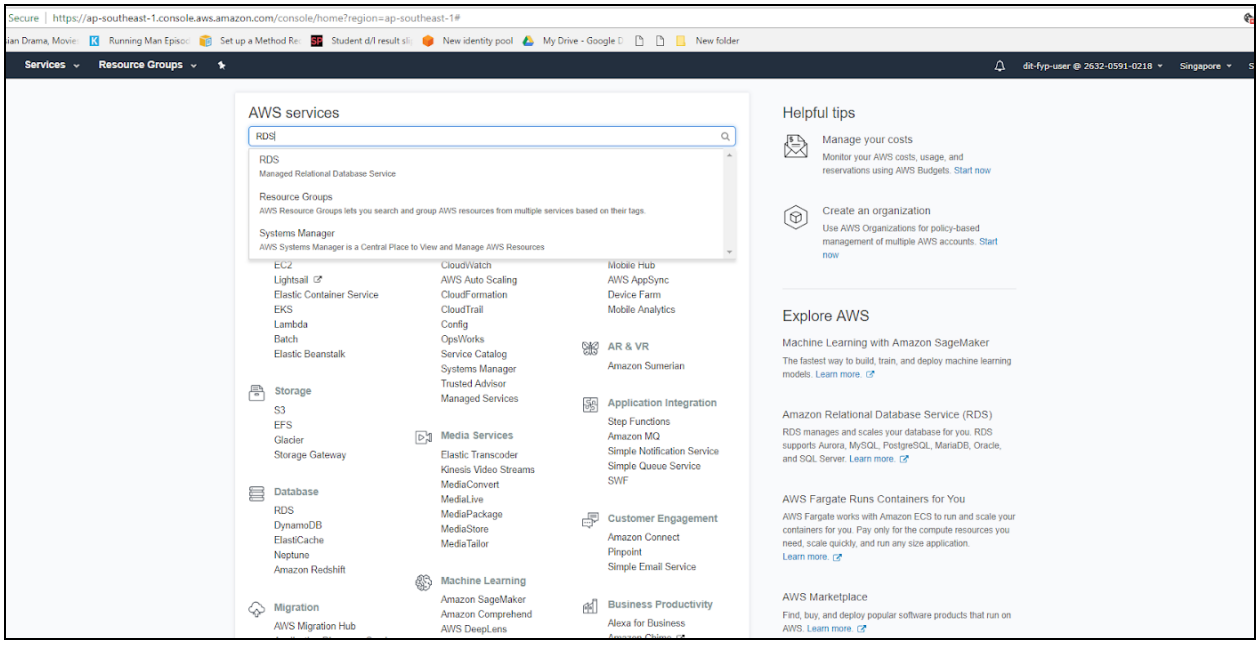


Go to https://aws.amazon.com/console/.

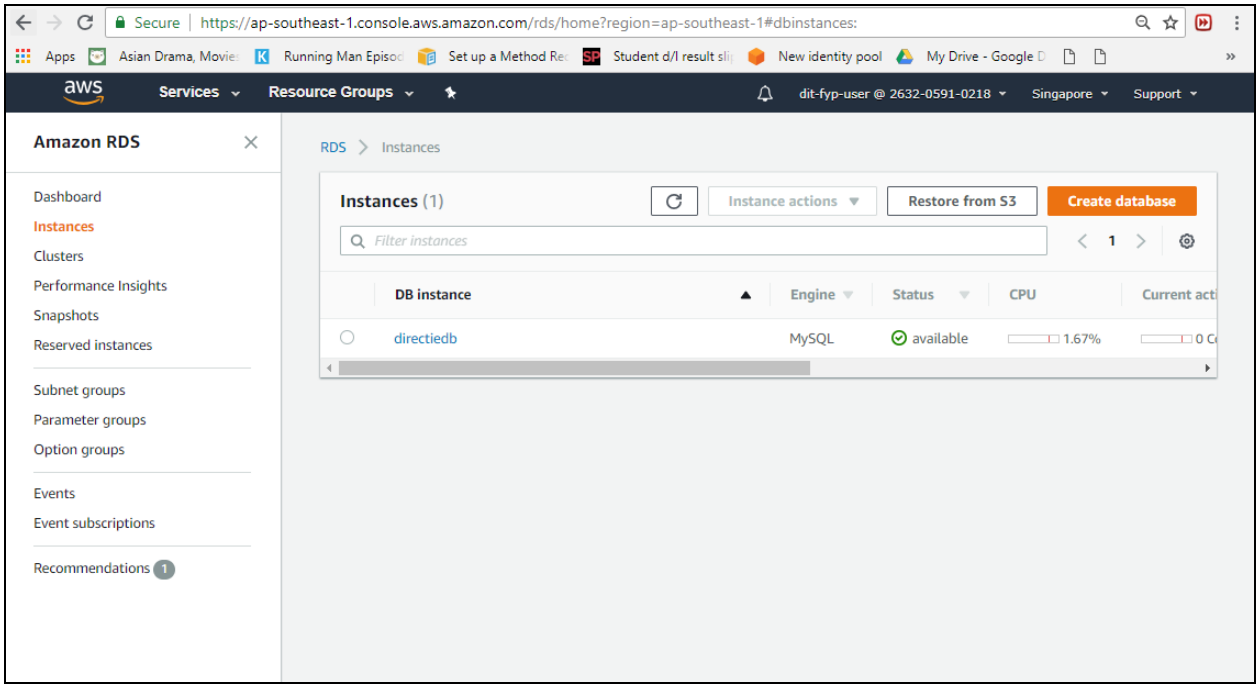


You should be redirected to the AWS console page.

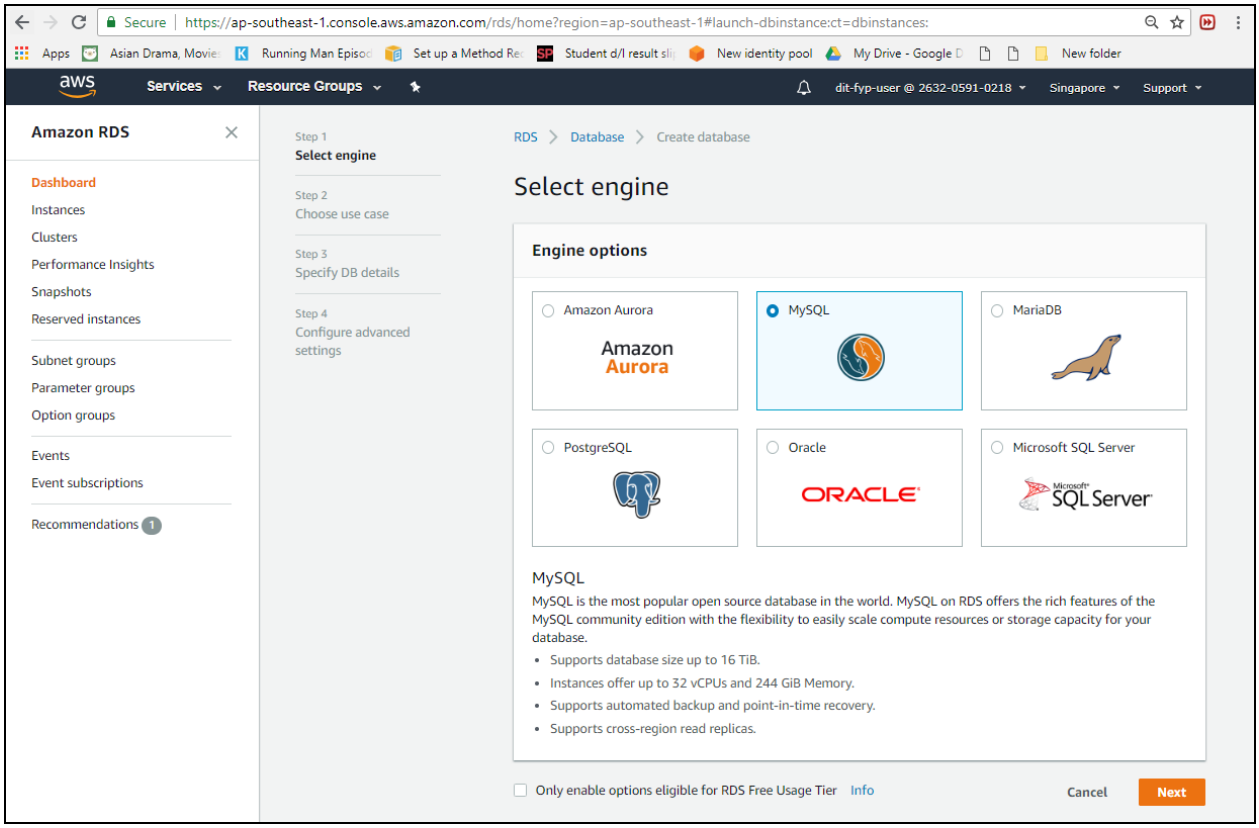
Step 2 - Click on RDS under the “Database” category or search “RDS”



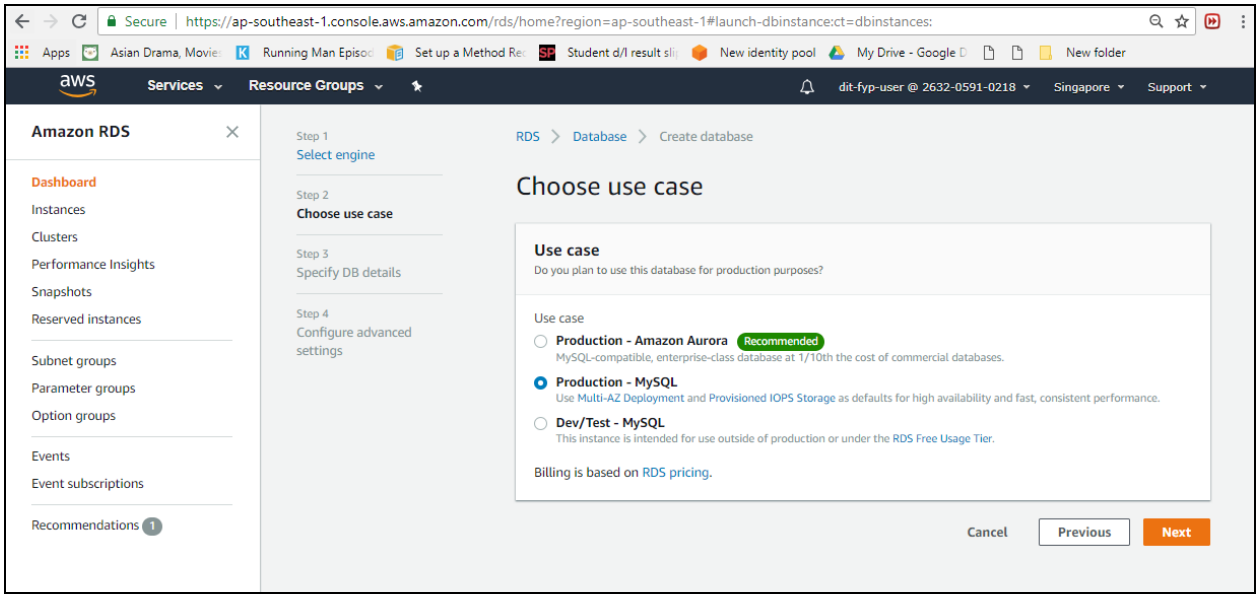
Step 3 - Click on instances on the left navigation bar and click on “Create database” button



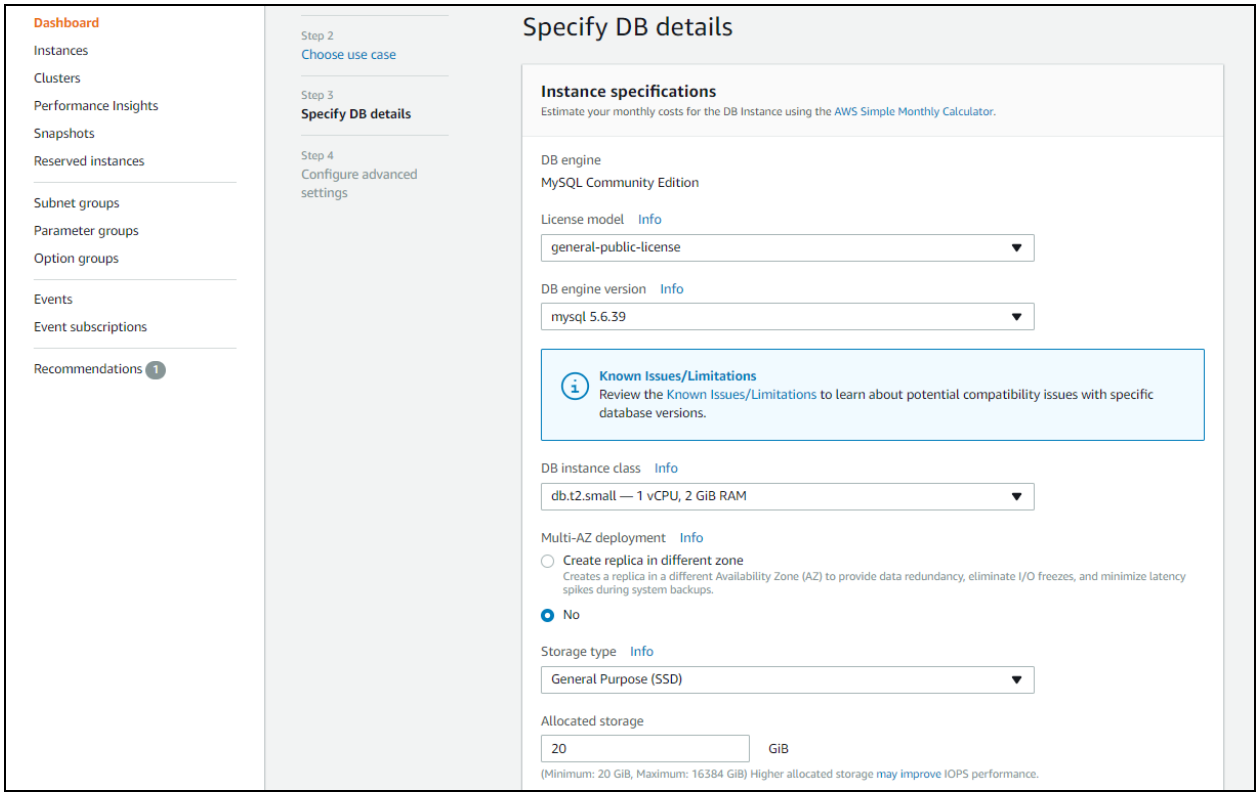
Step 4 - Select MySQL and proceed by clicking “Next” button



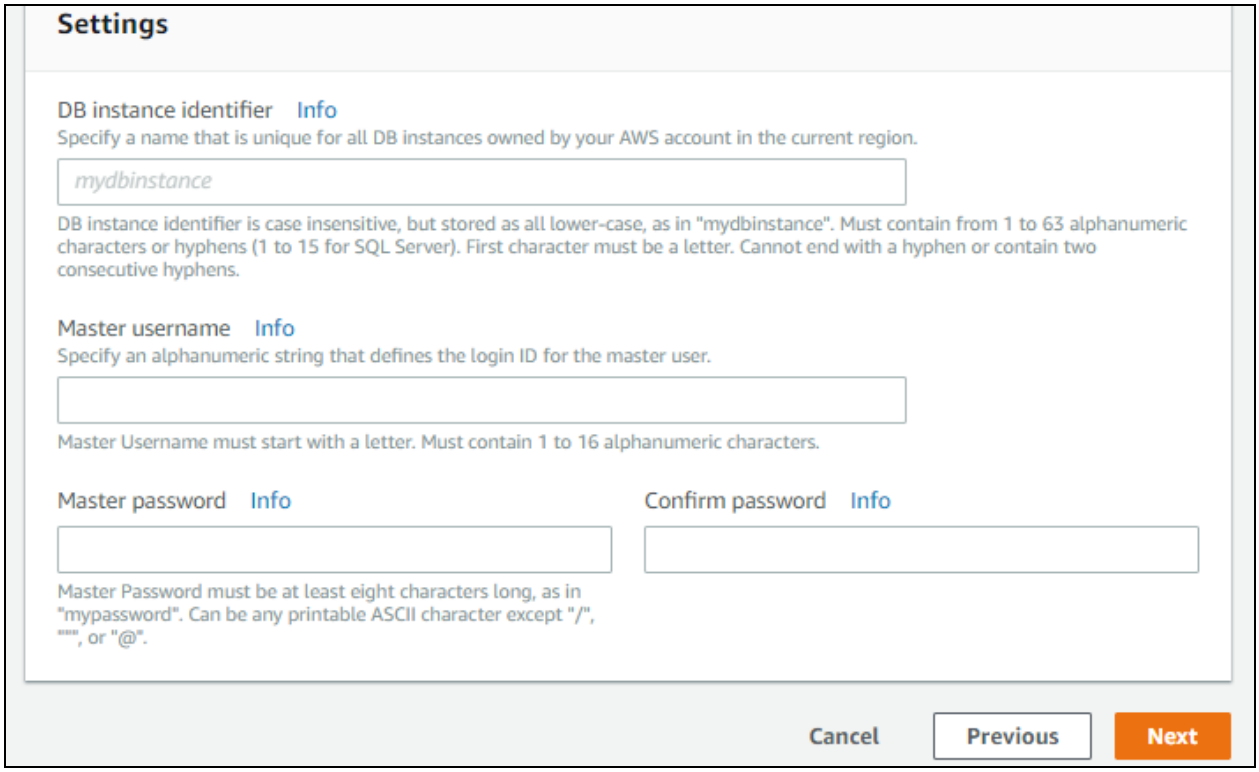
Step 5 - Choose Production-MySQL as Use Case and click “Next” button



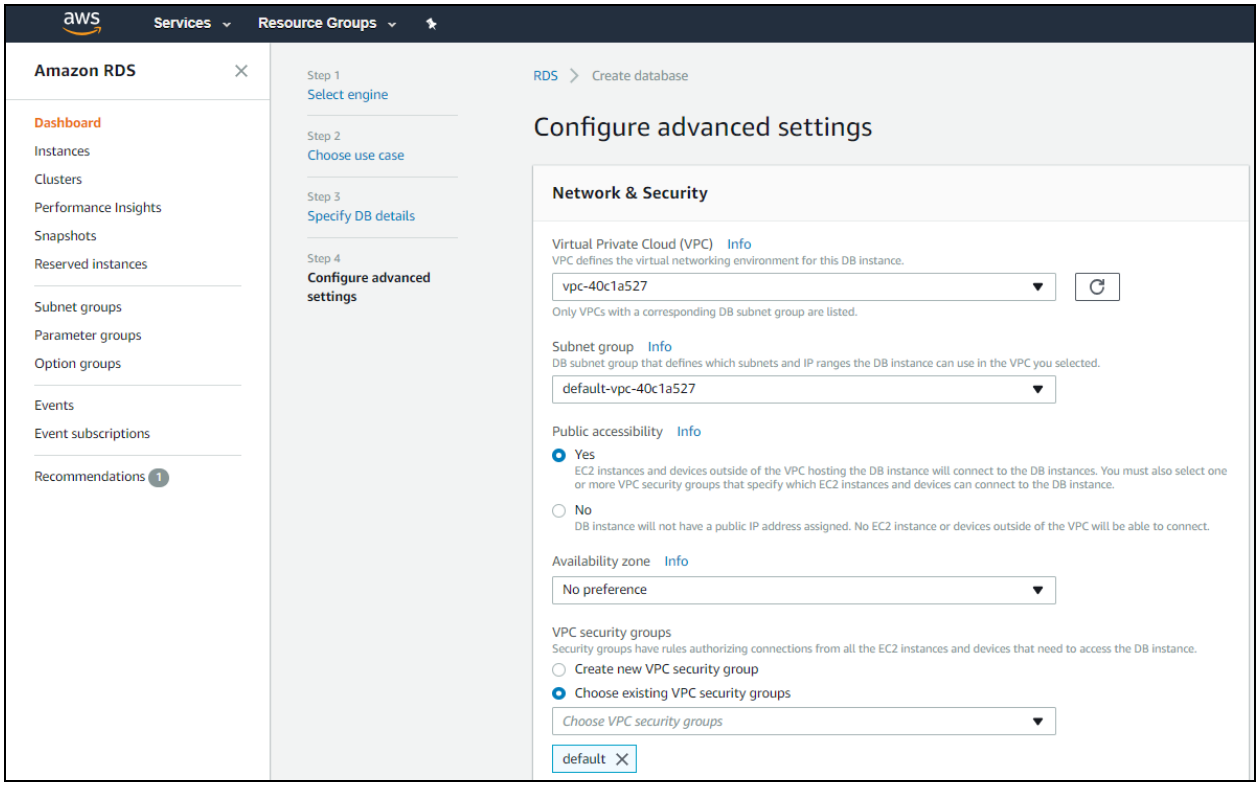
Step 6 - Follow the image below for the instance specifications



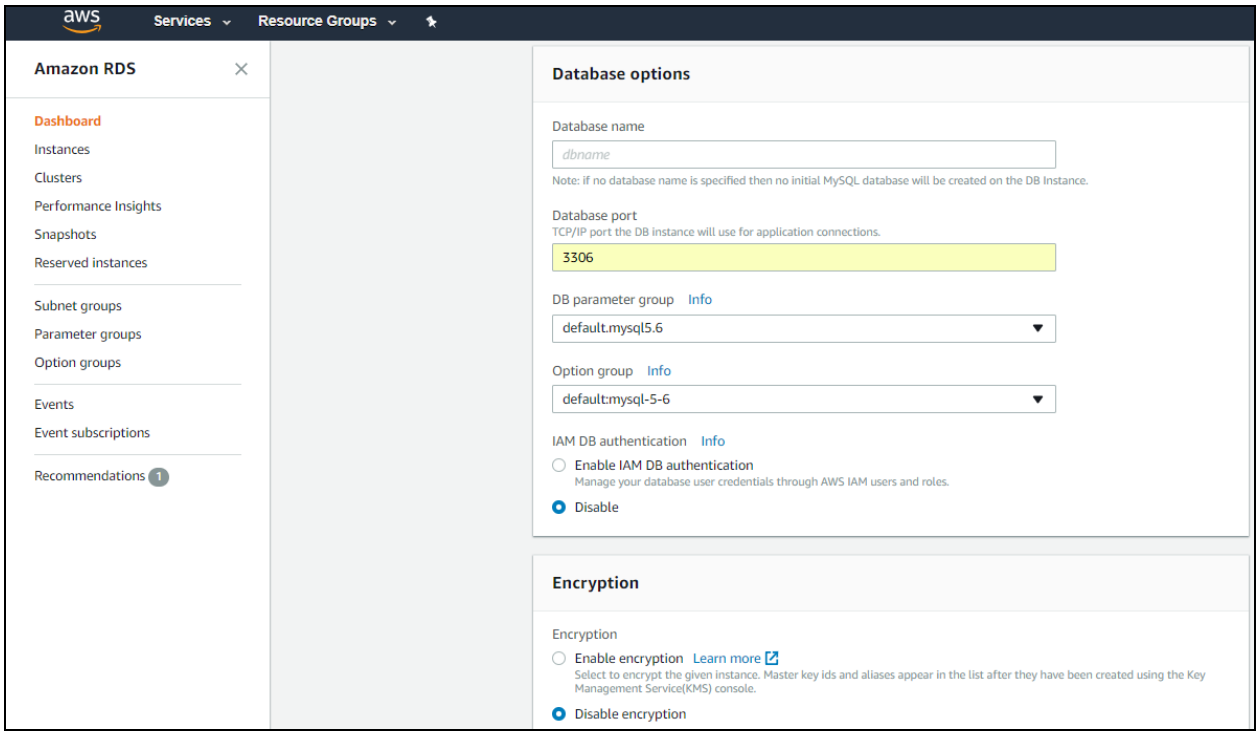
Under settings, enter your DB instance identifier, Master username, password and confirm password then click “next” button

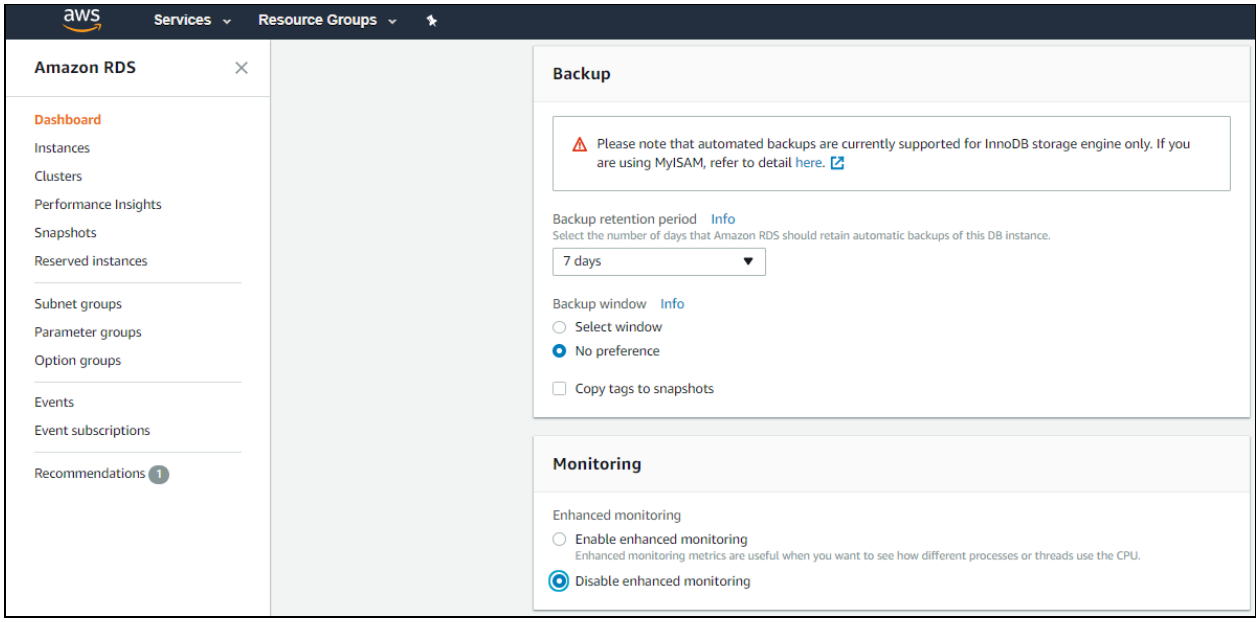


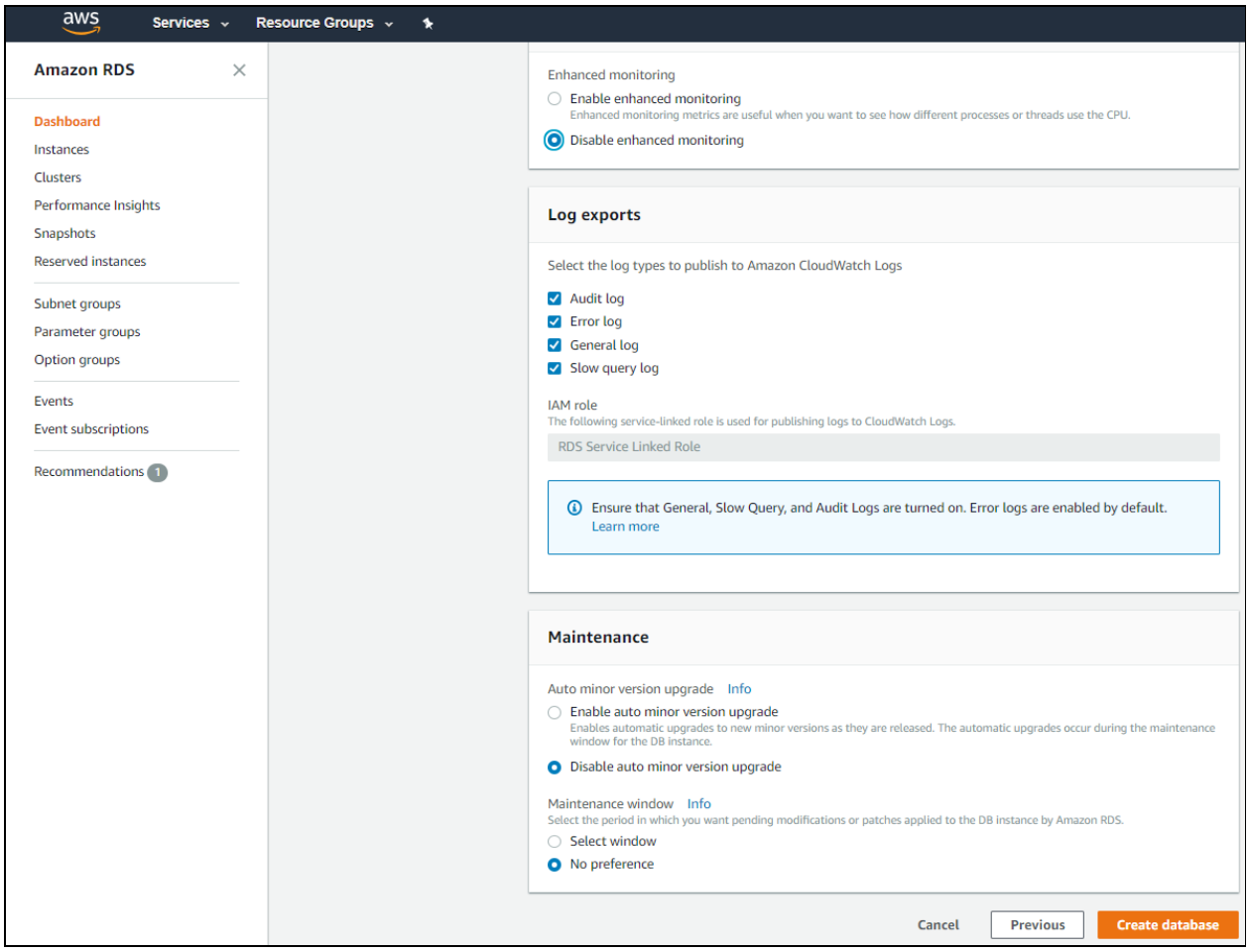
Step 7 - Follow the image below for advance settings and then click “create database” button



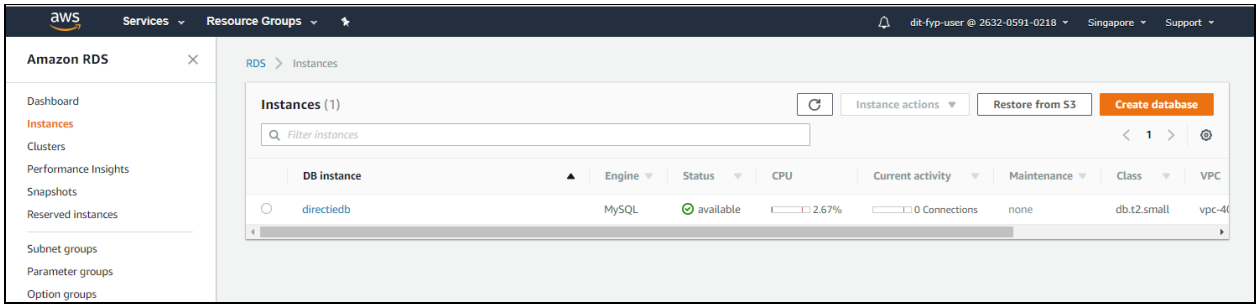
For VPC and subnet group select default and for VPC Security Group, create new or use existing one to allow connection from the IP address of your device to the database.




aws
Services
Resource Groups
Amazon RDS
Dashboard
Instances
Clusters
Performance Insights
Snapshots
Reserved instances
Subnet groups
Parameter groups
Option groups
Events
Event subscriptions
Recommendations 1
Backup
Please note that automated backups are currently supported for InnoDB storage engine only. If you are using MyISAM, refer to detail here.
Backup retention period Info
Select the number of days that Amazon RDS should retain automatic backups of this DB instance.
7 days
Backup window Info
Select window
No preference
Copy tags to snapshots
Monitoring
Enhanced monitoring
Enable enhanced monitoring
Enhanced monitoring metrics are useful when you want to see how different processes or threads use the CPU.
Disable enhanced monitoring

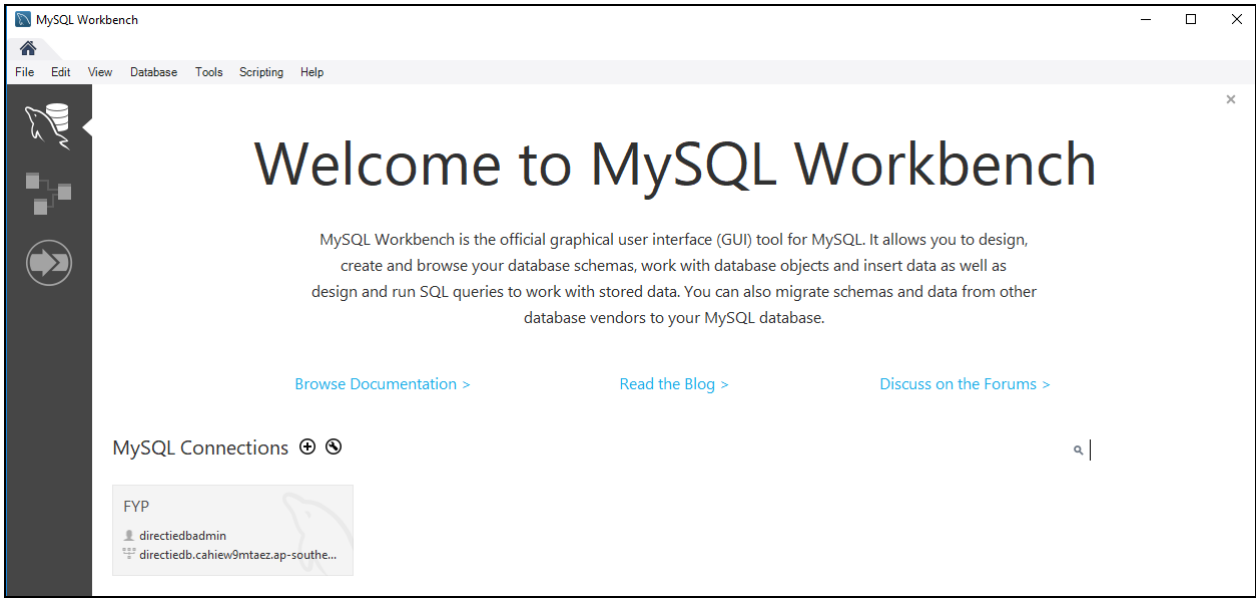

aws
Services
Resource Groups
Amazon RDS
Dashboard
Instances
Clusters
Performance Insights
Snapshots
Reserved instances
Subnet groups
Parameter groups
Option groups
Events
Event subscriptions
Recommendations 1
Enhanced monitoring
Enable enhanced monitoring
Enhanced monitoring metrics are useful when you want to see how different processes or threads use the CPU.
Disable enhanced monitoring
Log exports
Select the log types to publish to Amazon CloudWatch Logs
Audit log
Error log
General log
Slow query log
IAM role
The following service-linked role is used for publishing logs to CloudWatch Logs.
RDS Service Linked Role
Ensure that General, Slow Query, and Audit Logs are turned on. Error logs are enabled by default.
Learn more
Maintenance
Auto minor version upgrade Info
Enable auto minor version upgrade
Enables automatic upgrades to new minor versions as they are released. The automatic upgrades occur during the maintenance window for the DB instance.
Disable auto minor version upgrade
Maintenance window Info
Select the period in which you want pending modifications or patches applied to the DB instance by Amazon RDS.
Select window
No preference
Cancel
Previous
Create database

Step 8 - Go back to instances to view the database you have just created and click on the instance you created

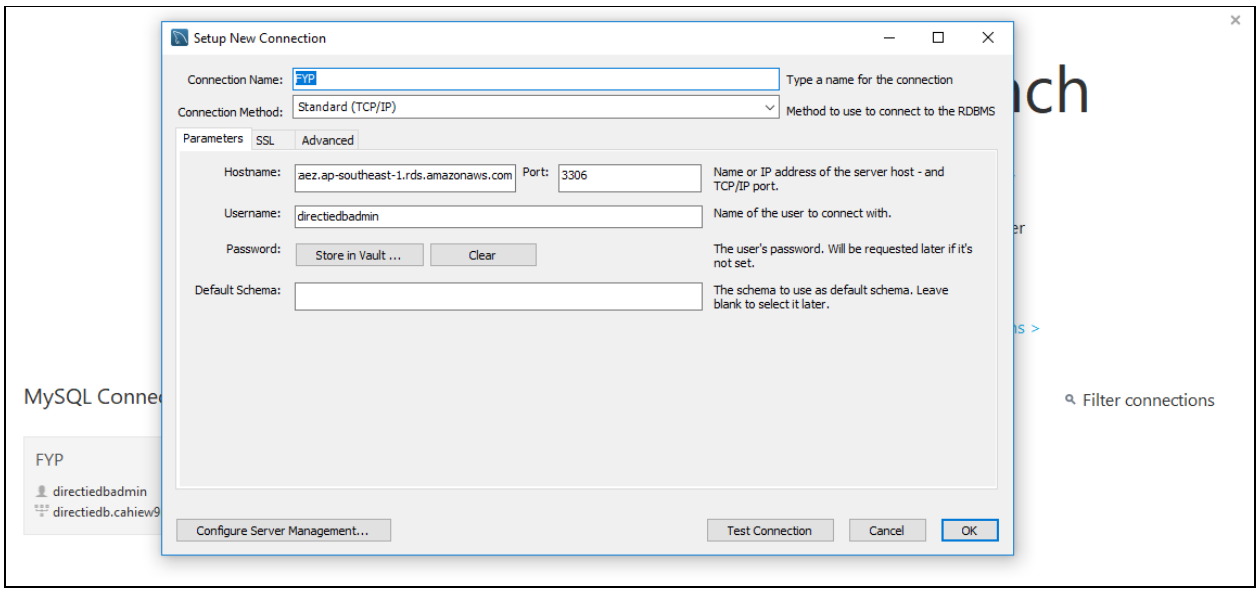


Step 9 - Copy the Endpoint from your AWS RDS instance

Step 10 - Open MySQL Workbench and create a new MySQL Connections by clicking on the plus button

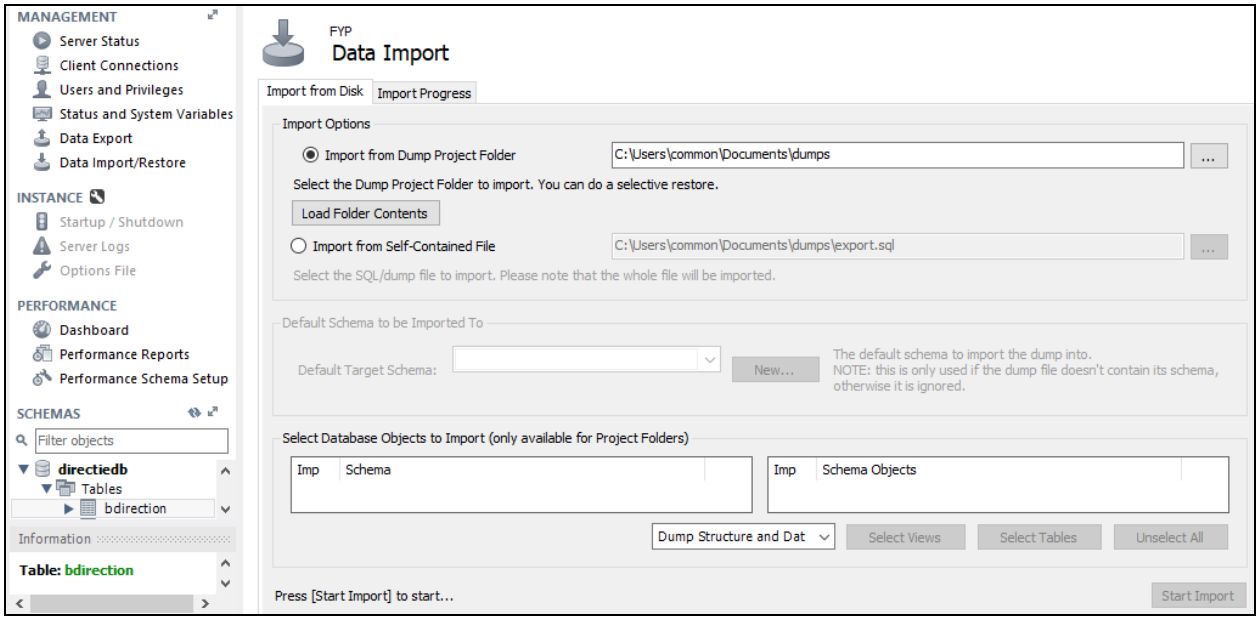


Step 11 - Enter your Connection Name and paste the Endpoint in the Hostname. For username and password enter the one you set in step 6 and click “OK” button

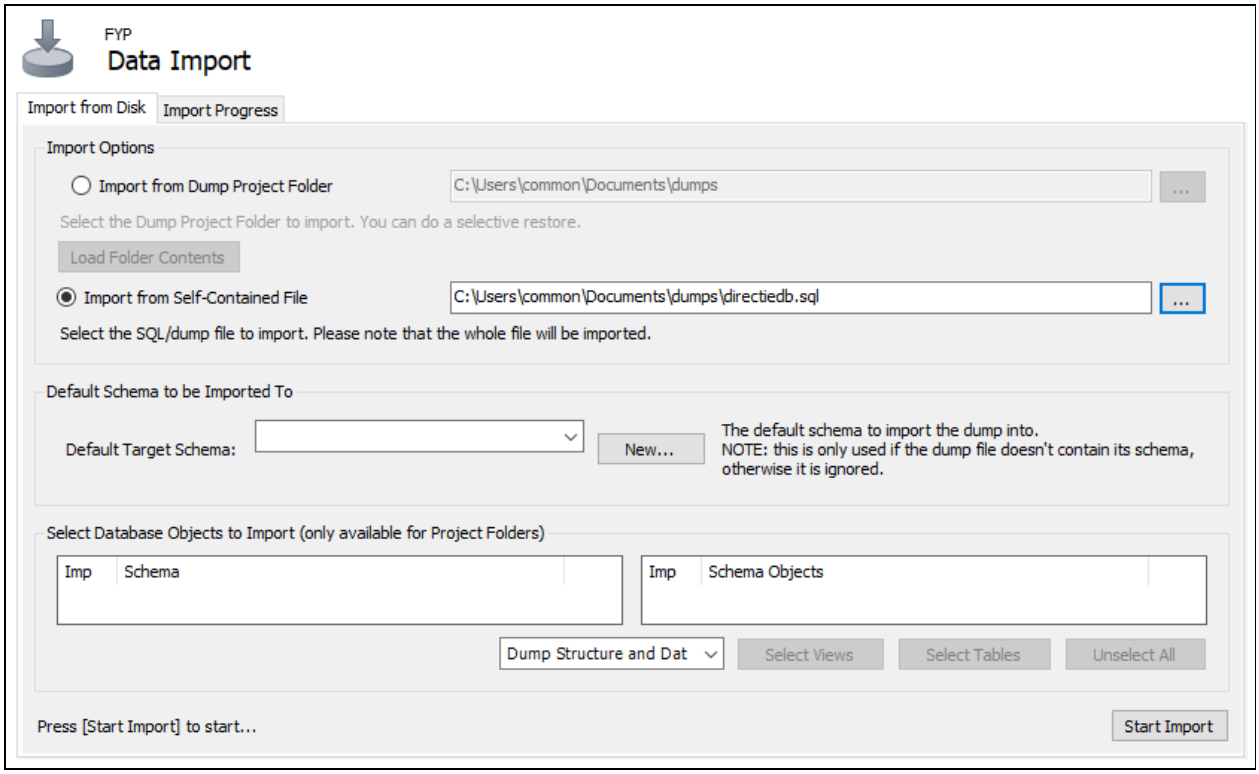


Step 12 - Download the sql file in this link : https://drive.google.com/open?id=1sZ5AwGbKKRIsilOdUPmEm8J6Y8L6oKBt

Step 13 - In your sql, click “Data Import/Restore” under “Management”

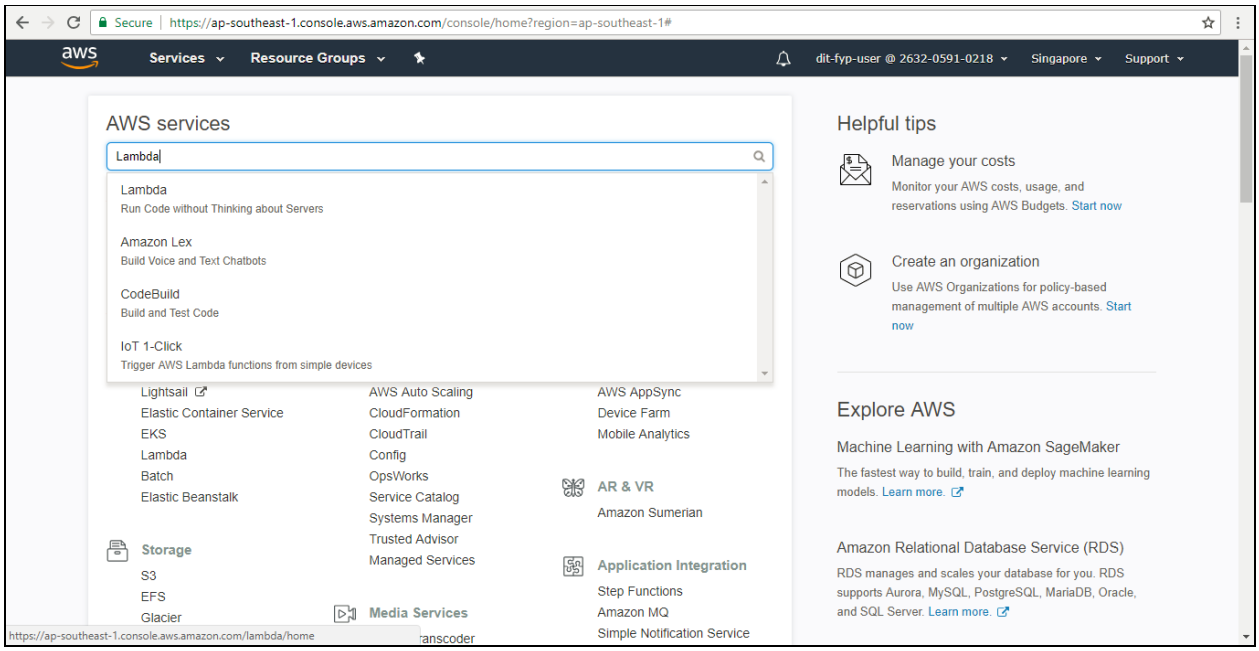


Step 14 - Select “Import from Self-Contained File” and choose the file downloaded in step 12 and click “Start Import” button

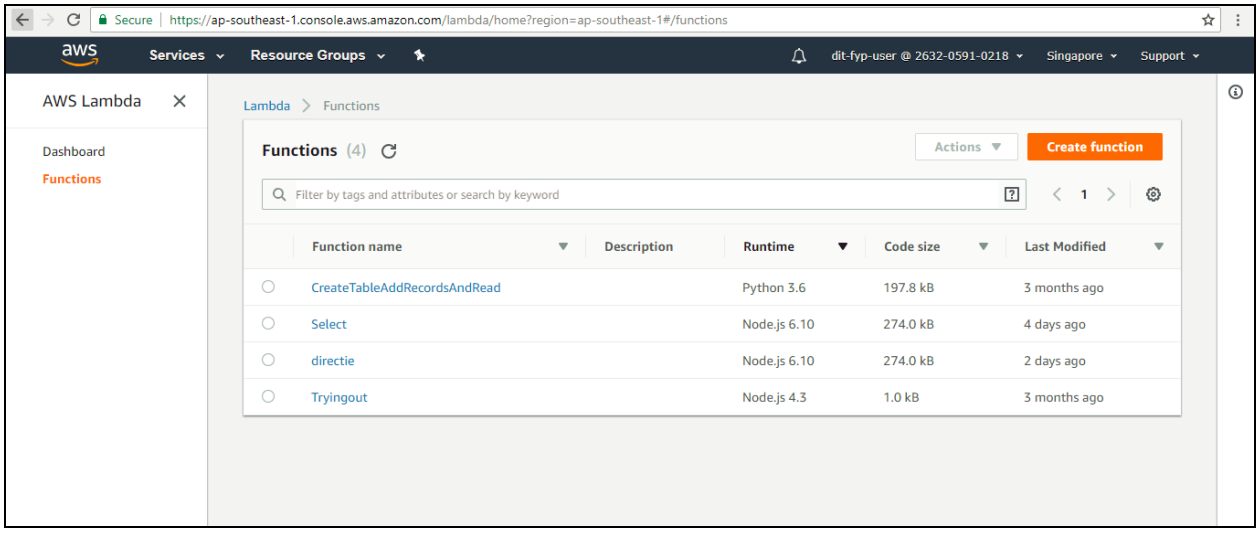


1.1 Setting up AWS Lambda

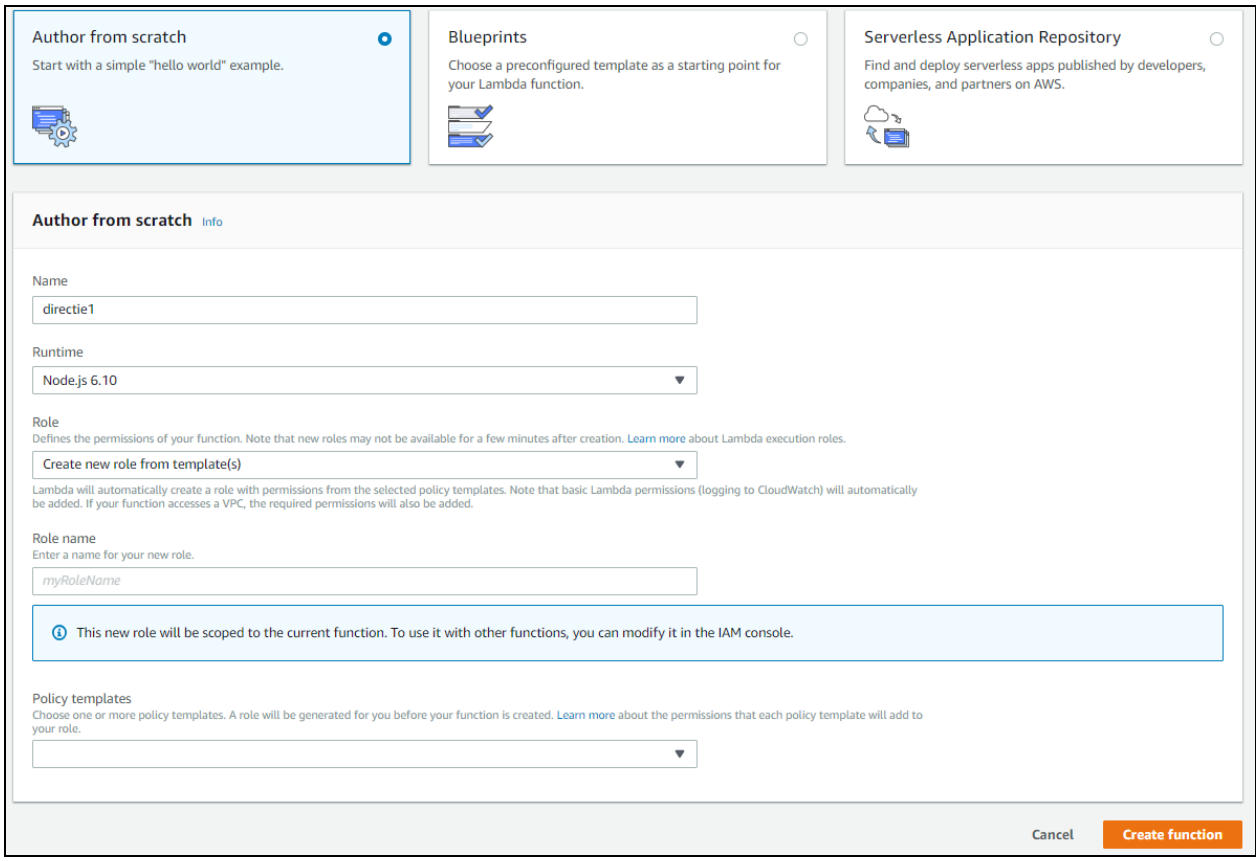
Step 1 - Click on Lambda under the “Compute” category or search “Lambda”



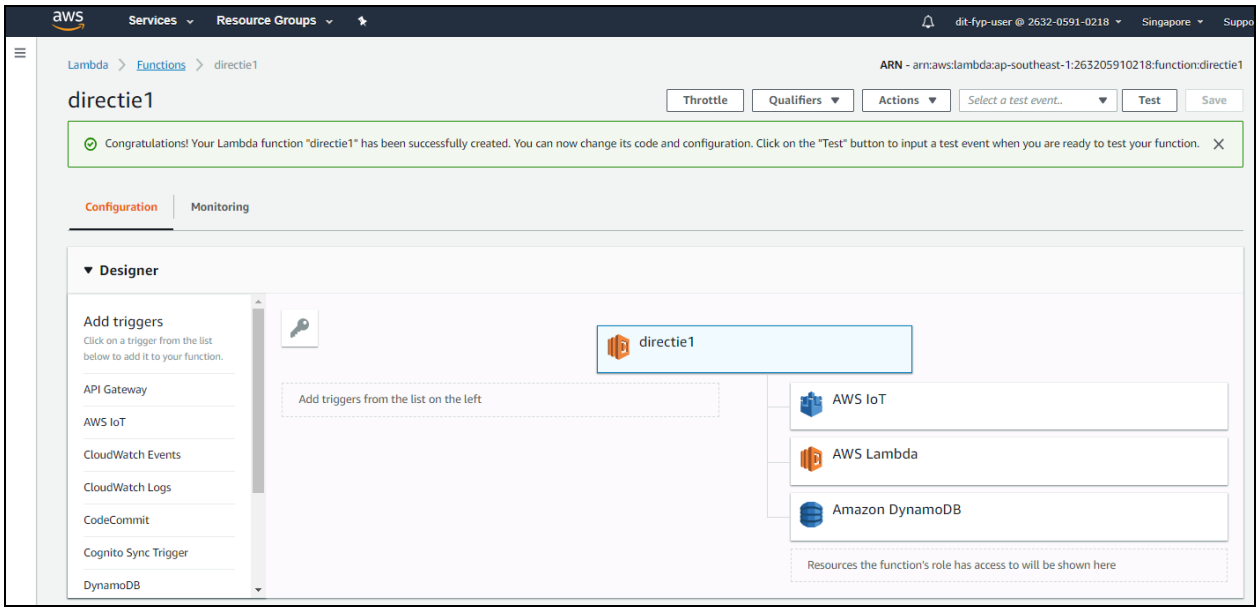
Step 2 - Click on “Create function” button



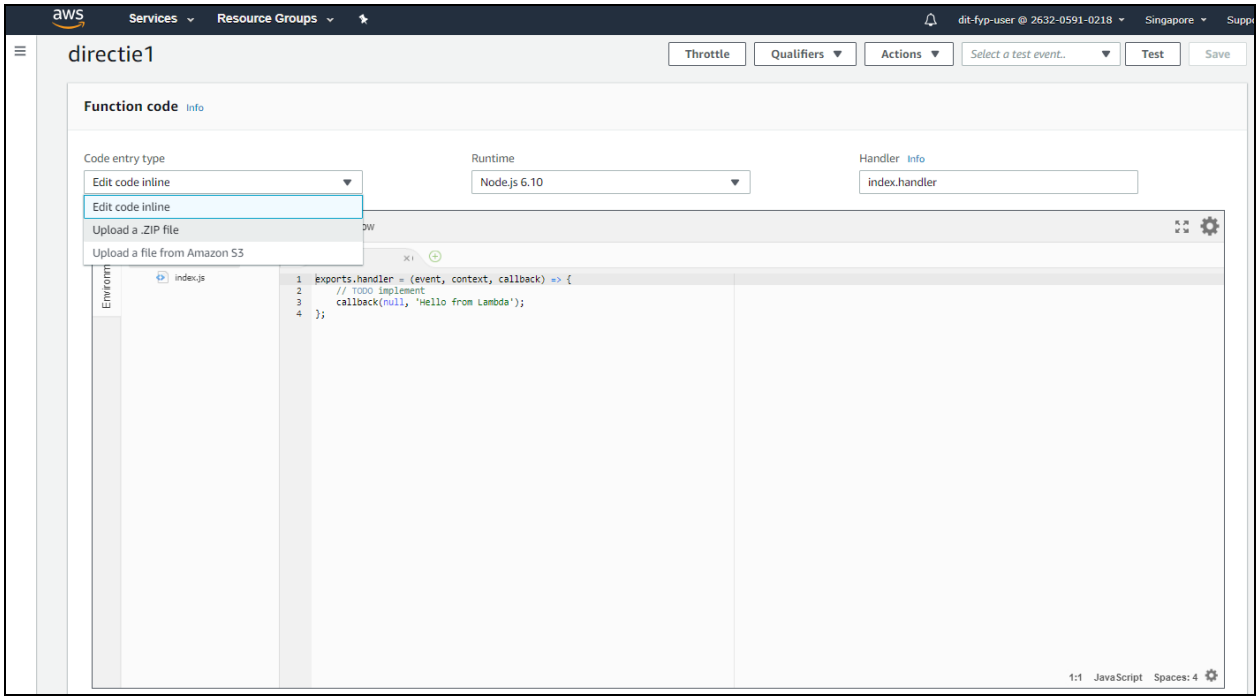
Step 3 - Select “Author from scratch” and enter your own Lambda function name. For runtime select “Node.js 6.10”. Create a new role or use an existing that allows full access to lambda.



Step 4 - You should be redirected to your new Lambda function page.



Step 5 - Scroll down to the “Function code”. Click on “Upload a .ZIP file” under “Code entry type”



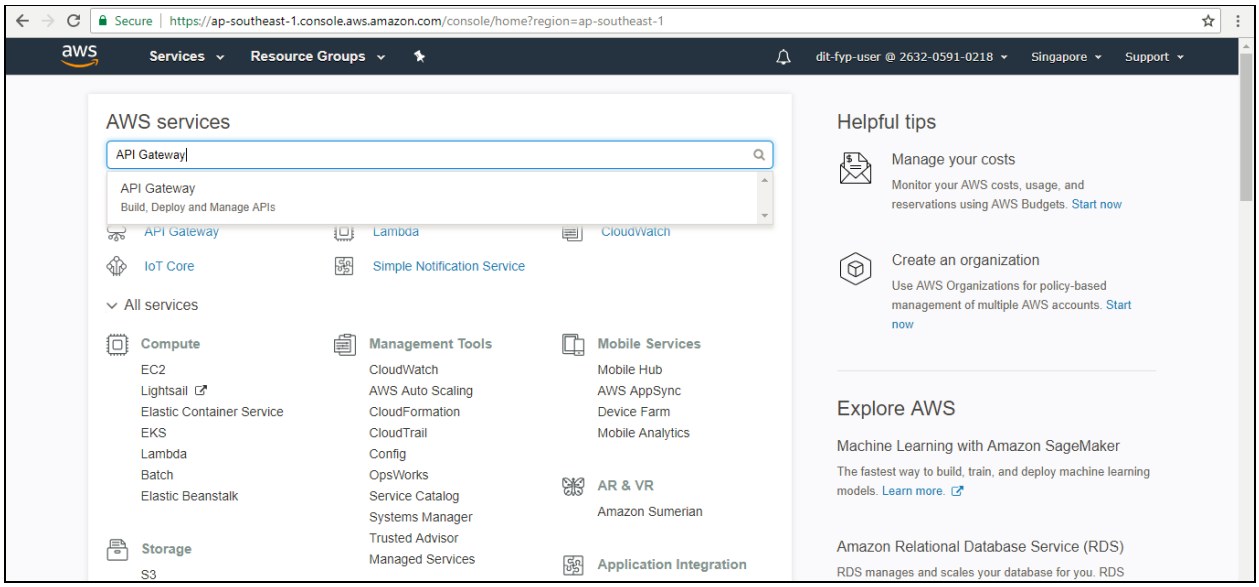
Step 6 - Download the .ZIP file in this link : https://drive.google.com/open?id=1gLdG_dkLwit0VO0uhbPFpuYv0frKIgTD

Step 7 - Change Handler to “directie.handler”. Click on “Upload” button and select the .ZIP file downloaded from Step 6.

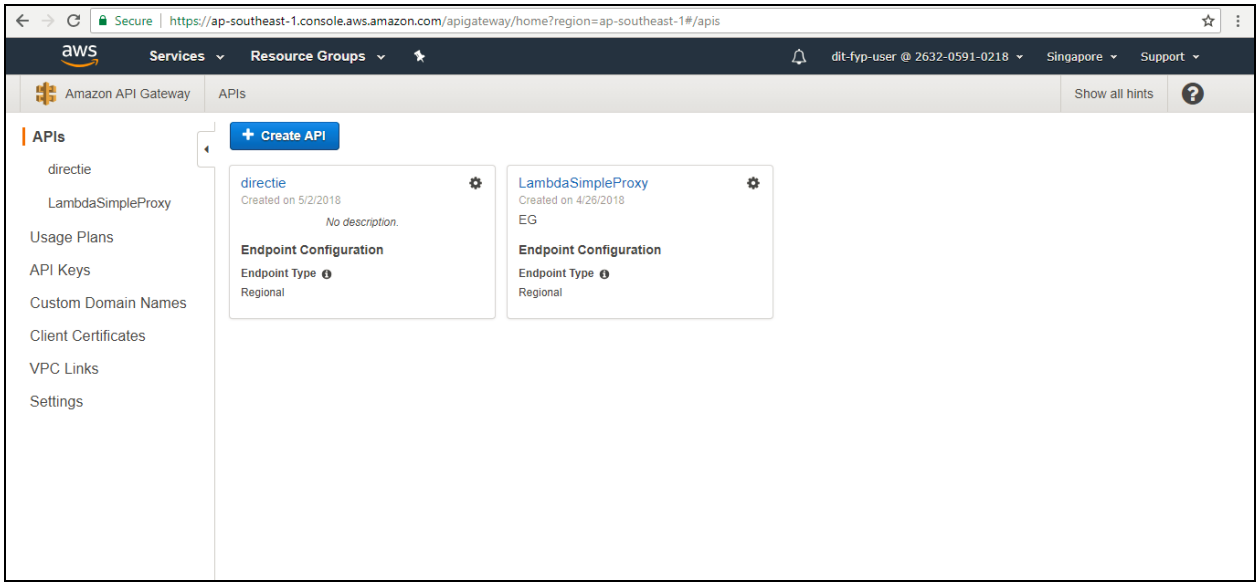
Step 8 - Click on the “Save” Button

1.2 Setting up AWS API Gateway

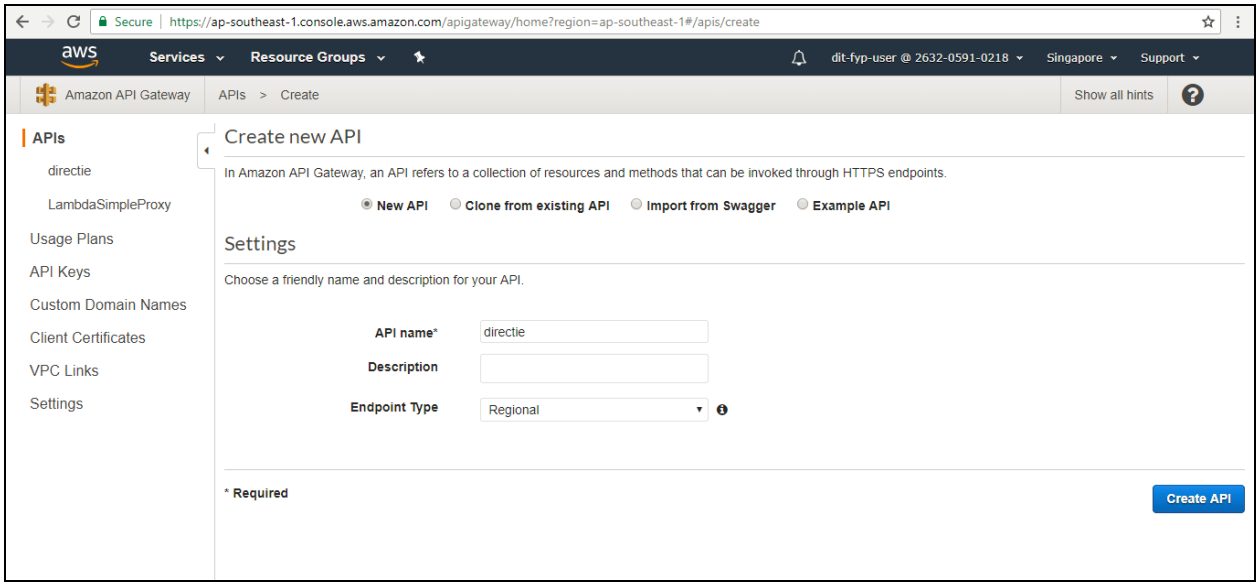
Step 1 - Click on API Gateway under the “Networking & Content Delivery” category or search “API Gateway”



Step 2 - Click on “Create API”

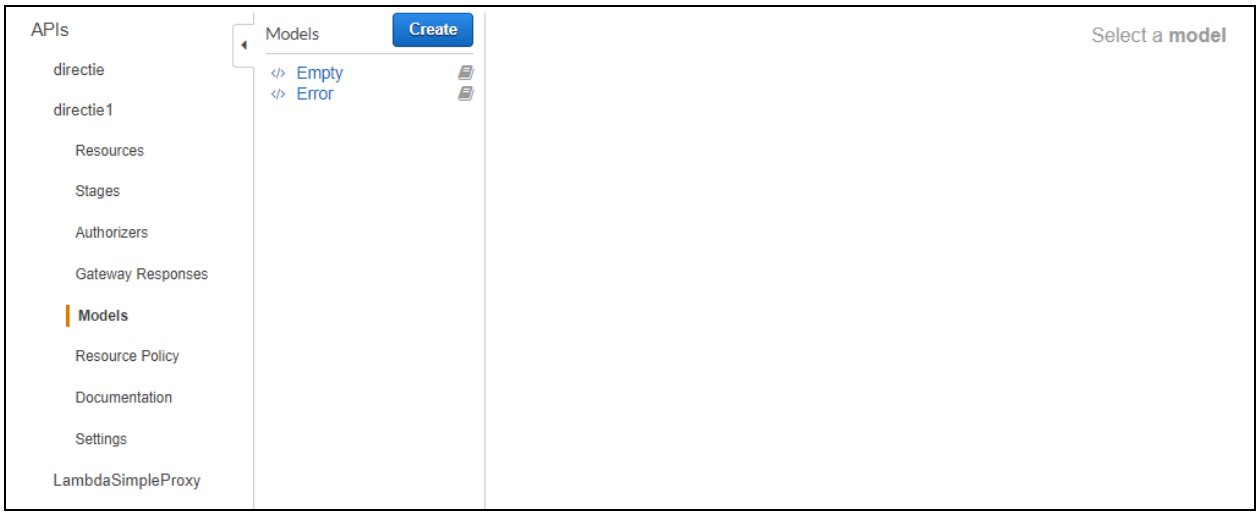


Step 3 - Enter your own API name and description and click “Create API”

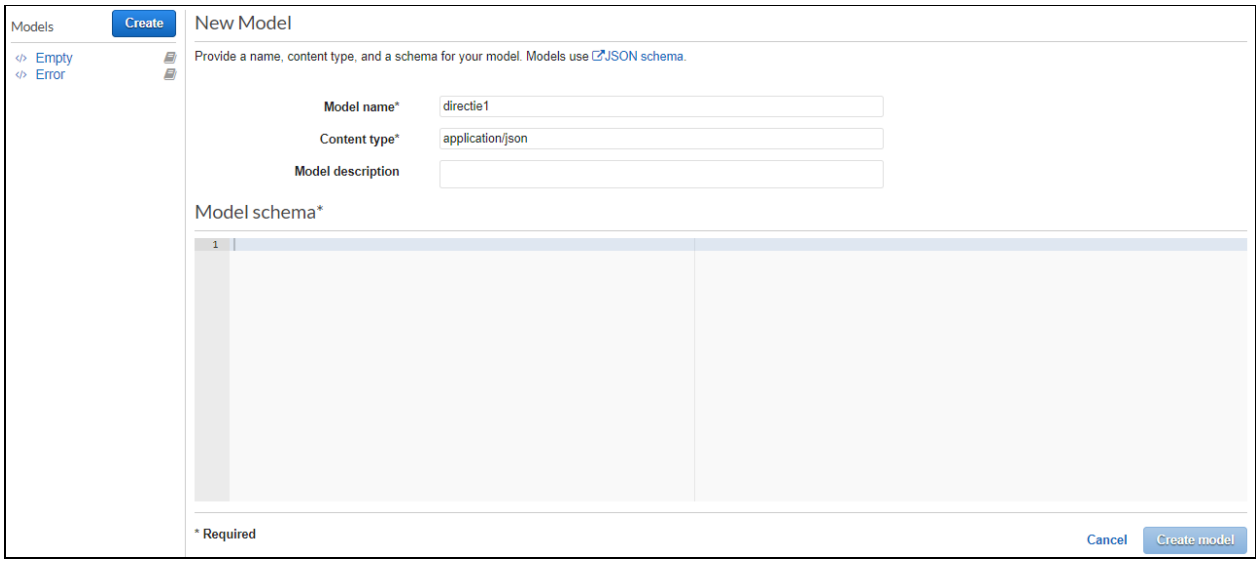


Step 4 - Download the text file in this link :
https://drive.google.com/open?id=13I7IrRuHFu68HK0dGiBTXgYWrhS_DvCZ

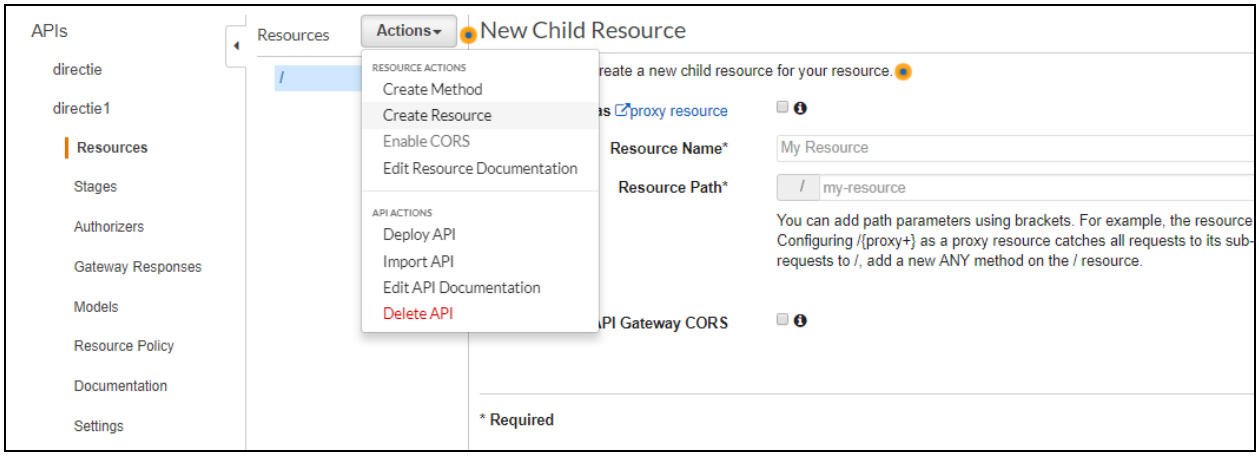
Step 5 - Click on “Models” on the left navigation bar and click “Create” button



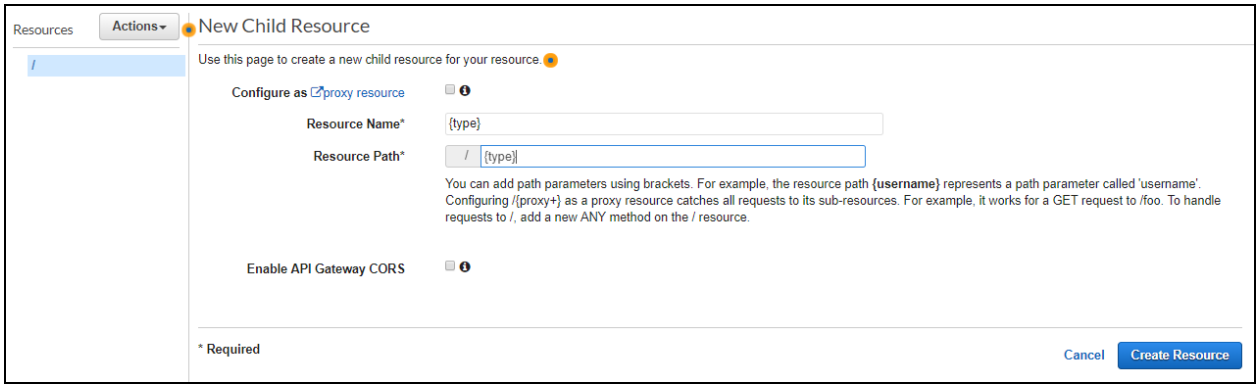
Step 6 - Enter your Model name and description. For content type enter “application/json”. Open the text file downloaded in step 4, copy and paste everything into the Model schema and click “Create Model” button.



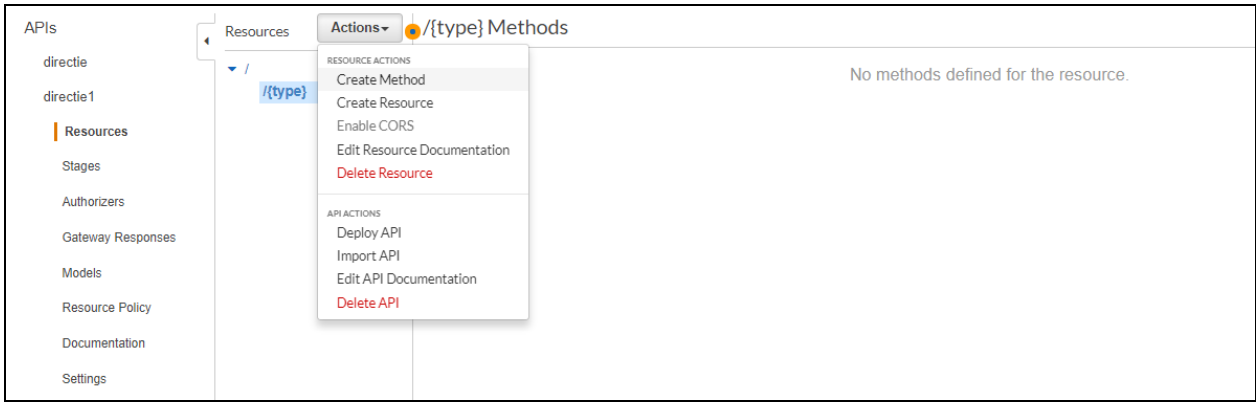
Step 7 - Go back to the Resources page and click “Create Method” under “Actions”



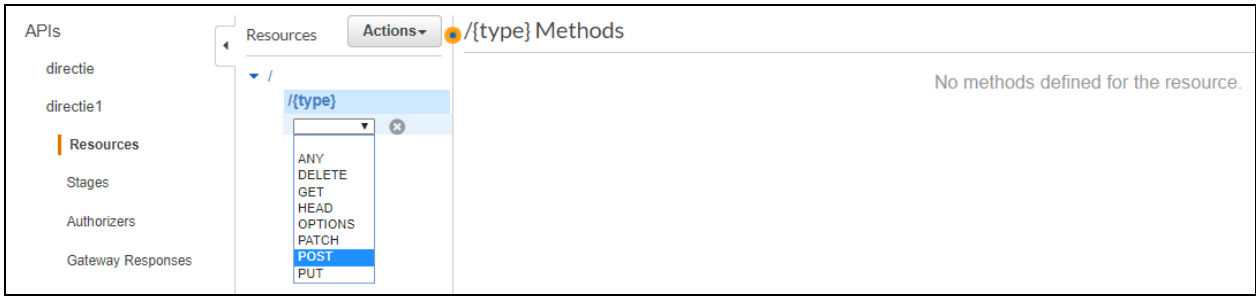
Step 8 - Follow the image below for the configurations of the new Resource and click “Create Resource” button



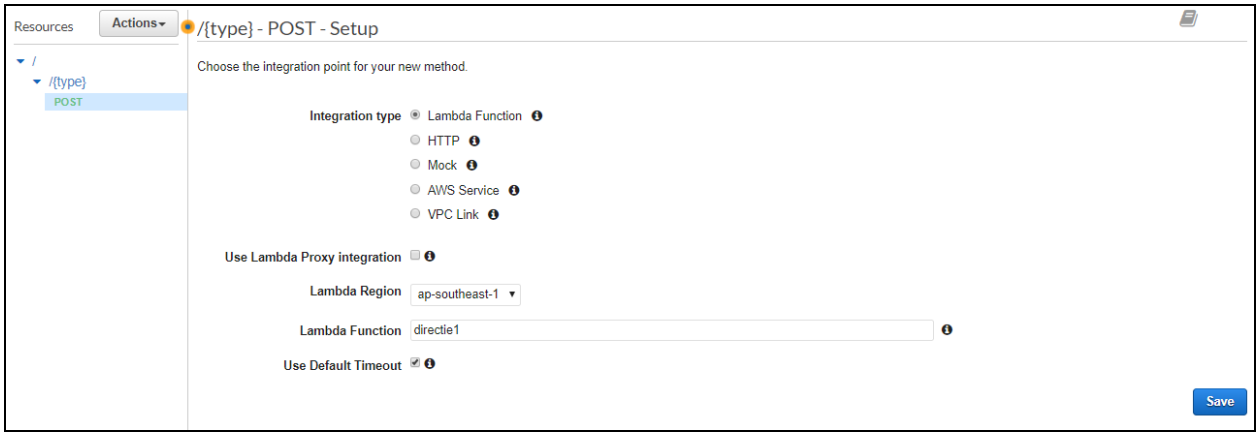
Step 9 - Under the {type} Resource, click on “Create Method” under “Actions”



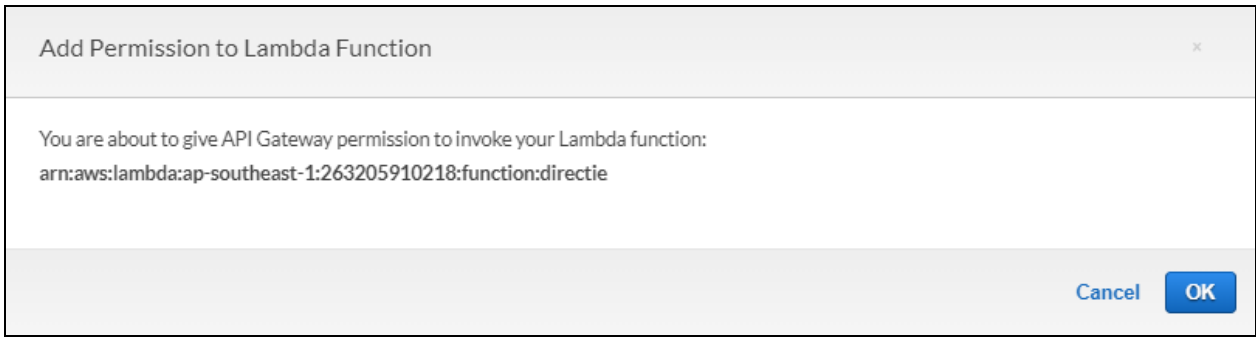
Step 10 - You can create GET, POST, PUT and DELETE methods with the same setup but we are going to only use POST method.



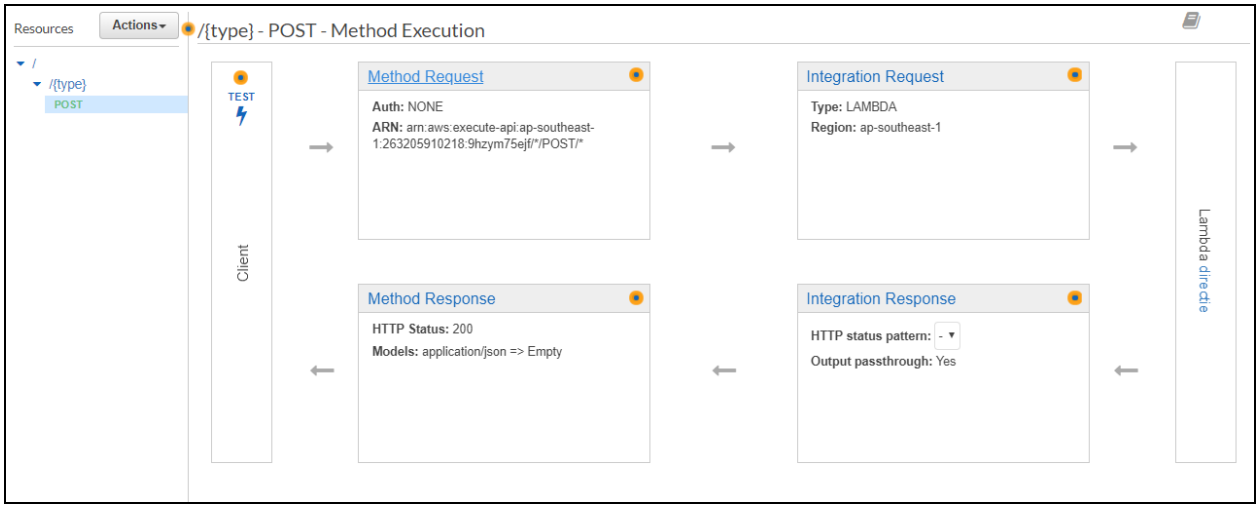
Step 11 - Follow the image below for the POST method Setup, enter the Lambda function name you created in Part 3.0 in this tutorial and click “Save” button



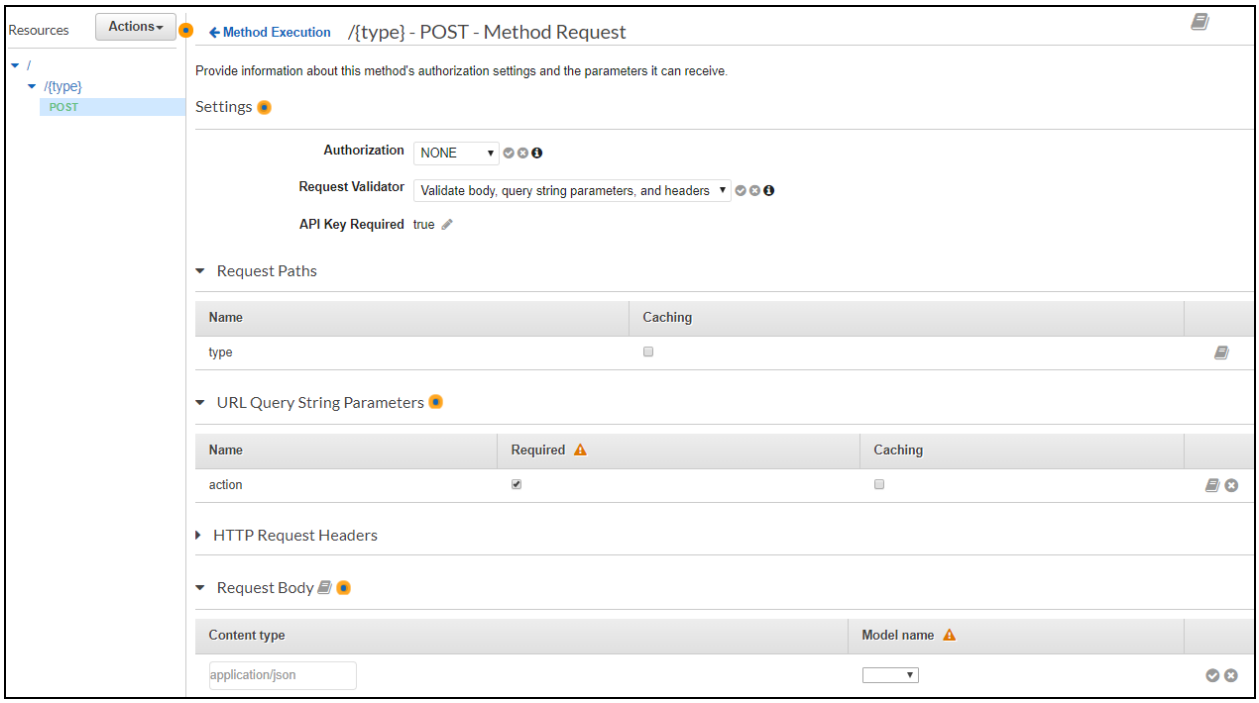
This will pop out after clicking “Save” button, click “OK”.



Step 12 - Click on “Method Request”



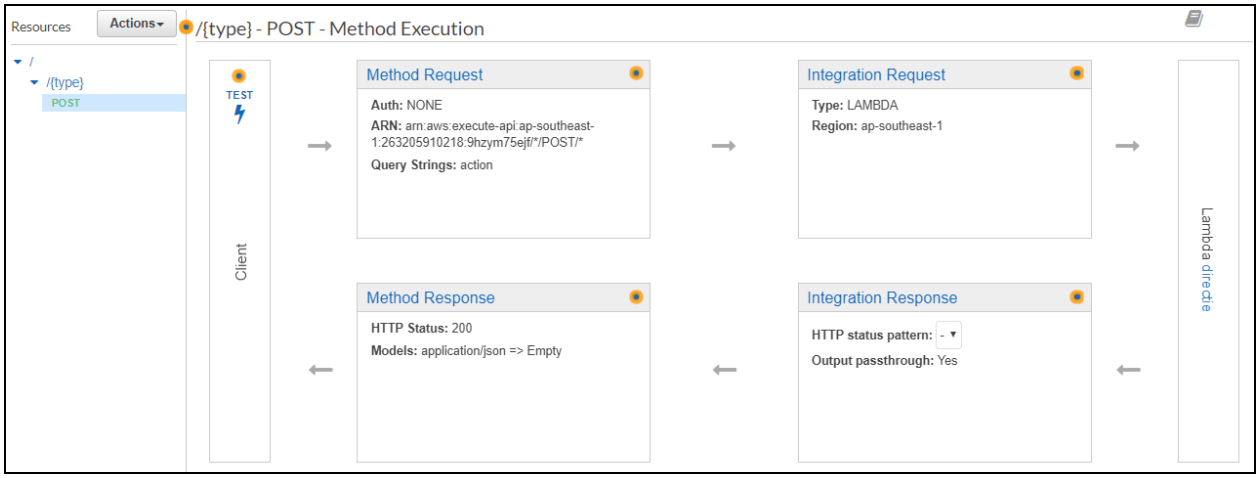
Step 13 - Follow the image below for Method Request. Under Request Body, enter Content type “application/json” and choose the Model created in step 6 and click “Method Execution”



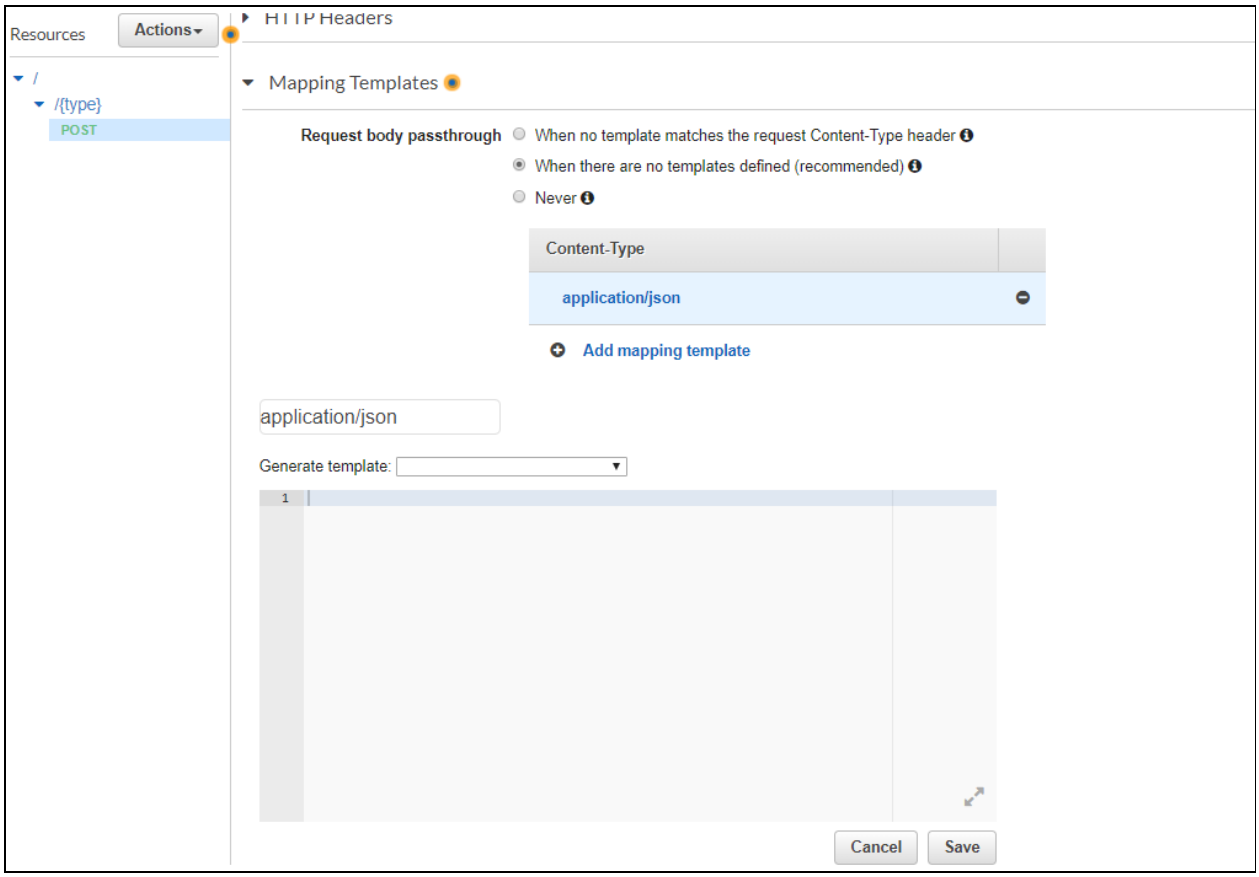
You can choose not to use the API Key, set it to “false” if you do not wish to do so.

Step 14 - Download the text file in this link :
https://drive.google.com/open?id=15L_7briG45qa12sgUGSpmOiuZVfajLHJ

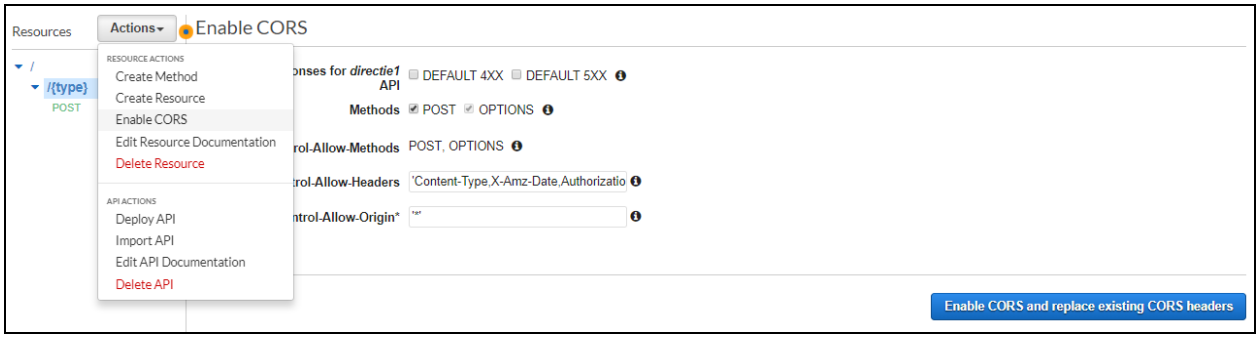
Step 15 - Click on “Integration Request”



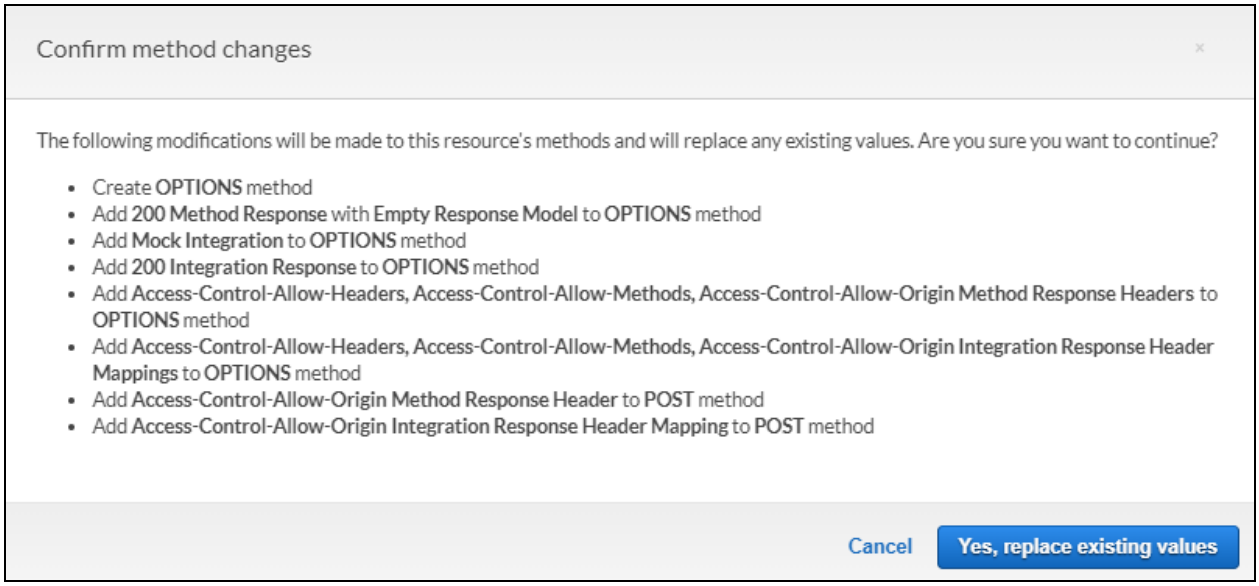
Step 16 - Follow the image below for Mapping Templates in Integration Request, open the text file downloaded in step 14, copy and paste everything into the Mapping Template, click “Save” button and click “Method execution”



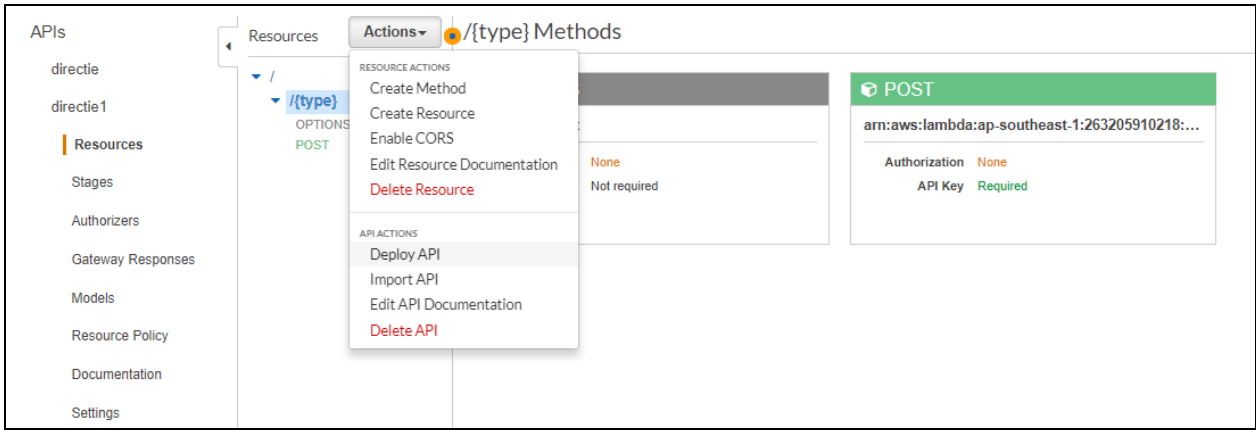
Step 17 - Click on “Enable CORS” under “Actions” and click “Enable CORS and replace existing CORS headers” button

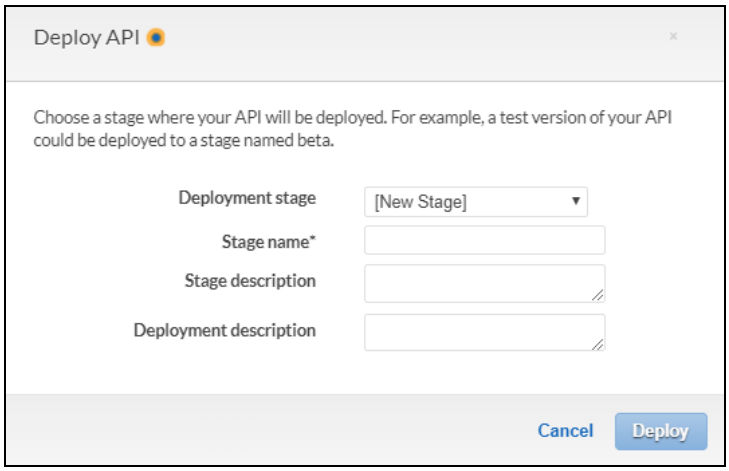


This will pop out after clicking “Enable CORS and replace existing CORS headers” button, click “Yes, replace existing values”.



Step 18 - Click on “Deploy API” under “Actions”

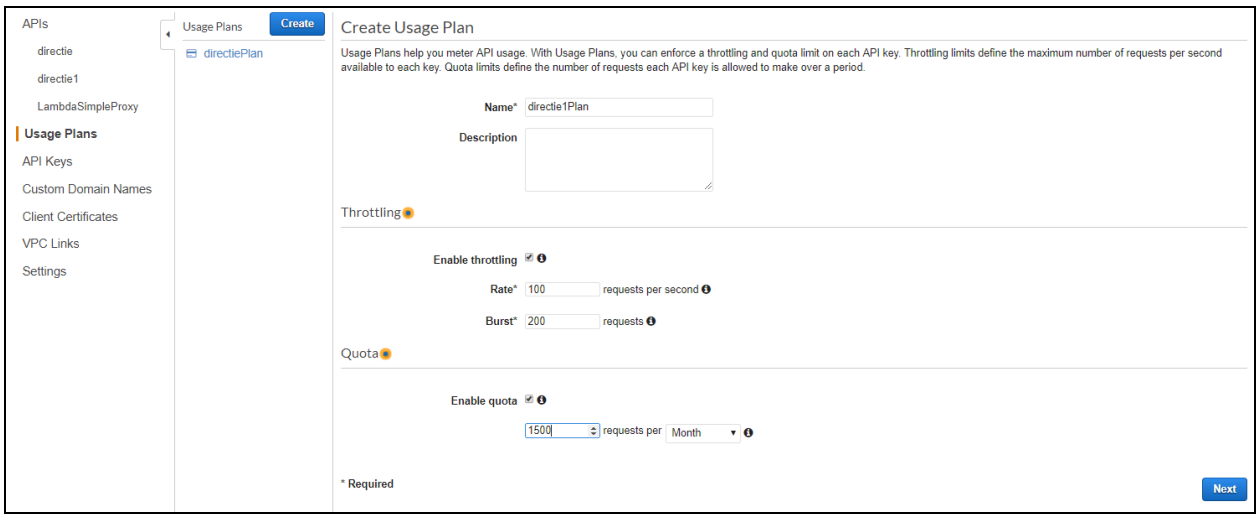




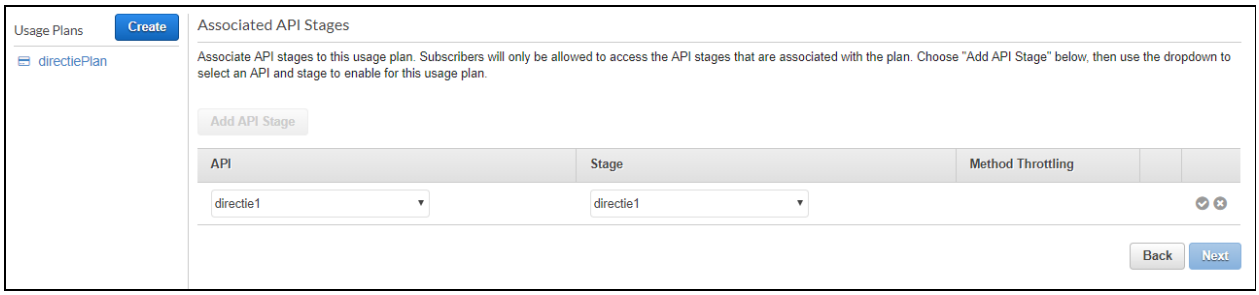
Enter your Stage name and descriptions for the API and click “Deploy” button

***If you chose to use API Key in step 13, continue with the tutorial below.**

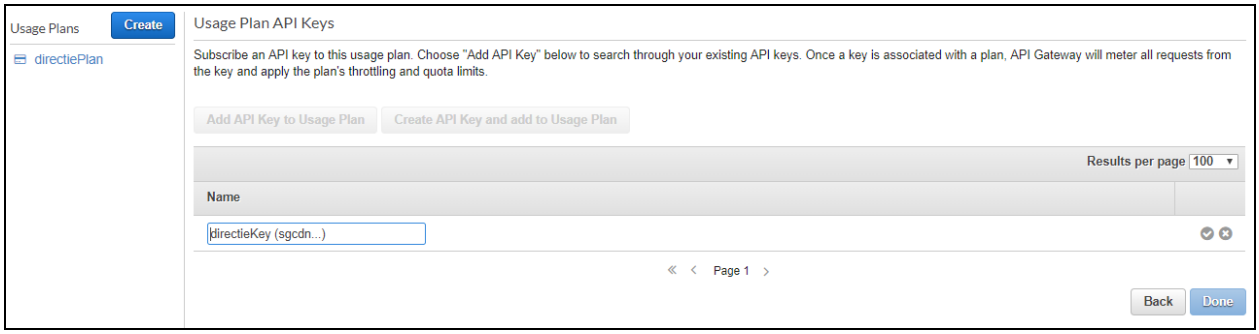
Step 19 - Go to Usage Plans and click “Create” button, enter your plan name and description and follow the image below for other configurations and click “Next”



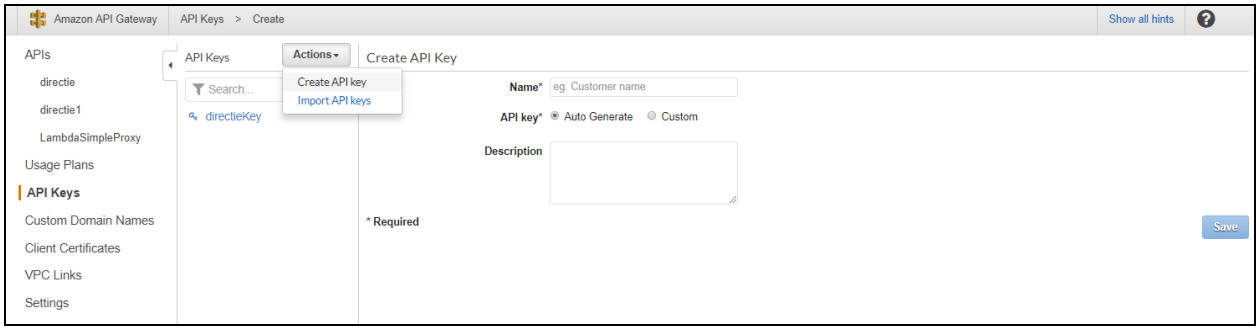
Step 20 - Add API Stage to the one in Step 18 and click “Next” button



Step 21 - Add an API Key to the Usage Plan and click “Done” button



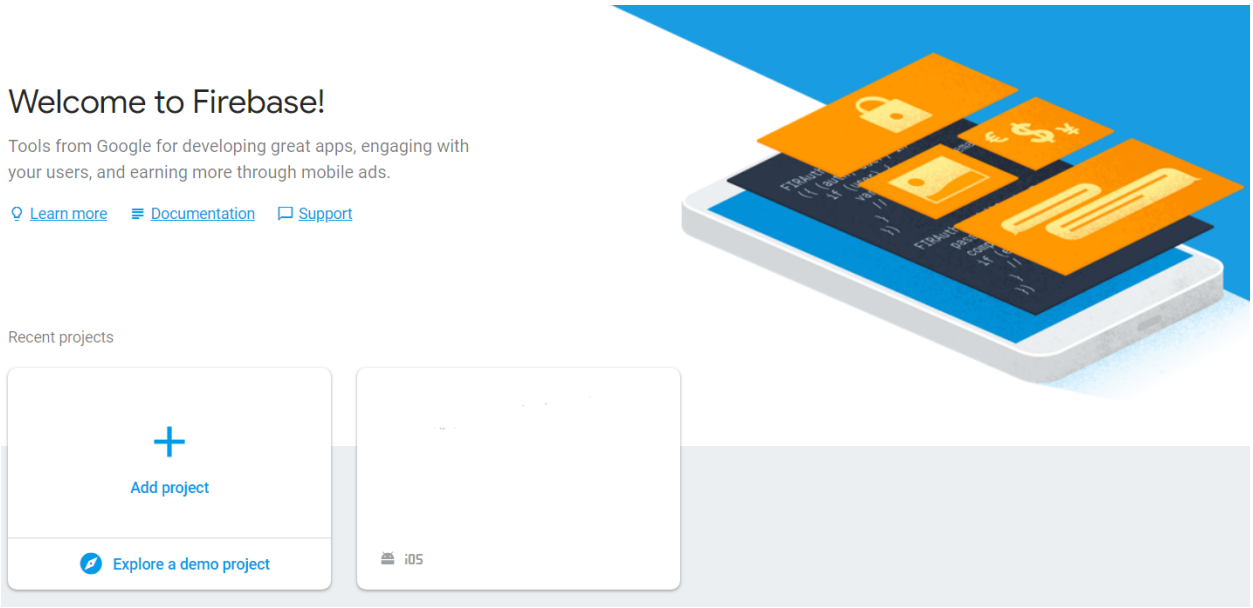
You can create an API Key by going to the “API Keys” in the left navigation bar and click “Create API key” under “Actions”

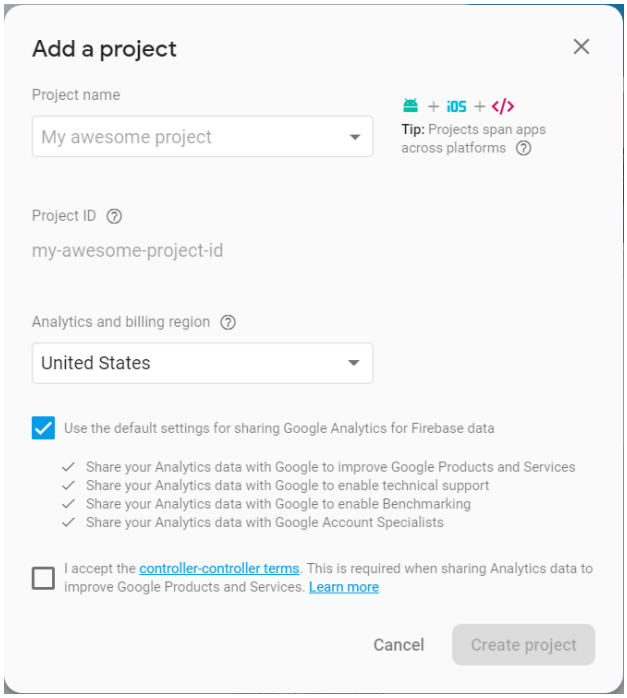


2.0 Setting up Firebase

Before setting up Firebase, you must sign up for a Google Account if you do not have one.

Step 1 - Go to https://console.firebase.google.com/ and click on Add project under Recent projects.



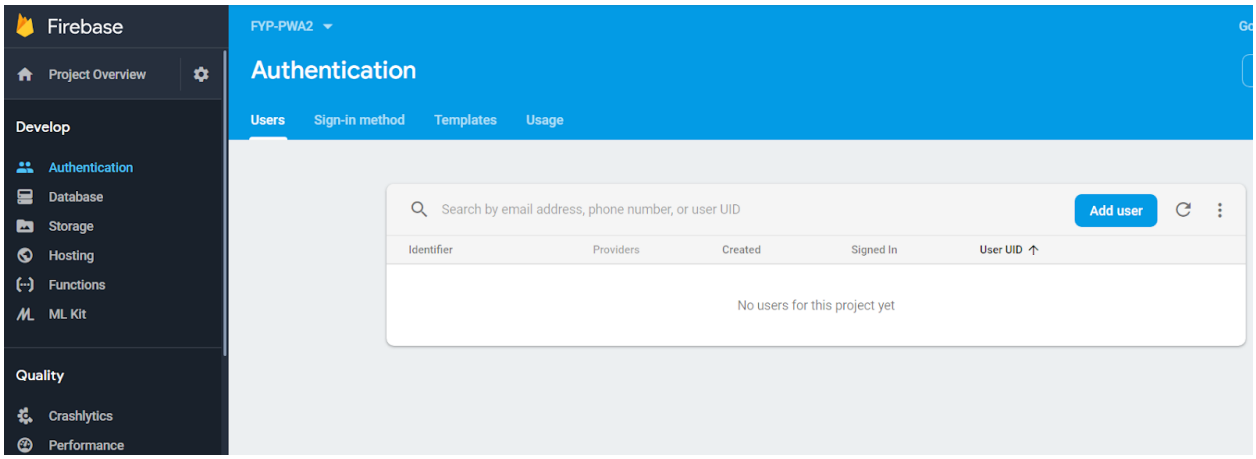


Step 2 -
Give the project a name, for example, “FYP-PWA”. Tick the last checkbox and click on ‘Create project’.

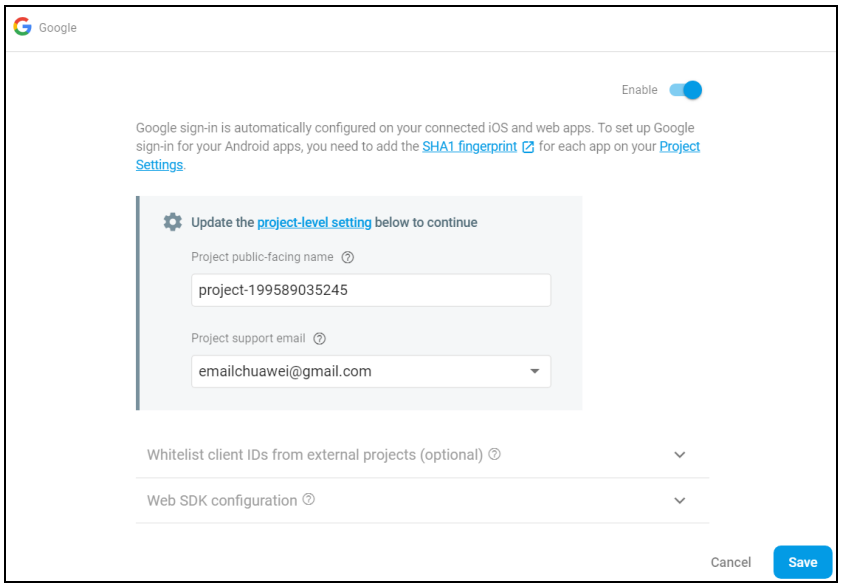
When it shows ‘Your new project is ready’, click on the Continue button.

2.1 Setting up Firebase Authentication

Step 1 - After creating your first project, click on Authentication on the menu. You will see a screen like this.



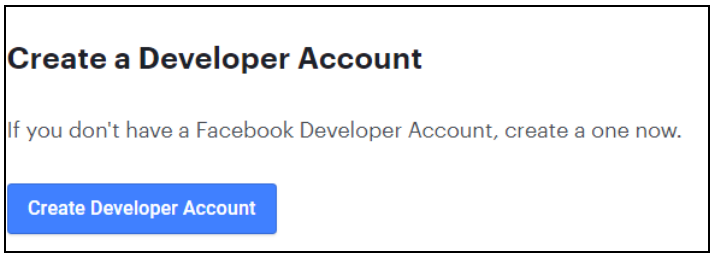
Step 2 - Click on the Sign-in method tab and enable 3 providers: Email/Password, Google and Facebook.



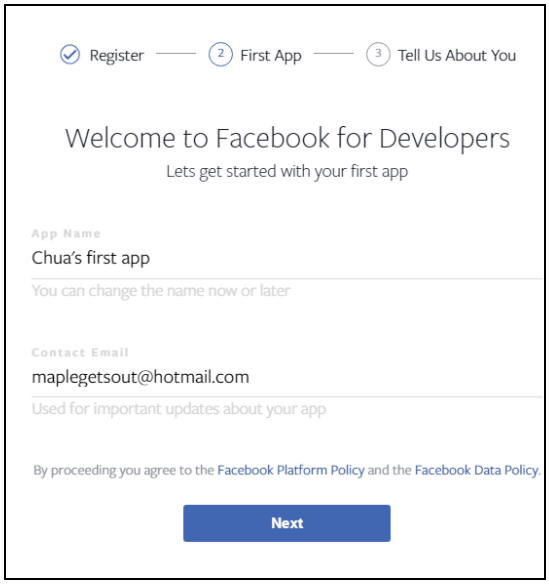
Step 3 - For Google, update the Project support email as the email that you have create for this project.

Step 4 - For Facebook, if you already have an existing App ID and Secret, you may use it and add the OAuth redirect URI to the Facebook app configuration.

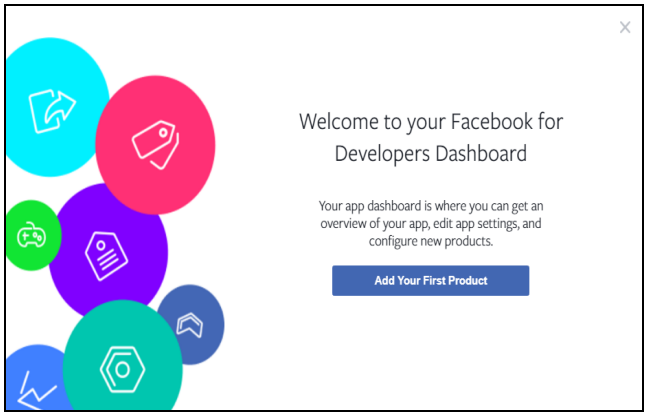
Step 4.1 - If you do not have a existing App ID and Secret, you need to create a Developer Account here: https://developers.facebook.com/docs/apps/register/ (You need to have a Facebook Account first before creating a Developer Account). After signing in, click on Create a Developer Account.

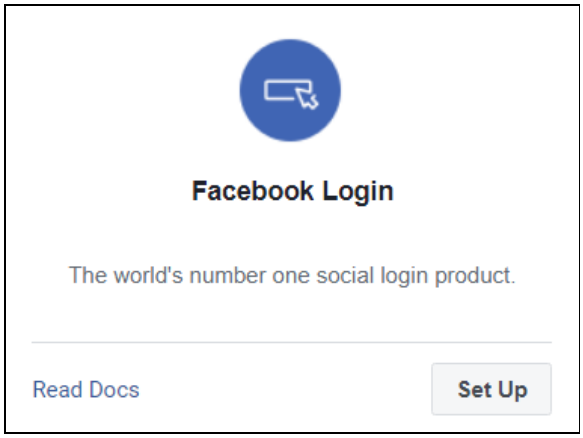


Step 4.2 - A pop-up will appear. Click on next and you may choose to edit the App Name or Contact Email if you want.

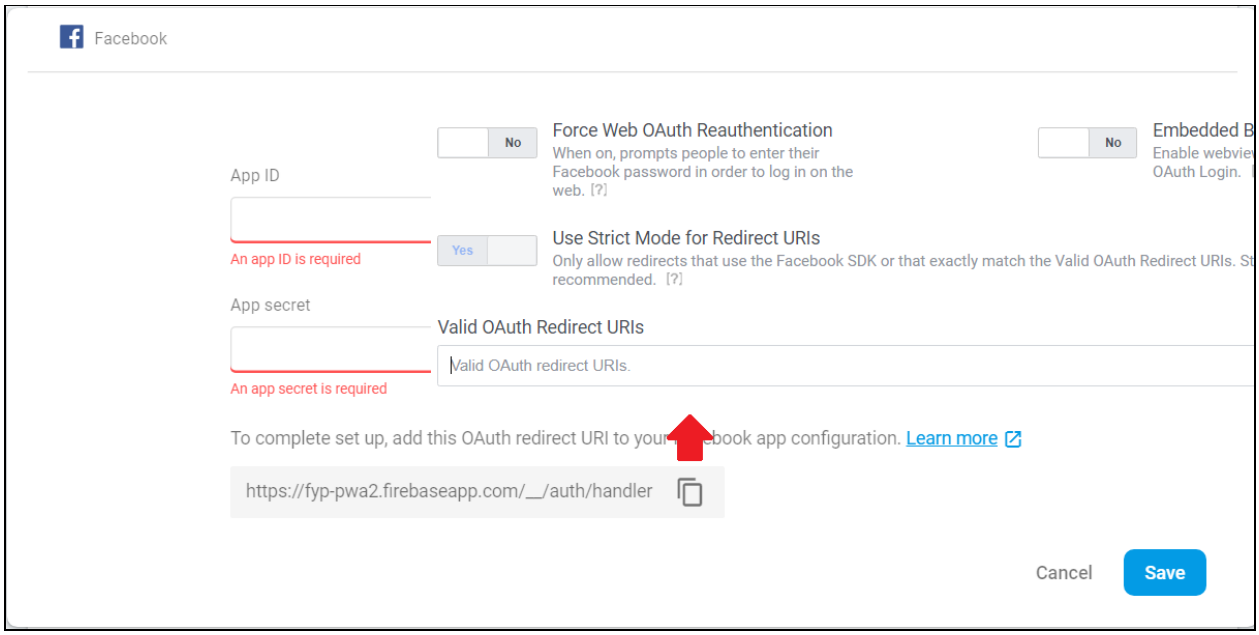


Step 4.3 - Click on Developer and after creating an account, click on Add Your First Product. Then, click on Set Up under Facebook Login

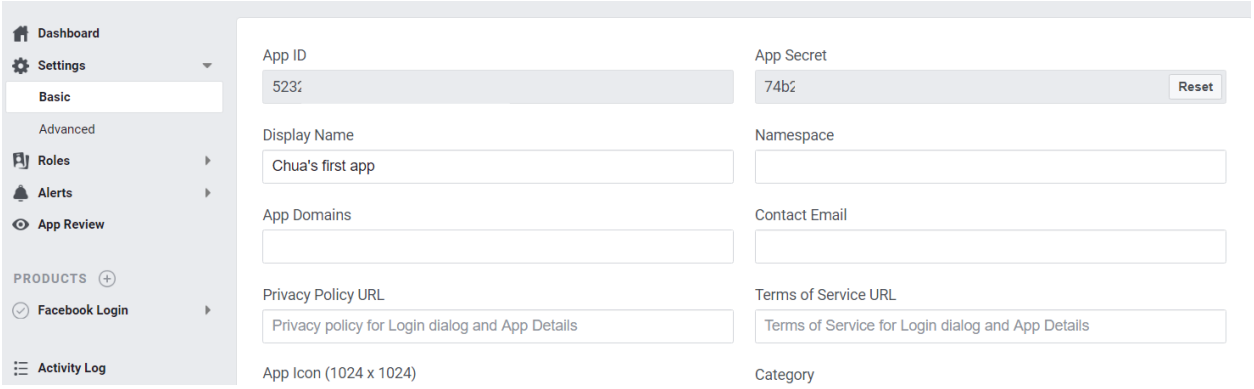




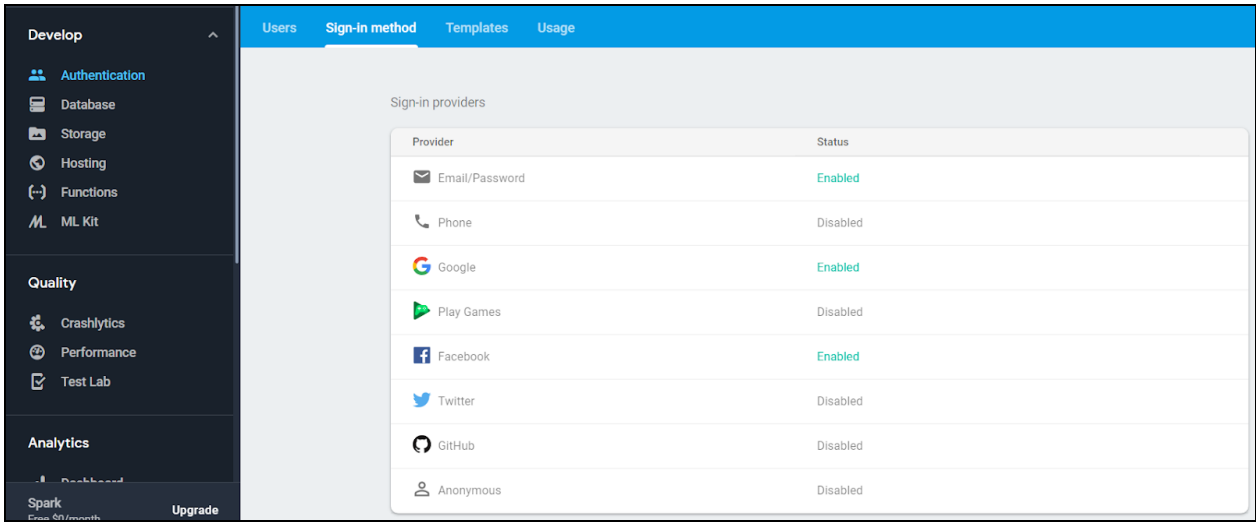
Step 4.4 - After being redirected to a page, click on Settings under the Products Section. Insert the OAuth Redirect URI from Firebase to the Facebook's Valid OAuth Redirect URIs.



Step 4.5 - To retrieve the App ID and App secret, navigate to Settings under the menu in the Facebook Developer page, click on show and copy them to Firebase.

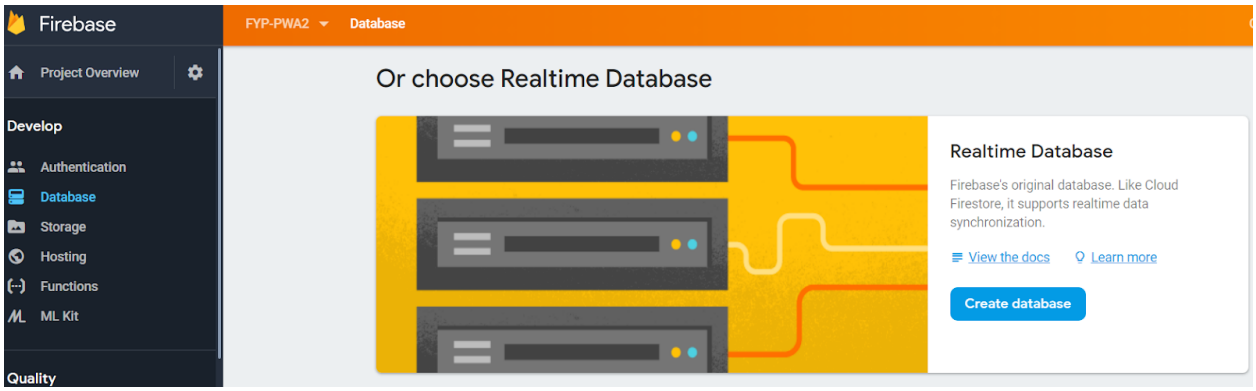


Step 5 - Click on Save once completed. You will now have 3 enabled providers for Authentication.

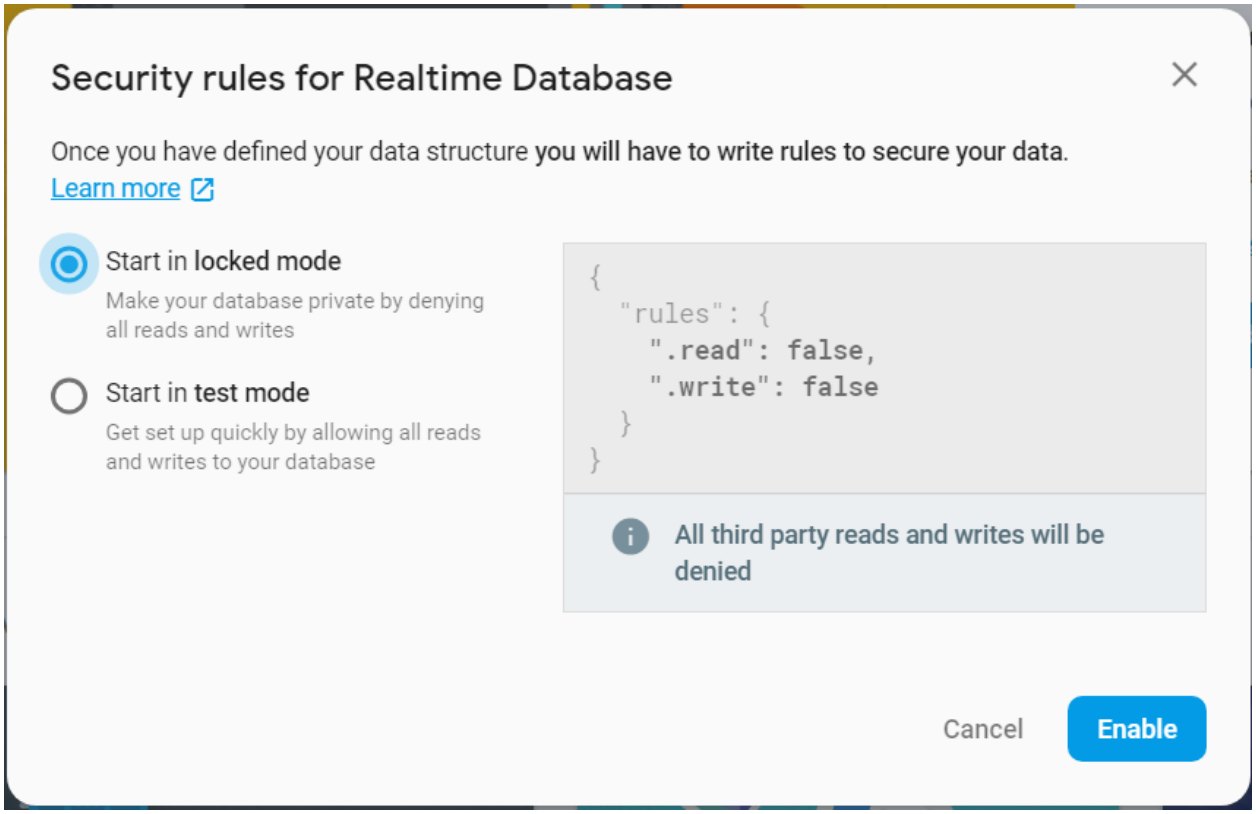


2.2 Setting up Firebase Database

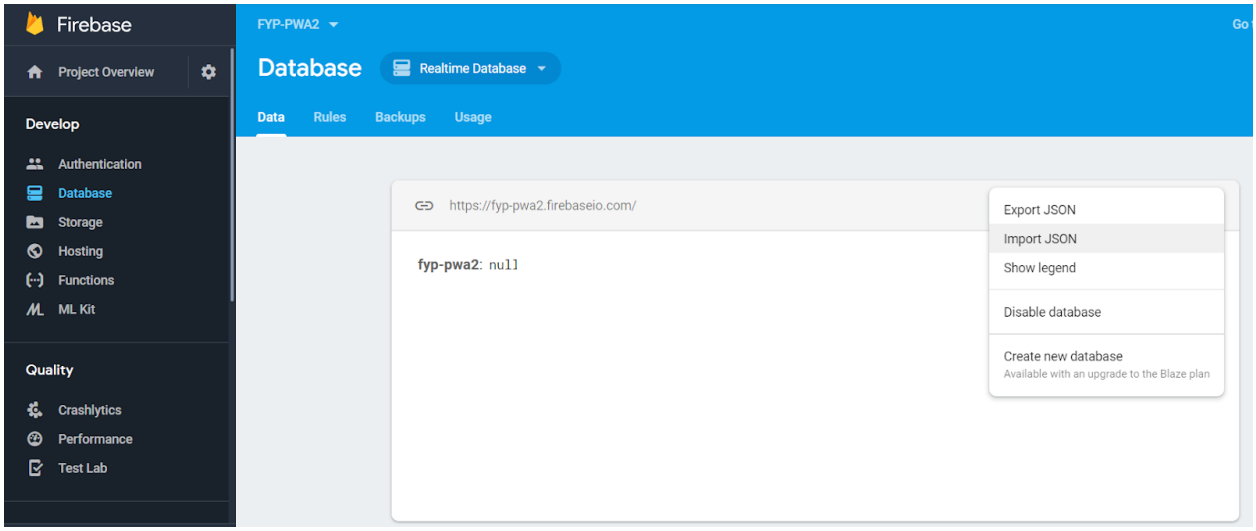
Step 1 - Click on Database on the menu and scroll down to ‘Or choose Realtime Database’. Click on create database and a pop-up will be shown.



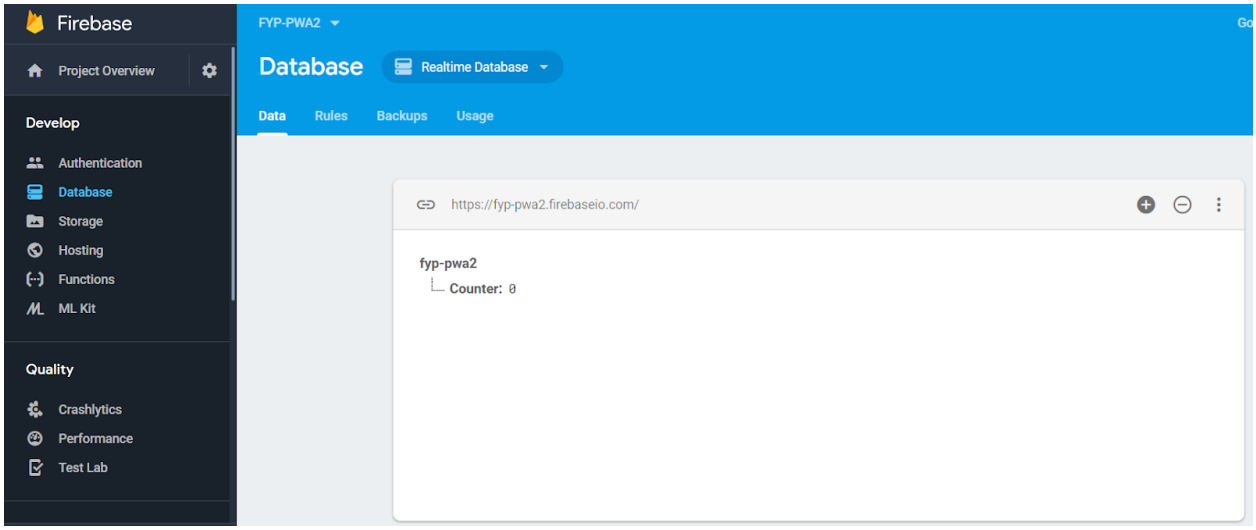
Step 2 - Click on Enable.



Step 3 - On the right side, click the more button (beside the + and - icon) and click on import JSON. Upload the file ‘pwa-firebase-hosting-export-final.json’ and click on Import.

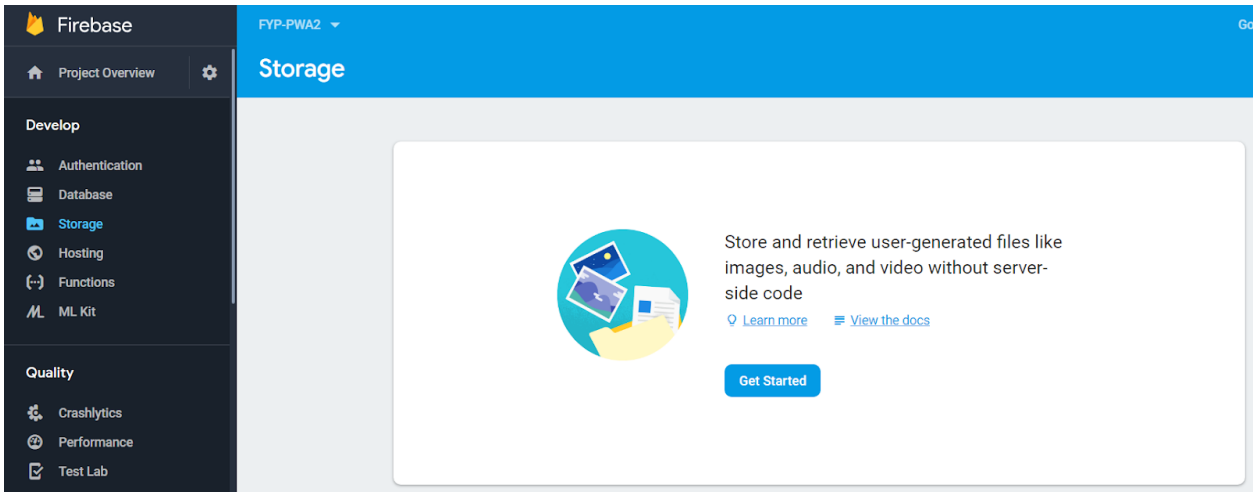


Step 4 - You should see something like this once you have imported successfully. Although you cannot see the structure of the database, you can see it once a route has been completed/failed, or when a user has been created.

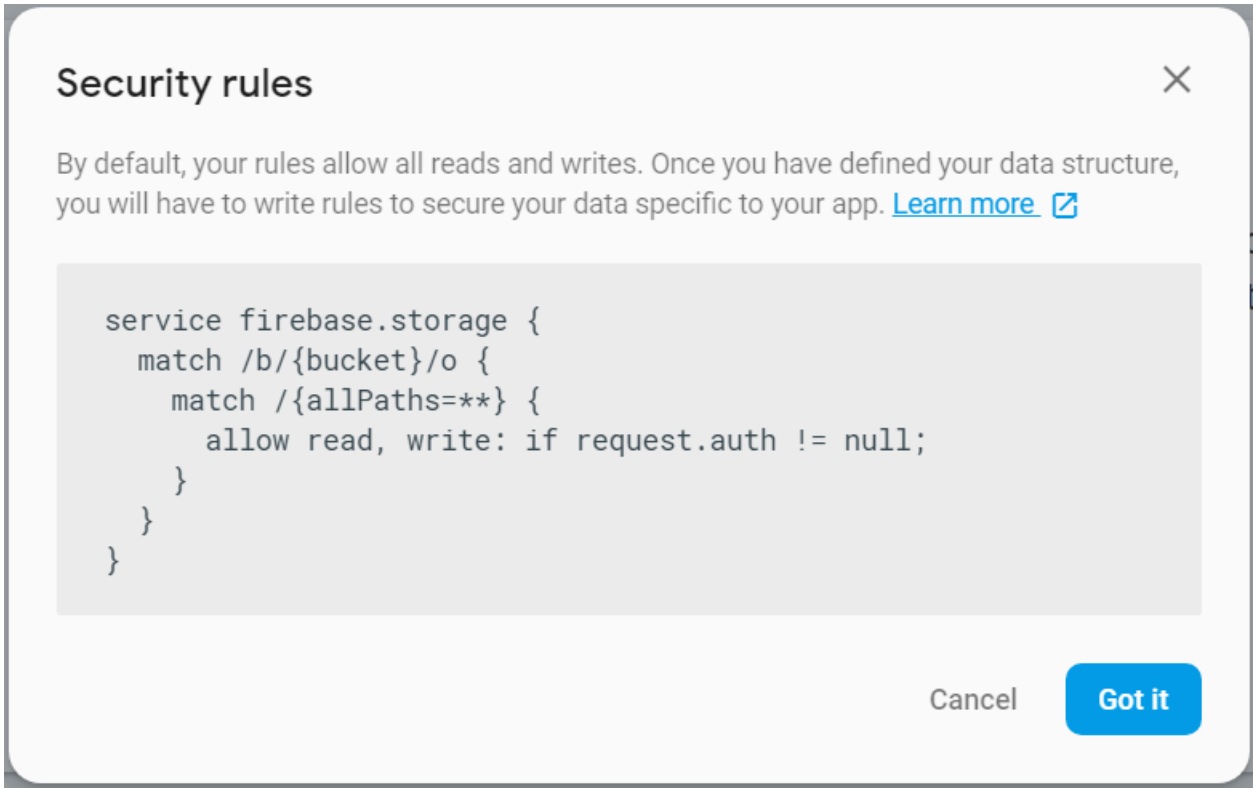


2.3 Setting up Firebase Storage

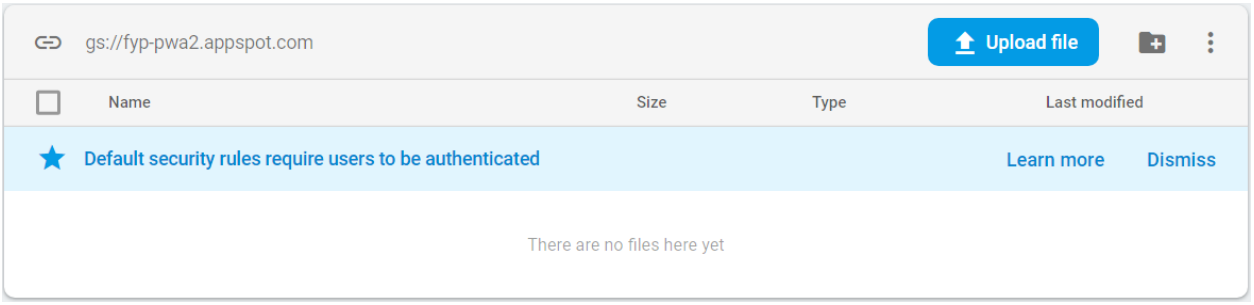
Step 1 - Navigate to Storage under the menu and click on ‘Get Started’.



Step 2 - A pop-up will appear. Click on ‘Got it’.

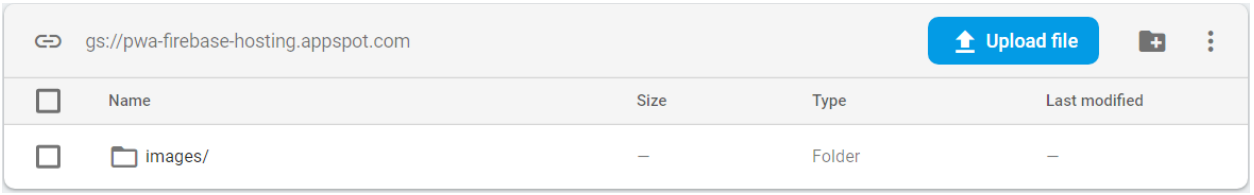


Step 3 - On the top right, click on the add folder icon



Step 4 - Make a directory something like this:
images/
images/facilities/
images/ProfilePicture/
images/units/

Your root folder will have something like this:



From there, the Firebase Storage is completed!

2.4 Setting up Firebase Hosting

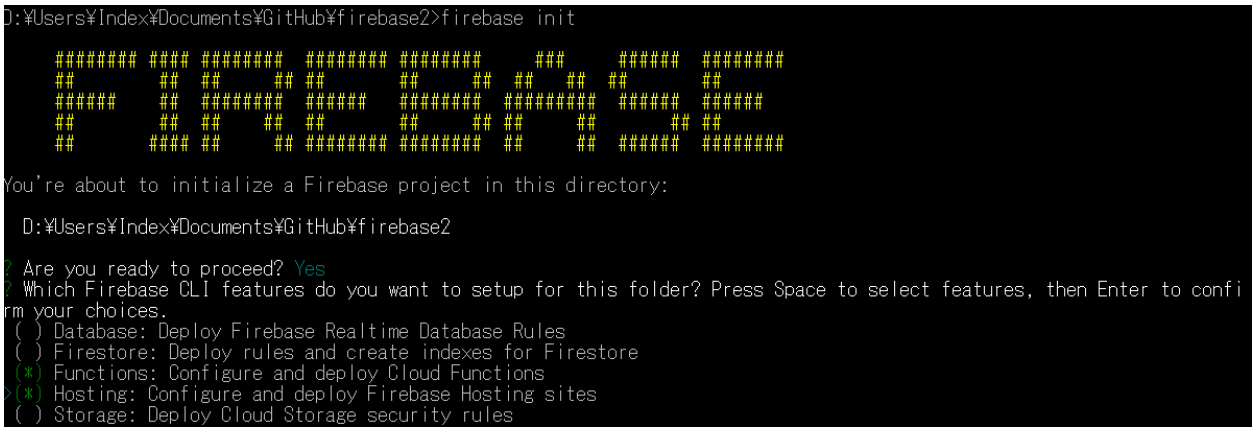
Before you proceed, make sure you have Node.js installed on your computer. If you do not have one, you can download from here: https://nodejs.org/en/.

Step 1 - Once you have Node.js installed, you will need to install the Firebase CLI Reference. The CLI Reference helps you to deploy your project to Firebase Hosting. Open up command prompt, type ‘npm install -g firebase-tools’ and enter.

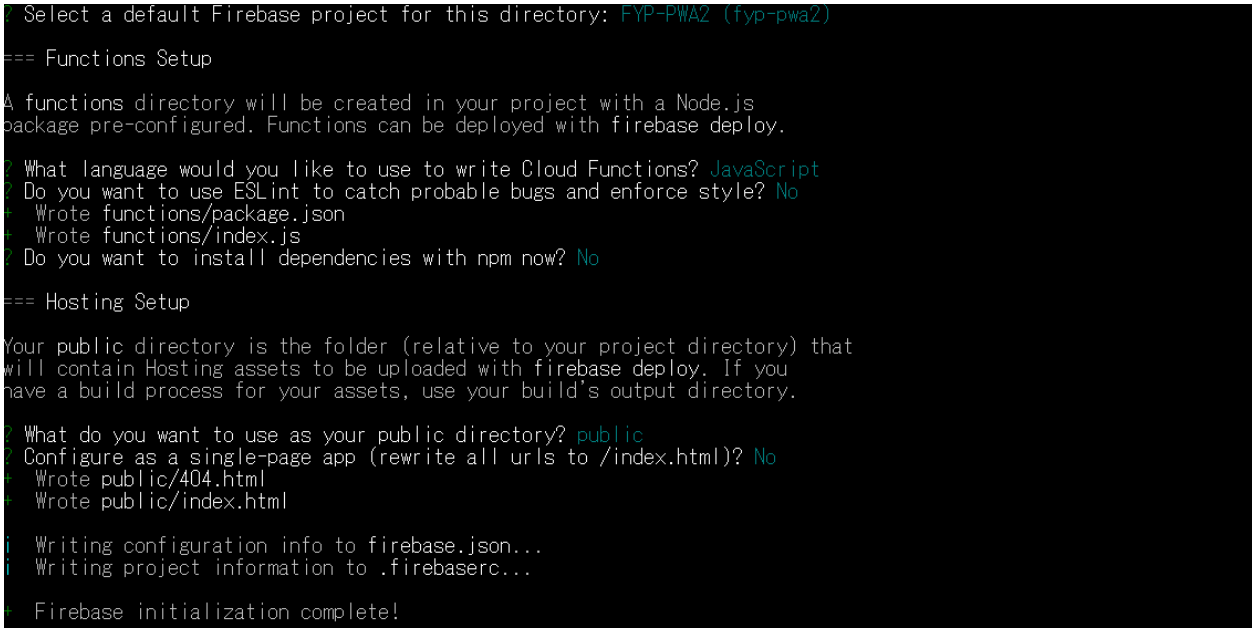
Step 2 - After a few seconds(or later), you will see something like this. This means that you have successfully installed. Login to firebase via this command ‘firebase login’. Enter your firebase credentials.

Step 3 - Once you have logged in, it means that you are connected to your account. Change your directory to a folder where you can setup the deployment with this command: ‘cd <your_directory_here>’. After changing your directory, it is time to create your firebase hosting folder. Type ‘firebase init’ and press enter.

Step 4 - Type ‘Y’ and press the Enter to confirm your directory. Then, select the features Functions and Hosting by navigating using up and down arrow keys, space to select features and Enter to confirm your choice.



Step 5 - After that, select the project that you are going to use, then press Enter twice. If you were given a choice to use ESLint or install the dependencies with npm, type ‘n’ and press Enter. Type enter again and if you are asked to configure as a single page-app, type ‘n’ and press Enter. From there, you have successfully created a folder!



Step 6 - Now, it is time to insert the files to your Firebase project. Unzip the backendsetup.zip and overwrite the files and folders on your project directory.

Step 7 - Before you start to deploy, delete the file at public/index.html. This file prevents you from entering any website that is hosted by the Node.js's Server API.

Step 8 - When you are done, use your command prompt and type in 'cd functions'. You will be directed to the functions folder. Type in 'npm install' and press enter. This will download and install all the plugins that are integrated in this project. You will get something like this after installation.

Step 9 -
Finally, you are now able to deploy your own firebase website!
To host on your website locally, type ‘firebase serve’ and press enter.
To host on your website on the firebase server, type ‘firebase deploy’ and press enter.
Take note that hosting on the firebase server may take some time.

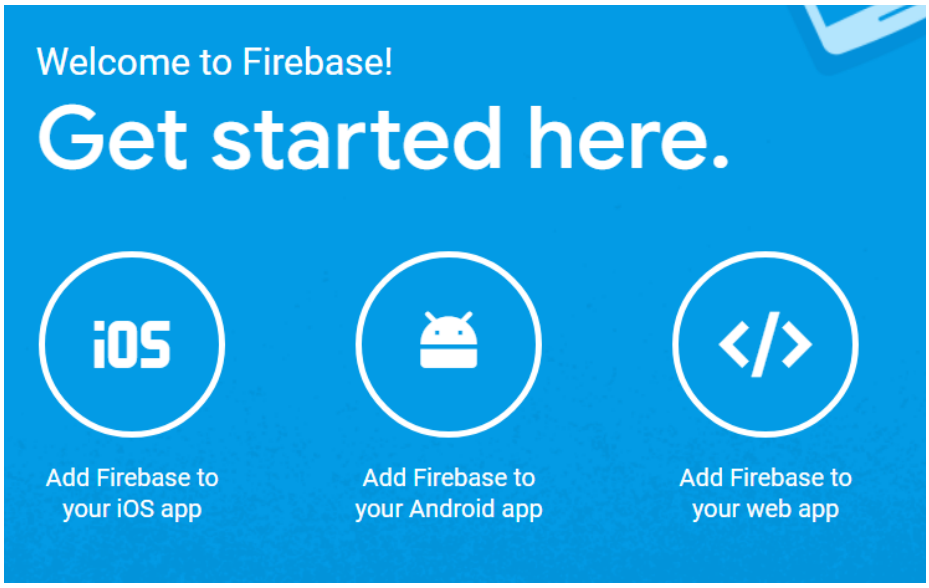
Once deployed on the firebase server, you should see something like this. Use the link given to access your website. You **should not deploy until you finish integrating the back-end site**.

```
D:¥Users¥Index¥Documents¥GitHub¥firebase¥functions>firebase deploy

=== Deploying to 'pwa-firebase-hosting'...

i  deploying functions, hosting
i  functions: ensuring necessary APIs are enabled...
+  functions: all necessary APIs are enabled
i  functions: preparing functions directory for uploading...
i  functions: packaged functions (1.86 MB) for uploading
+  functions: functions folder uploaded successfully
i  hosting: preparing public directory for upload...
!  Warning: Public directory does not contain index.html
+  hosting: 1 files uploaded successfully
i  functions: updating Node.js 6 function app(us-central1)...
+  functions[app(us-central1)]: Successful update operation.

+  Deploy complete!

Project Console: https://console.firebase.google.com/project/pwa-firebase-hosting/overview
Hosting URL: https://pwa-firebase-hosting.firebaseapp.com
```

2.5 Integrating Firebase into Front-End/Back-End

In order to integrate your Firebase Authentication, Database and Storage to your Web and Mobile Application, you require some keys in order to access them. In this setup, we will show you how to get your configure keys and where to insert them in both mobile and web application.

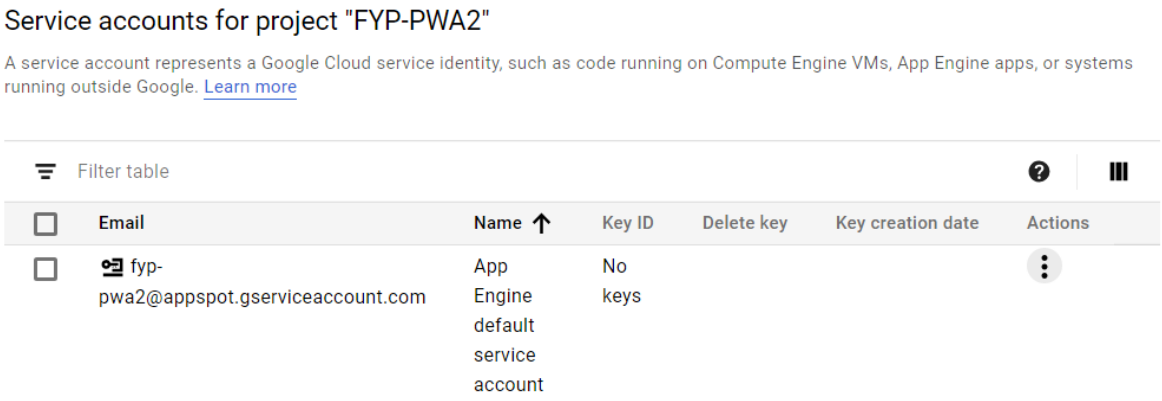
Step 1 - First, go to https://console.firebase.google.com/ and select the project that you have created on Step 2.0. From there, click on Project Overview located at the top left on the menu. You will see something like this:



Step 2 - Click on web app, and a code will appear something like this below. This is your configure key and you will need it for both front-end and back-end. Do not close this tab, or you can also copy the config into a notepad.

```
<script src="https://www.gstatic.com/firebasejs/5.3.1/firebase.js"></script>
<script>
  // Initialize Firebase
  var config = {
    apiKey: "AIzaSyCnrv                          mbLM",
    authDomain: "fyp-                  com",
    databaseURL: "https:                        .com",
    projectId: "fyp-     ",
    storageBucket: "fyp-                ",
    messagingSenderId: "        5245"
  };
  firebase.initializeApp(config);
</script>
```

Step 3 - Now, you need to create another key for back-end to retrieve the information from database. Go to the page https://console.cloud.google.com/iam-admin/serviceaccounts/ and select your project on the top menu. You will see something like this once you have selected.



Step 4 - Under “Actions”, click on Create Key and press Create. A .json file will be downloaded to your computer.

Next, we need to modify them on both front-end and back-end.

Front-End

Step 5 - On the front-end folder, navigate to Directie/src/app/ and open app.component.ts on a editor (such as a Notepad). Search for ‘firebase.initializeApp’ and then replace the config with the ones you have on Step 2.

```
constructor(public platform: Platform, public statusBar:
AuthProvider, public keyboard: Keyboard) {
  this.initializeApp();

  // used for an example of ngFor and navigation


  firebase.initializeApp({
    apiKey: "AIzaSXXXXXXXXXGa-CmBHidWrOGV2RSBI",
    authDomain: "pwa-firebase.firebaseapp.com",
    databaseURL: "https://pwa-firebase.firebaseio.com",
    projectId: "pwa-firebase",
    storageBucket: "pwa-firebase.appspot.com",
    messagingSenderId: "72415123456"
  });
```

Back-End

Step 6.1 - On the back-end folder, navigate to functions/ and paste the json file that you have just downloaded. Open index.js on a editor and search for ‘var firebaseConfig’. Replace the config with the ones you have on Step 2.

Step 6.2 - Below the config, you should see another configuration for firebase-admin. Rename the serviceAccount to the file name that you have just downloaded. After that, copy the databaseURL and storageBucket from the firebaseConfig.

```
//Firebase configurations
var firebaseConfig = {
  apiKey: "AIzaSyAVpnGuapXXXXXX-CmBHidWrOGV2RSBI",
  authDomain: "pwa-hosting.firebaseapp.com",
  databaseURL: "https://pwa-hosting.firebaseio.com/",
  projectId: "pwa-hosting",
  storageBucket: "pwa-hosting.appspot.com",
  messagingSenderId: "12345678"
};
firebase.initializeApp(firebaseConfig);

//firebase-admin configuration
var serviceAccount = require("./firebase-adminsdk-slnc4-9999999999.json");
admin.initializeApp({
  credential: admin.credential.cert(serviceAccount),
  databaseURL: "https://pwa-hosting.firebaseio.com/",
  storageBucket: "pwa-hosting.appspot.com"
});
var bucket = admin.storage().bucket();
var db = admin.database();
```

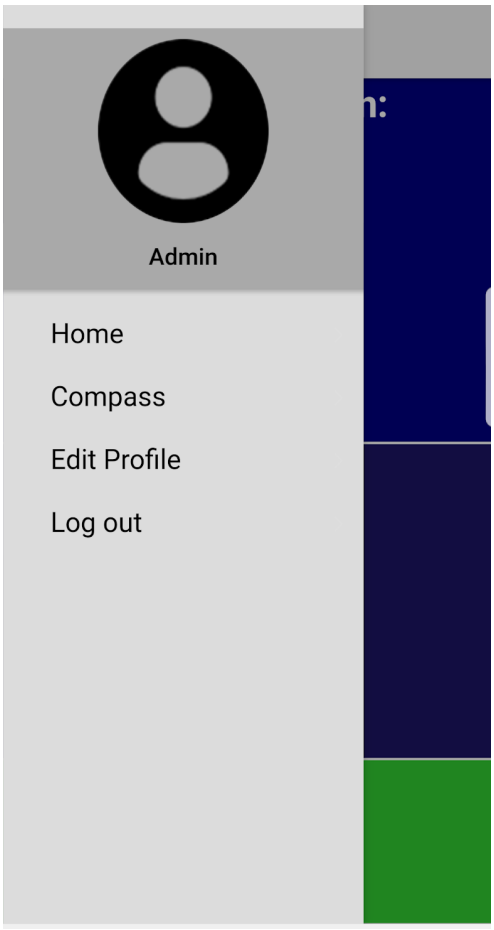
You are now ready to deploy your application!

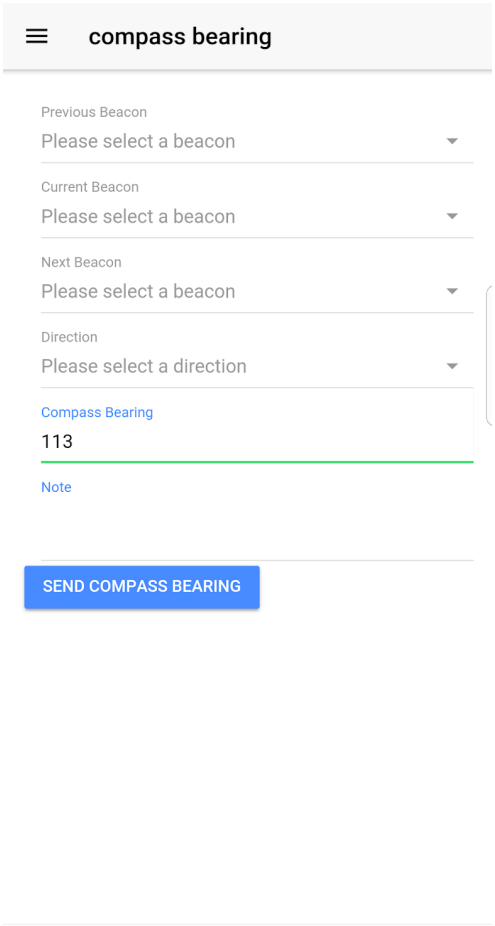
3.0 Setting up Beacon Direction (Bearings)

The administrator will have to set up the bearings associated with each beacon direction when implementing the system.

Step 1:
Place all the beacons at where you would want them in the indoor environment.

Step 2:





Using the Directie mobile application, login using an administrator account.
In the side menu, tap on **“Compass”**.

Step 3:
Stand beside one of the beacon.

Step 4:
For **“Previous Beacon”**, choose the beacon ID of the previous beacon to which you are standing beside.

Step 5:
For **“Current Beacon”**, choose the beacon ID of the beacon which you are standing beside.

Step 6:
For **“Next Beacon”**, choose the beacon ID of the next beacon to which you are standing beside.

Step 7:
Choose a direction for the path you want to set up in relation to the 3 beacons you have just selected.

Step 8:
When standing beside your “Current Beacon”, point your phone towards the direction of the “Next Beacon” that you have selected. The compass bearing will increase/decrease accordingly.

Step 9 (Optional):
If there is anything you want the user to take note of during his/her journey, enter a note under the “Note” field. What you enter will be read out along with the instructions when the user is about to reach that beacon.

Step 10:
Once you are sure that the bearing is at where your phone is facing to the next beacon, tap on **“Send Compass Bearing”** and the beacon direction along with the bearing will be added into the database.