



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



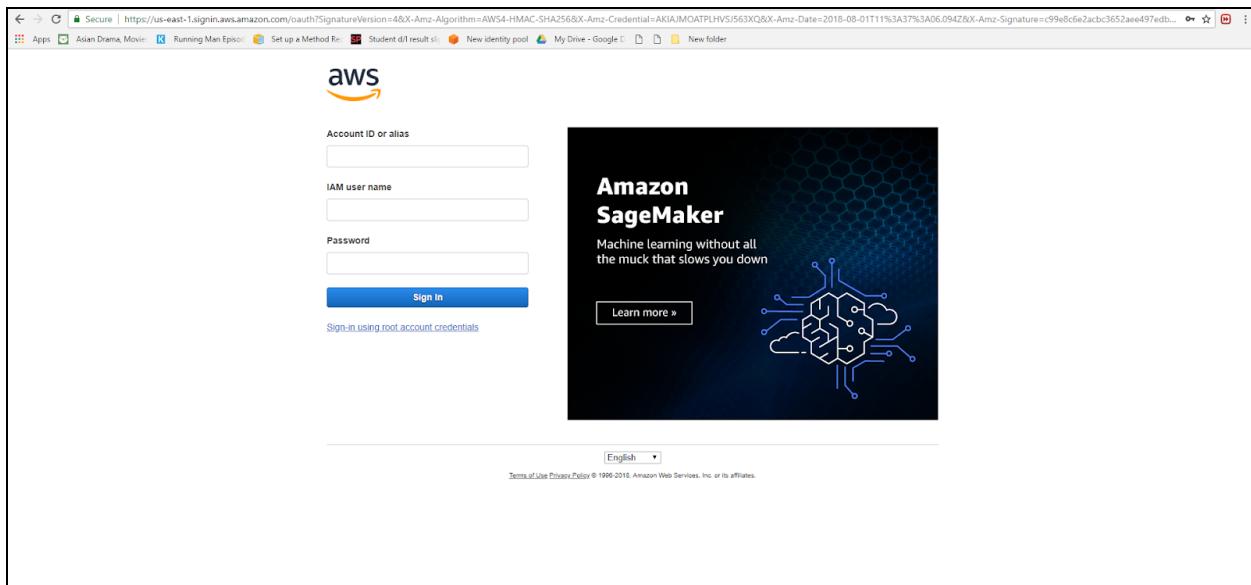
<b>1.0 Setting up AWS Database</b>	<b>3</b>
1.1 Setting up AWS Lambda	12
1.2 Setting up AWS API Gateway	15
<b>2.0 Setting up Firebase</b>	<b>26</b>
2.1 Setting up Firebase Authentication	27
2.2 Setting up Firebase Database	31
2.3 Setting up Firebase Storage	33
2.4 Setting up Firebase Hosting	35
2.5 Integrating Firebase into Front-End/Back-End	39
<b>3.0 Setting up Beacon Direction (Bearings)</b>	<b>42</b>



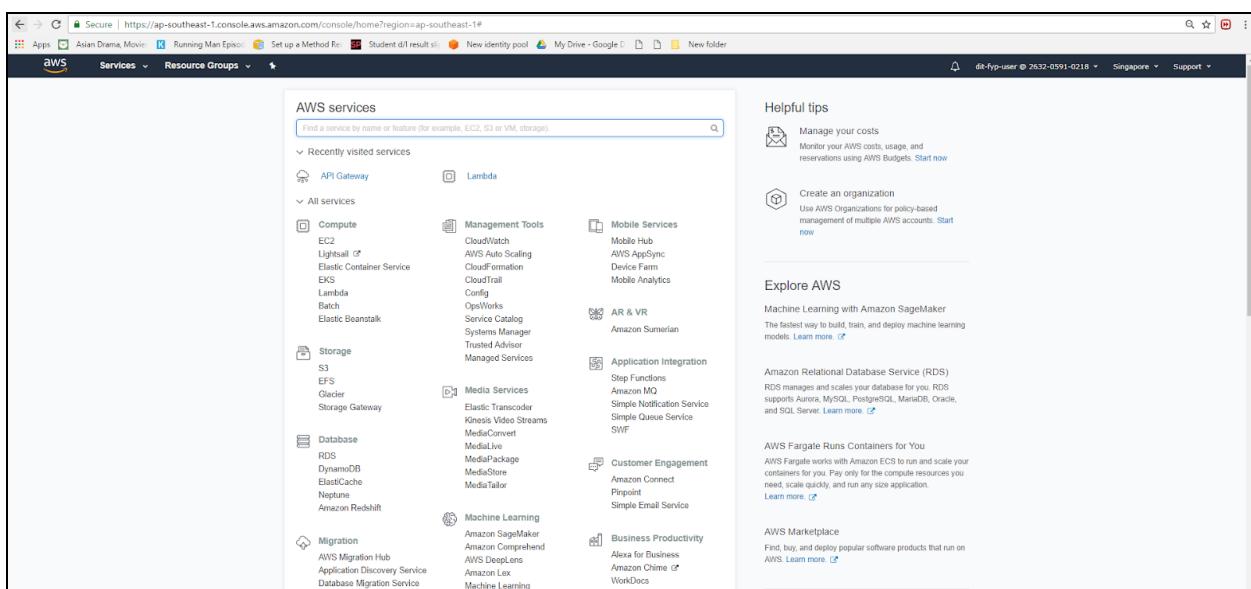
## 1.0 Setting up AWS Database

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”

The screenshot shows the AWS Services Catalog interface. In the top-left search bar, the text 'RDS' is typed. Below the search bar, the results list includes 'RDS Managed Relational Database Service'. To the right of the search bar, there is a 'Helpful tips' section with links to 'Manage your costs' and 'Create an organization'. Further down, there is an 'Explore AWS' section with links to 'Machine Learning with Amazon SageMaker', 'Amazon Relational Database Service (RDS)', 'AWS Fargate Runs Containers for You', and 'AWS Marketplace'.

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

The screenshot shows the Amazon RDS Instances page. On the left, a navigation menu is open under 'Amazon RDS' with options like 'Instances' (which is highlighted in orange), 'Clusters', 'Performance Insights', 'Snapshots', 'Reserved instances', 'Subnet groups', 'Parameter groups', 'Option groups', 'Events', 'Event subscriptions', and 'Recommendations'. The main content area shows a table titled 'Instances (1)'. The table has columns for 'DB instance', 'Engine', 'Status', 'CPU', and 'Current activity'. There is one row for an instance named 'directedb' which is running on 'MySQL' and is 'available'. At the top right of the table, there is a red 'Create database' button.



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

The screenshot shows the 'Amazon RDS' service in the AWS console. On the left, the 'Amazon RDS' sidebar lists options like Dashboard, Instances, Clusters, Performance Insights, Snapshots, Reserved instances, Subnet groups, Parameter groups, Option groups, Events, Event subscriptions, and Recommendations. The main panel is titled 'Step 1 Select engine'. It has four steps listed: Step 1 Select engine, Step 2 Choose use case, Step 3 Specify DB details, and Step 4 Configure advanced settings. The 'Select engine' section contains a grid of database engine options:

Engine options		
<input type="radio"/> Amazon Aurora Amazon Aurora	<input checked="" type="radio"/> MySQL MySQL logo	<input type="radio"/> MariaDB MariaDB logo
<input type="radio"/> PostgreSQL PostgreSQL logo	<input type="radio"/> Oracle Oracle logo	<input type="radio"/> Microsoft SQL Server Microsoft SQL Server logo

**MySQL**  
MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.

- Supports database size up to 16 TiB.
- Instances offer up to 32 vCPUs and 244 GiB Memory.
- Supports automated backup and point-in-time recovery.
- Supports cross-region read replicas.

Only enable options eligible for RDS Free Usage Tier [Info](#)

Cancel **Next**

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

The screenshot shows the 'Amazon RDS' service in the AWS console. The sidebar and main panel structure are identical to the previous step. The 'Step 1 Select engine' panel is still visible. The 'Step 2 Choose use case' panel is now active, showing the 'Use case' section. It asks, "Do you plan to use this database for production purposes?" and lists three options:

Use case	Status
Production - Amazon Aurora	Recommended
MySQL-compatible, enterprise-class database at 1/10th the cost of commercial databases.	
Production - MySQL	
Use Multi-AZ Deployment and Provisioned IOPS Storage as defaults for high availability and fast, consistent performance.	
Dev/Test - MySQL	
This instance is intended for use outside of production or under the RDS Free Usage Tier.	

Billing is based on [RDS pricing](#).

Cancel **Previous** **Next**



## Step 6 - Follow the image below for the instance specifications

Specify DB details

**Instance specifications**  
Estimate your monthly costs for the DB Instance using the AWS Simple Monthly Calculator.

DB engine MySQL Community Edition  
License model [Info](#) general-public-license  
DB engine version [Info](#) mysql 5.6.39

**Known Issues/Limitations**  
Review the Known Issues/Limitations to learn about potential compatibility issues with specific database versions.

DB instance class [Info](#) db.t2.small — 1 vCPU, 2 GiB RAM  
Multi-AZ deployment [Info](#)  
 Create replica in different zone Creates a replica in a different Availability Zone (AZ) to provide data redundancy, eliminate I/O freezes, and minimize latency spikes during system backups.  
 No  
Storage type [Info](#) General Purpose (SSD)  
Allocated storage 20 GiB  
(Minimum: 20 GiB, Maximum: 16384 GiB) Higher allocated storage may improve IOPS performance.

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

**Settings**

**DB instance identifier [Info](#)**  
Specify a name that is unique for all DB instances owned by your AWS account in the current region.  
  
DB instance identifier is case insensitive, but stored as all lower-case, as in "mydbinstance". Must contain from 1 to 63 alphanumeric characters or hyphens (1 to 15 for SQL Server). First character must be a letter. Cannot end with a hyphen or contain two consecutive hyphens.

**Master username [Info](#)**  
Specify an alphanumeric string that defines the login ID for the master user.  
  
Master Username must start with a letter. Must contain 1 to 16 alphanumeric characters.

**Master password [Info](#)**  
  
Master Password must be at least eight characters long, as in "mypassword". Can be any printable ASCII character except "/", "", or "@".

**Confirm password [Info](#)**

**Cancel** **Previous** **Next**

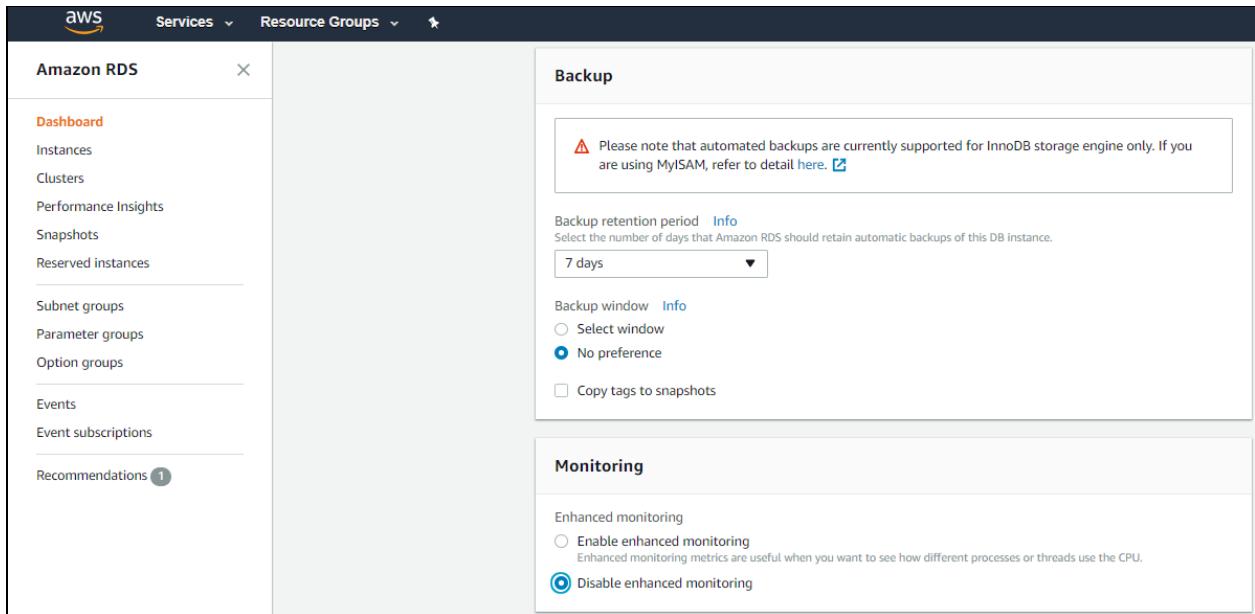


**Step 7** - Follow the image below for advance settings and then click “create database” button

The screenshot shows the 'Configure advanced settings' step of the RDS creation wizard. Under 'Network & Security', the VPC dropdown is set to 'vpc-40c1a527'. The 'Subnet group' dropdown is set to 'default-vpc-40c1a527'. Under 'Public accessibility', the 'Yes' radio button is selected. Under 'Availability zone', the 'No preference' dropdown is selected. Under 'VPC security groups', the 'Choose existing VPC security groups' radio button is selected, and the 'default' option is chosen from the dropdown.

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.

The screenshot shows the 'Database options' and 'Encryption' sections of the RDS creation wizard. In the 'Database options' section, the 'Database name' is 'dbname', the 'Database port' is '3306', the 'DB parameter group' is 'default.mysql5.6', the 'Option group' is 'default:mysql-5-6', and the 'IAM DB authentication' is set to 'Disable'. In the 'Encryption' section, the 'Disable encryption' radio button is selected.

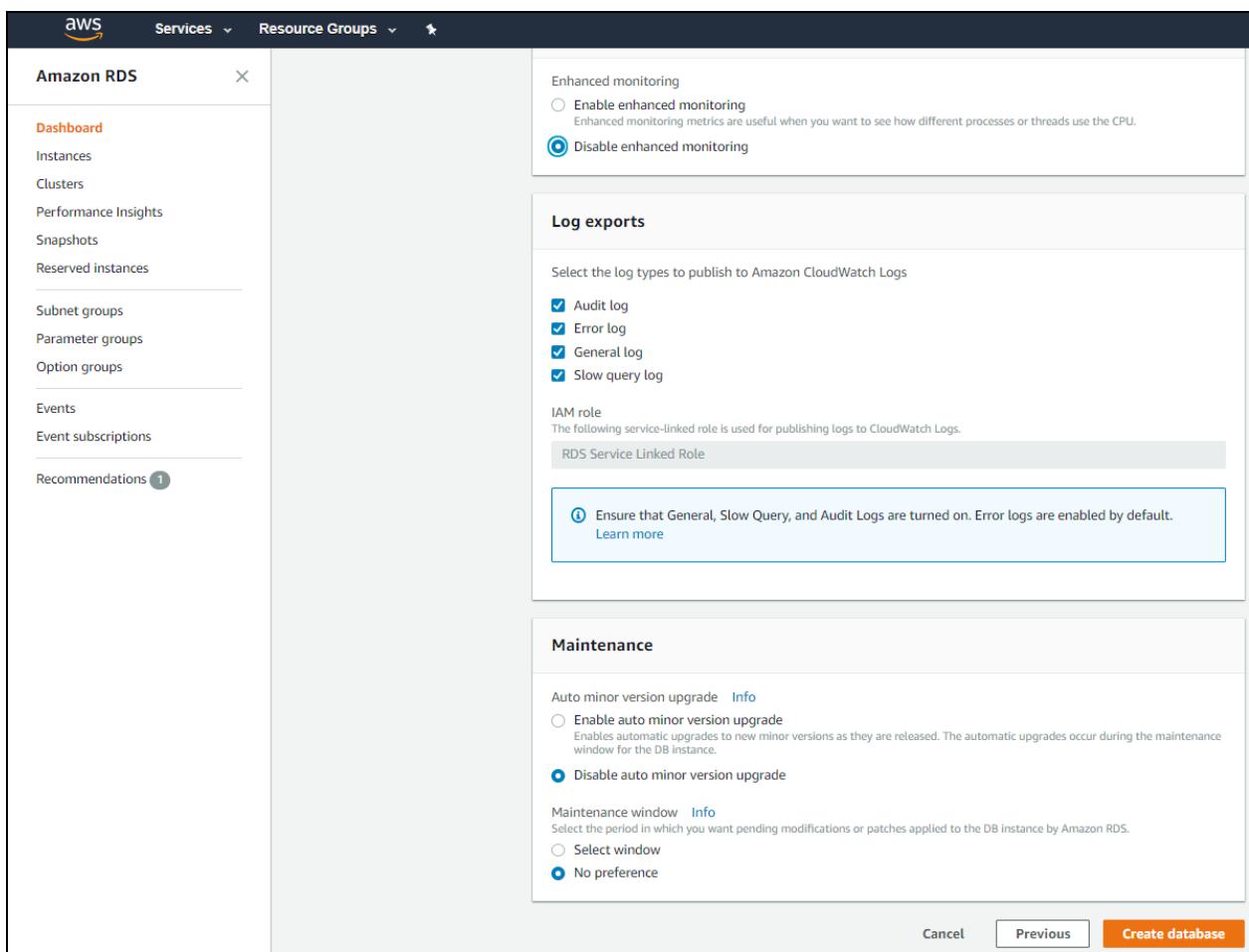


The screenshot shows the AWS RDS configuration interface for a database instance. On the left, a sidebar lists various options like Dashboard, Instances, Clusters, and Performance Insights. The main panel is titled 'Backup' and contains the following settings:

- Backup retention period**: Set to 7 days.
- Backup window**: Set to 'No preference'.
- Copy tags to snapshots**: Unchecked.

Below the backup section is a 'Monitoring' section with the following options:

- Enhanced monitoring**: Set to 'Disable enhanced monitoring'.



This screenshot shows the continuation of the RDS configuration process. The sidebar remains the same. The main panel now includes the 'Log exports' section:

- Select the log types to publish to Amazon CloudWatch Logs**:
  - Audit log
  - Error log
  - General log
  - Slow query log
- IAM role**: A service-linked role named 'RDS Service Linked Role' is selected.

A note at the bottom of the log exports section states: **Ensure that General, Slow Query, and Audit Logs are turned on. Error logs are enabled by default.** [Learn more](#)

At the bottom of the page are three buttons: 'Cancel', 'Previous', and a prominent orange 'Create database' button.



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

The screenshot shows the AWS RDS Instances page. On the left, there's a sidebar with options like Dashboard, Instances (which is selected), Clusters, Performance Insights, Snapshots, Reserved instances, Subnet groups, Parameter groups, and Option groups. The main area is titled 'Instances (1)' and shows a table with one row. The row details are: DB instance name is 'directiedb', Engine is MySQL, Status is 'available' with a green circle icon, CPU usage is 2.67%, Current activity shows 0 connections, Maintenance is 'none', Class is 'db.t2.small', and VPC is 'vpc-40'. There are buttons for 'Instance actions', 'Restore from S3', and 'Create database'.

**Step 9** - Copy the Endpoint from your AWS RDS instance

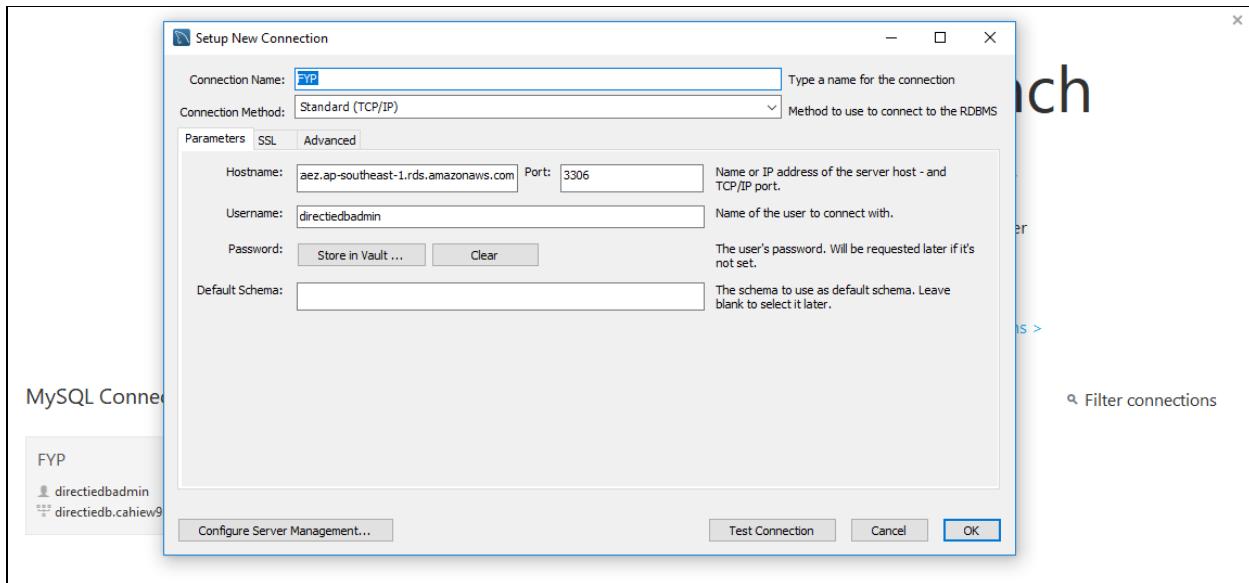
The screenshot shows the AWS RDS Connect page for the 'directiedb' instance. It has sections for 'Endpoint' (directiedb.cahiew9mtaez.ap-southeast-1.rds.amazonaws.com), 'Port' (3306), and 'Publicly accessible' (Yes). Below this is a section for 'Security group rules (3)' with a search bar and navigation buttons.

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

The screenshot shows the MySQL Workbench welcome screen. It features a central title 'Welcome to MySQL Workbench' with a subtext explaining its purpose: 'MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.' Below the title are links to 'Browse Documentation >', 'Read the Blog >', and 'Discuss on the Forums >'. On the left, there's a sidebar with icons for Home, Database, Tools, and Help. The main area shows a list of 'MySQL Connections' with one entry: 'FYP' (with a profile picture placeholder) and 'directiedbadmin' (with a profile picture placeholder).



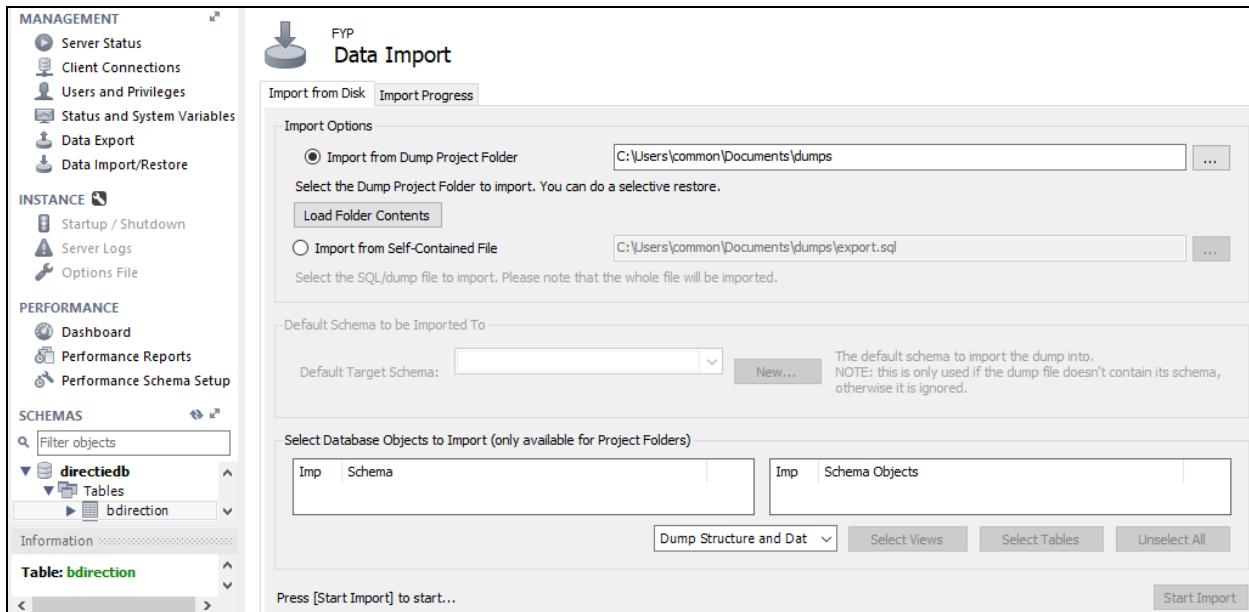
**Step 11 - Enter your Connection Name and paste the Endpoint in the Hostname. For username and password enter the one you set in step 6 and click “OK” button**



**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

**Data Import**

Import from Disk Import Progress

**Import Options**

Import from Dump Project Folder C:\Users\common\Documents\dumps ...  
Select the Dump Project Folder to import. You can do a selective restore.  
 Import from Self-Contained File C:\Users\common\Documents\dumps\directiedb.sql ...  
Select the SQL/dump file to import. Please note that the whole file will be imported.

**Default Schema to be Imported To**

Default Target Schema: [dropdown] New... The default schema to import the dump into.  
NOTE: this is only used if the dump file doesn't contain its schema, otherwise it is ignored.

**Select Database Objects to Import (only available for Project Folders)**

Imp Schema | Imp Schema Objects | Dump Structure and Data | Select Views | Select Tables | Unselect All | Start Import

Press [Start Import] to start...



## 1.1 Setting up AWS Lambda

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

The screenshot shows the AWS Management Console homepage. At the top, there's a search bar with the text "Lambda". Below it, under the heading "AWS services", there's a list of services. The "Lambda" service is highlighted with a blue border. To the right of the search bar, there's a "Helpful tips" section with links to "Manage your costs" and "Create an organization". Further down, there's an "Explore AWS" section with links to "Machine Learning with Amazon SageMaker" and "Amazon Relational Database Service (RDS)".

**Step 2 - Click on “Create function” button**

The screenshot shows the "Functions" page within the AWS Lambda service. On the left, there's a sidebar with "AWS Lambda" and "Dashboard" options. The main area displays a table of existing Lambda functions. The table has columns for "Function name", "Description", "Runtime", "Code size", and "Last Modified". The first four functions listed are "CreateTableAddRecordsAndRead", "Select", "directive", and "Tryingout". Above the table, there's a search bar and a "Create function" button. The URL in the browser address bar is "https://ap-southeast-1.console.aws.amazon.com/lambda/home?region=ap-southeast-1#functions".



**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.

Author from scratch  
Start with a simple “hello world” example.

Blueprints  
Choose a preconfigured template as a starting point for your Lambda function.

Serverless Application Repository  
Find and deploy serverless apps published by developers, companies, and partners on AWS.

**Author from scratch** [Info](#)

Name

Runtime

Role  
Defines the permissions of your function. Note that new roles may not be available for a few minutes after creation. [Learn more](#) about Lambda execution roles.

Lambda will automatically create a role with permissions from the selected policy templates. Note that basic Lambda permissions (logging to CloudWatch) will automatically be added. If your function accesses a VPC, the required permissions will also be added.

Role name  
Enter a name for your new role.

ⓘ This new role will be scoped to the current function. To use it with other functions, you can modify it in the IAM console.

Policy templates  
Choose one or more policy templates. A role will be generated for you before your function is created. [Learn more](#) about the permissions that each policy template will add to your role.

Cancel **Create function**

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

Screenshot of the AWS Lambda Functions page showing the newly created function "directie1".

The function details:

- ARN: arn:aws:lambda:ap-southeast-1:263205910218:function:directie1
- Throttle, Qualifiers, Actions, Select a test event, Test, Save buttons.

The Configuration tab is selected, showing the Designer section:

- Add triggers: Click on a trigger from the list below to add it to your function.
- Available triggers:
  - API Gateway
  - AWS IoT
  - CloudWatch Events
  - AWS Lambda
  - CloudWatch Logs
  - Amazon DynamoDB
  - CodeCommit
  - Cognito Sync Trigger
  - DynamoDB
- Resources the function's role has access to will be shown here.

A green message box says: "Congratulations! Your Lambda function "directie1" has been successfully created. You can now change its code and configuration. Click on the "Test" button to input a test event when you are ready to test your function." X



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

The screenshot shows the AWS Lambda function configuration interface. In the 'Function code' section, the 'Code entry type' dropdown is currently set to 'Edit code inline'. The 'Handler' field is set to 'index.handler'. The code editor displays the following Node.js code:

```
1 exports.handler = (event, context, callback) => {
2     // TODO implement
3     callback(null, 'Hello from Lambda');
4 };
```

**Step 6** - Download the .ZIP file in this link :

[https://drive.google.com/open?id=1gLdG\\_dkLwit0VO0uhbPFpuYv0frKIgTD](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.

The screenshot shows the AWS Lambda function configuration interface. In the 'Function code' section, the 'Code entry type' dropdown is now set to 'Upload a .ZIP file'. The 'Handler' field is set to 'directie.handler'. The 'Function package' section contains a 'Upload' button and a note: 'For files larger than 10 MB, consider uploading via S3.'

**Step 8** - Click on the “Save” Button

The screenshot shows the AWS Lambda function configuration interface after saving. The 'Function code' section remains the same as in Step 7. The 'Function package' section now shows a successfully uploaded file: 'directie-65f90fe1-ca63-48b4-8cc8-33dd2e3de99b.zip (274.0 kB)'. A note at the bottom says: 'For files larger than 10 MB, consider uploading via S3.'



## 1.2 Setting up AWS API Gateway

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

The screenshot shows the AWS Management Console homepage. At the top, there is a search bar with the text "Secure | https://ap-southeast-1.console.aws.amazon.com/console/home?region=ap-southeast-1". Below the search bar, the AWS logo is followed by "Services" and "Resource Groups". On the right side, there are "Helpful tips" and "Explore AWS" sections. The main area displays various AWS services in a grid:

Category	Service	Service	Service
All services	API Gateway	Lambda	CloudWatch
	IoT Core	Simple Notification Service	
	Compute	Management Tools	Mobile Services
	EC2	CloudWatch	Mobile Hub
	Lightsail	AWS Auto Scaling	AWS AppSync
	Elastic Container Service	CloudFormation	Device Farm
	EKS	CloudTrail	Mobile Analytics
	Lambda	Config	
	Batch	OpsWorks	AR & VR
	Elastic Beanstalk	Service Catalog	Amazon Sumerian
Storage	Systems Manager		
	Trusted Advisor	Managed Services	Application Integration
	S3		

**Step 2** - Click on “Create API”

The screenshot shows the "Amazon API Gateway" service page. The left sidebar includes options like "APIs", "Usage Plans", "API Keys", "Custom Domain Names", "Client Certificates", "VPC Links", and "Settings". A prominent blue button labeled "+ Create API" is located in the center. Two API entries are listed:

API Name	Created On	Description	Endpoint Configuration
directie	Created on 5/2/2018	No description.	Endpoint Type: Regional
LambdaSimpleProxy	Created on 4/26/2018	EG	Endpoint Type: Regional



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

The screenshot shows the 'Create new API' form in the AWS API Gateway console. The 'API name\*' field contains 'directie'. The 'Description' and 'Endpoint Type' fields are empty. The 'Create API' button is visible at the bottom right.

### Step 4 - Download the text file in this link :

[https://drive.google.com/open?id=13I7lrRuHFu68HK0dGiBTXgYWrhS\\_DvCZ](https://drive.google.com/open?id=13I7lrRuHFu68HK0dGiBTXgYWrhS_DvCZ)

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

The screenshot shows the 'Models' page in the AWS API Gateway console. The 'Create' button is highlighted in blue. A tooltip 'Select a model' is shown above the 'Create' button. Below the 'Create' button, there are two entries: 'Empty' and 'Error'.



**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.

Models      **Create**

New Model

Provide a name, content type, and a schema for your model. Models use [JSON schema](#).

Model name\*

Content type\*

Model description

Model schema\*

1

\* Required      [Cancel](#)      **Create model**

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

APIs      **Resources**      Actions **New Child Resource**

directie      directie1

Resources

Stages

Authorizers

Gateway Responses

Models

Resource Policy

Documentation

Settings

/

RESOURCE ACTIONS

Create Method

Create Resource

Enable CORS

Edit Resource Documentation

API ACTIONS

Deploy API

Import API

Edit API Documentation

Delete API

create a new child resource for your resource. [?](#)

[is proxy resource](#)

Resource Name\*

Resource Path\*

You can add path parameters using brackets. For example, the resource Configuring /{proxy+} as a proxy resource catches all requests to its sub-requests to /, add a new ANY method on the / resource.

API Gateway CORS

\* Required



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

New Child Resource

Configure as  proxy resource

Resource Name\*

Resource Path\*

You can add path parameters using brackets. For example, the resource path {username} represents a path parameter called 'username'. Configuring /{proxy+} as a proxy resource catches all requests to its sub-resources. For example, it works for a GET request to /foo. To handle requests to /, add a new ANY method on the / resource.

Enable API Gateway CORS

\* Required

Cancel Create Resource

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

APIs

Resources

/

/ {type}

Actions

RESOURCES ACTIONS

- Create Method
- Create Resource
- Enable CORS
- Edit Resource Documentation
- Delete Resource

API ACTIONS

- Deploy API
- Import API
- Edit API Documentation
- Delete API

No methods defined for the resource.

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

APIs

Resources

/

/ {type}

Actions

ANY

DELETE

GET

HEAD

OPTIONS

PATCH

POST

PUT

No methods defined for the resource.



**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

Resources Actions /{type} - POST - Setup

Choose the integration point for your new method.

Integration type  Lambda Function ⓘ  
 HTTP ⓘ  
 Mock ⓘ  
 AWS Service ⓘ  
 VPC Link ⓘ

Use Lambda Proxy integration ⓘ

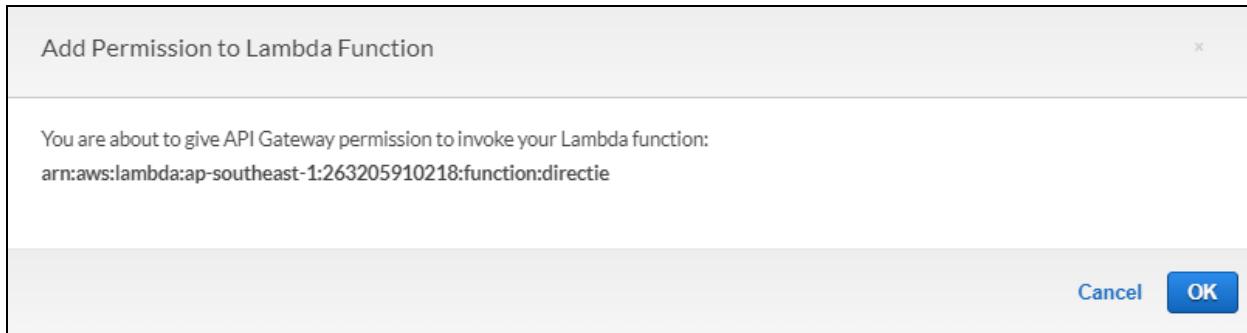
Lambda Region ap-southeast-1

Lambda Function directie1

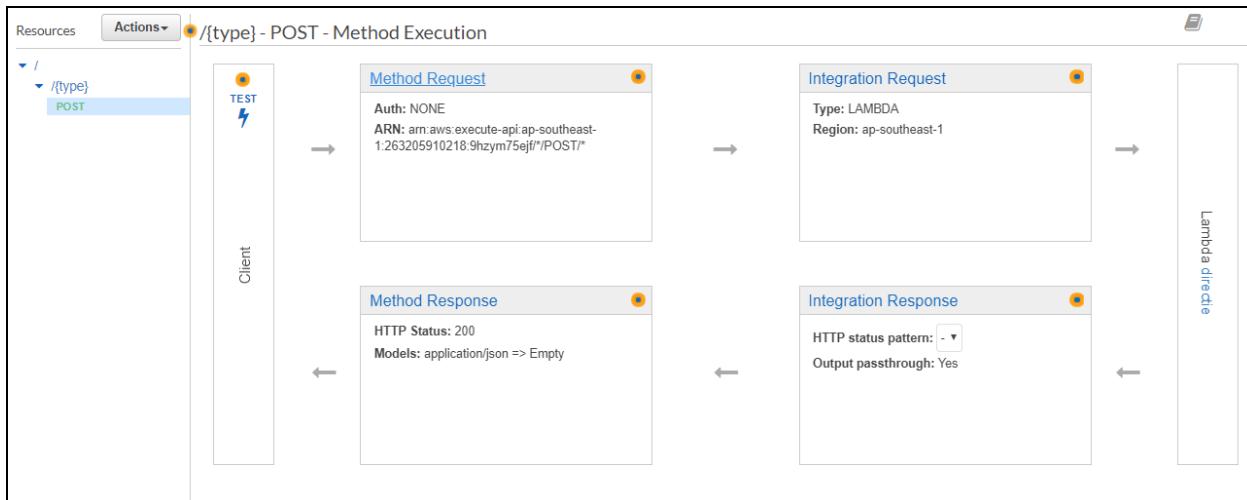
Use Default Timeout  ⓘ

Save

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”

Method Execution /{type} - POST - Method Request

Authorization: NONE

Request Validator: Validate body, query string parameters, and headers

API Key Required: true

Name: type

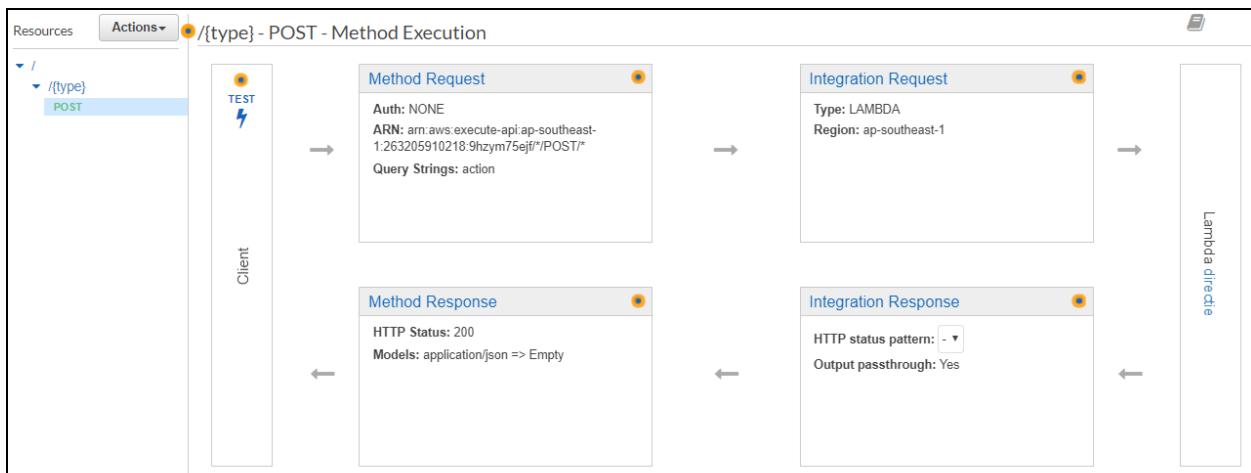
Content type: application/json

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](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”

The screenshot shows the Mule Studio interface for configuring a POST request. The left sidebar lists resources: a single resource named '/' with one endpoint '/{type}' and a POST method selected. The main panel is titled 'HTTP Headers' and contains a section for 'Mapping Templates'. A dropdown menu for 'Request body passthrough' has three options: 'When no template matches the request Content-Type header' (radio button), 'When there are no templates defined (recommended)' (radio button, selected), and 'Never' (radio button). Below this is a 'Content-Type' field set to 'application/json'. A 'Generate template:' dropdown is present. At the bottom right are 'Cancel' and 'Save' buttons.



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

The screenshot shows the AWS API Gateway Actions panel. On the left, there's a tree view with 'Resources' expanded, showing a path '/(type)'. Underneath it, 'POST' is selected. In the center, under 'Actions', 'Enable CORS' is highlighted with a blue circle. Below the actions, there are sections for 'Responses for directive1 API', 'Methods' (set to POST and OPTIONS), 'Access-Control-Allow-Methods' (set to POST, OPTIONS), 'Access-Control-Allow-Headers' (set to 'Content-Type X-Amz-Date Authorization'), and 'Access-Control-Allow-Origin' (set to '\*'). At the bottom right of the panel is a blue button labeled 'Enable CORS and replace existing CORS headers'.

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

The screenshot shows a confirmation dialog box. The title is 'Confirm method changes'. The main content asks: 'The following modifications will be made to this resource's methods and will replace any existing values. Are you sure you want to continue?'. Below this is a bulleted list of changes:

- Create OPTIONS method
- Add 200 Method Response with Empty Response Model to OPTIONS method
- Add Mock Integration to OPTIONS method
- Add 200 Integration Response to OPTIONS method
- Add Access-Control-Allow-Headers, Access-Control-Allow-Methods, Access-Control-Allow-Origin Method Response Headers to OPTIONS method
- Add Access-Control-Allow-Headers, Access-Control-Allow-Methods, Access-Control-Allow-Origin Integration Response Header Mappings to OPTIONS method
- Add Access-Control-Allow-Origin Method Response Header to POST method
- Add Access-Control-Allow-Origin Integration Response Header Mapping to POST method

At the bottom right of the dialog are two buttons: 'Cancel' and a blue 'Yes, replace existing values' button.



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

The screenshot shows the AWS Lambda API Actions menu. The left sidebar lists resources like directie, directie1, Resources, Stages, Authorizers, etc. In the center, under 'Actions', there's a dropdown for '/{type} Methods'. A sub-menu is open for '/(type) METHODS' showing 'POST' details: 'None' for Authorization and 'API Key Required'. The 'API ACTIONS' section has 'Deploy API' highlighted. Other options include Create Method, Create Resource, Enable CORS, Edit Resource Documentation, Delete Resource, Import API, Edit API Documentation, and Delete API.

The 'Deploy API' dialog box contains fields for Deployment stage (set to [New Stage]), Stage name\*, Stage description, and Deployment description. At the bottom are 'Cancel' and 'Deploy' buttons.

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”

APIs      Usage Plans      Create

directie  
directie1  
LambdaSimpleProxy

**Usage Plans**

API Keys  
Custom Domain Names  
Client Certificates  
VPC Links  
Settings

directiePlan

Create Usage Plan

Usage Plans help you meter API usage. With Usage Plans, you can enforce a throttling and quota limit on each API key. Throttling limits define the maximum number of requests per second available to each key. Quota limits define the number of requests each API key is allowed to make over a period.

Name\* directie1Plan

Description

Throttling

Enable throttling  ⓘ

Rate\* 100 requests per second ⓘ

Burst\* 200 requests ⓘ

Quota

Enable quota  ⓘ

1500 requests per Month ⓘ

\* Required

Next

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

Usage Plans      Create

Associated API Stages

Associate API stages to this usage plan. Subscribers will only be allowed to access the API stages that are associated with the plan. Choose “Add API Stage” below, then use the dropdown to select an API and stage to enable for this usage plan.

Add API Stage

API	Stage	Method Throttling
directie1	directie1	<input checked="" type="checkbox"/> <input type="button" value="X"/>

Back Next



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

The screenshot shows the 'Usage Plan API Keys' page in the AWS CloudFormation console. The left sidebar shows 'Usage Plans' with 'directiePlan' selected. The main area has a heading 'Subscribe an API key to this usage plan. Choose "Add API Key" below to search through your existing API keys. Once a key is associated with a plan, API Gateway will meter all requests from the key and apply the plan's throttling and quota limits.' Below this are two buttons: 'Add API Key to Usage Plan' and 'Create API Key and add to Usage Plan'. A search bar labeled 'Name' contains the text 'directieKey (sgcdn...)'. To the right is a dropdown for 'Results per page' set to 100. At the bottom are navigation arrows ('< < Page 1 >'), a 'Back' button, and a prominent blue '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”

The screenshot shows the 'Create API Key' page in the AWS API Gateway console. The left sidebar lists 'APIs' (directie, directie1, LambdaSimpleProxy) and 'Usage Plans'. Under 'API Keys', 'API Keys' is selected. The main area has a 'Actions' dropdown open with options 'Create API key' and 'Import API keys'. The 'Create API Key' form includes fields for 'Name' (with placeholder 'eg. Customer name'), 'API key\*' (radio buttons for 'Auto Generate' and 'Custom'), and 'Description'. A note '\* Required' is shown below the form. At the bottom right is a blue 'Save' button.



## 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.firebaseio.google.com/> and click on Add project under Recent projects.

Welcome to Firebase!

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

[Learn more](#) [Documentation](#) [Support](#)

Recent projects

+ Add project

Explore a demo project

iOS

Add a project

Project name

My awesome project + iOS + </> Tip: Projects span apps across platforms

Project ID

my-awesome-project-id

Analytics and billing region

United States

Use the default settings for sharing Google Analytics for Firebase data

- ✓ Share your Analytics data with Google to improve Google Products and Services
- ✓ Share your Analytics data with Google to enable technical support
- ✓ Share your Analytics data with Google to enable Benchmarking
- ✓ Share your Analytics data with Google Account Specialists

I accept the [controller-controller terms](#). This is required when sharing Analytics data to improve Google Products and Services. [Learn more](#)

[Cancel](#) [Create project](#)

### 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.

The screenshot shows the Firebase console's Project Overview page for a project named 'FYP-PWA2'. The left sidebar has sections for Develop (Authentication, Database, Storage, Hosting, Functions, ML Kit), Quality (Crashlytics, Performance), and a general Project Overview. The main content area is titled 'Authentication' and includes tabs for 'Users', 'Sign-in method', 'Templates', and 'Usage'. A search bar at the top says 'Search by email address, phone number, or user UID'. Below it is a table with columns 'Identifier', 'Providers', 'Created', 'Signed In', and 'User UID'. A message 'No users for this project yet' is displayed. At the bottom right of the table is a blue 'Add user' button.

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

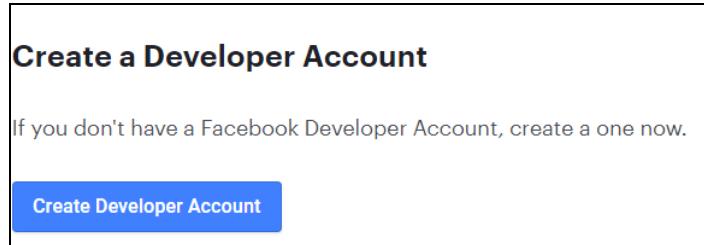
The screenshot shows the 'Sign-in method' configuration for Google. It features a 'Enable' toggle switch which is turned on. Below it, a note states: 'Google sign-in is automatically configured on your connected iOS and web apps. To set up Google sign-in for your Android apps, you need to add the SHA1 fingerprint for each app on your Project Settings.' A section titled 'Update the project-level setting below to continue' contains fields for 'Project public-facing name' (set to 'project-199589035245') and 'Project support email' (set to 'emailchuawei@gmail.com'). There are also dropdown menus for 'Whitelist client IDs from external projects (optional)' and 'Web SDK configuration'. At the bottom are 'Cancel' and 'Save' buttons.

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

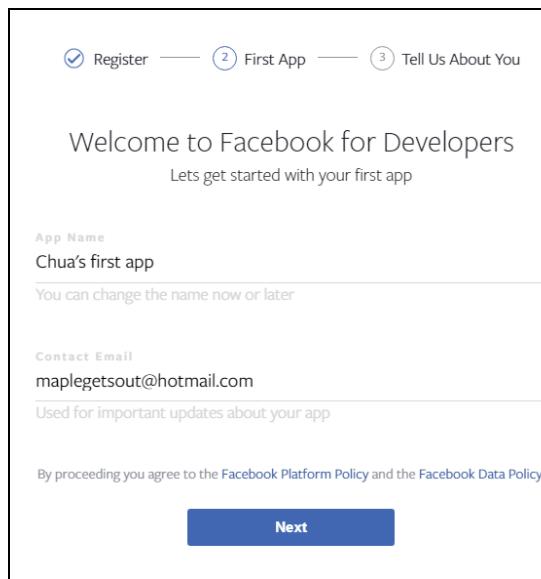
**Step 4** - For Facebook, if you already have an existing App ID and Secret, you may use it and add the OAuth redirect URI to the Facebook app configuration.



**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

Welcome to your Facebook for Developers Dashboard

Your app dashboard is where you can get an overview of your app, edit app settings, and configure new products.

Add Your First Product

**Facebook Login**

The world's number one social login product.

Read Docs

Set Up

**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.

Facebook

App ID  No Force Web OAuth Reauthentication  
When on, prompts people to enter their Facebook password in order to log in on the web. [?]

An app ID is required  Yes Use Strict Mode for Redirect URIs  
Only allow redirects that use the Facebook SDK or that exactly match the Valid OAuth Redirect URIs. Strongly recommended. [?]

App secret  Valid OAuth Redirect URIs  
An app secret is required  Valid OAuth redirect URIs.

To complete set up, add this OAuth redirect URI to your Facebook app configuration. [Learn more](#)

https://fyp-pwa2.firebaseioapp.com/\_/auth/handler

Cancel



**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.

The screenshot shows the Facebook Developer Settings page. On the left, there's a sidebar with options like Dashboard, Settings (which is selected), Advanced, Roles, Alerts, App Review, Products, Facebook Login, and Activity Log. The main area contains fields for app configuration:

App ID	5232	App Secret	74b2
Display Name	Chua's first app	Namespace	
App Domains		Contact Email	
Privacy Policy URL	Privacy policy for Login dialog and App Details	Terms of Service URL	Terms of Service for Login dialog and App Details
App Icon (1024 x 1024)		Category	

A 'Reset' button is located at the top right of the form.

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

The screenshot shows the Firebase Authentication Sign-in method screen. The left sidebar includes Develop, Authentication (selected), Database, Storage, Hosting, Functions, ML Kit, Quality, Crashlytics, Performance, Test Lab, and Analytics. At the bottom are Spark (free) and Upgrade buttons. The main area has tabs for Users, Sign-in method (selected), Templates, and Usage. Under 'Sign-in providers', there's a table:

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



## 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.

The screenshot shows the Firebase console interface. On the left is a sidebar with 'Project Overview', 'Develop' (selected), 'Authentication', 'Database' (selected), 'Storage', 'Hosting', 'Functions', and 'ML Kit'. Below that is a 'Quality' section. The main area has a title 'Or choose Realtime Database' above a diagram of three servers connected by lines. To the right is a box for 'Realtime Database' which says 'Firebase's original database. Like Cloud Firestore, it supports realtime data synchronization.' It includes links to 'View the docs' and 'Learn more', and a blue 'Create database' button.

**Step 2** - Click on Enable.

The screenshot shows a modal dialog titled 'Security rules for Realtime Database'. It contains instructions: 'Once you have defined your data structure you will have to write rules to secure your data.' with a 'Learn more' link. Two radio buttons are shown: 'Start in locked mode' (selected) and 'Start in test mode'. The 'locked mode' option is described as making the database private by denying all reads and writes. The 'test mode' option is described as getting set up quickly by allowing all reads and writes. To the right, the security rules code is displayed:

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

A note below states: 'All third party reads and writes will be denied'. At the bottom are 'Cancel' and 'Enable' buttons, with 'Enable' being highlighted in blue.



**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.

The screenshot shows the Firebase Realtime Database interface for project 'FYP-PWA2'. The left sidebar has sections for Develop (Authentication, Database, Storage, Hosting, Functions, ML Kit) and Quality (Crashlytics, Performance, Test Lab). The main area shows a single node 'fyp-pwa2' with the value 'null'. A context menu is open on the right, with 'Import JSON' being the selected option. Other options in the menu include 'Export JSON', 'Show legend', 'Disable database', and 'Create new database'.

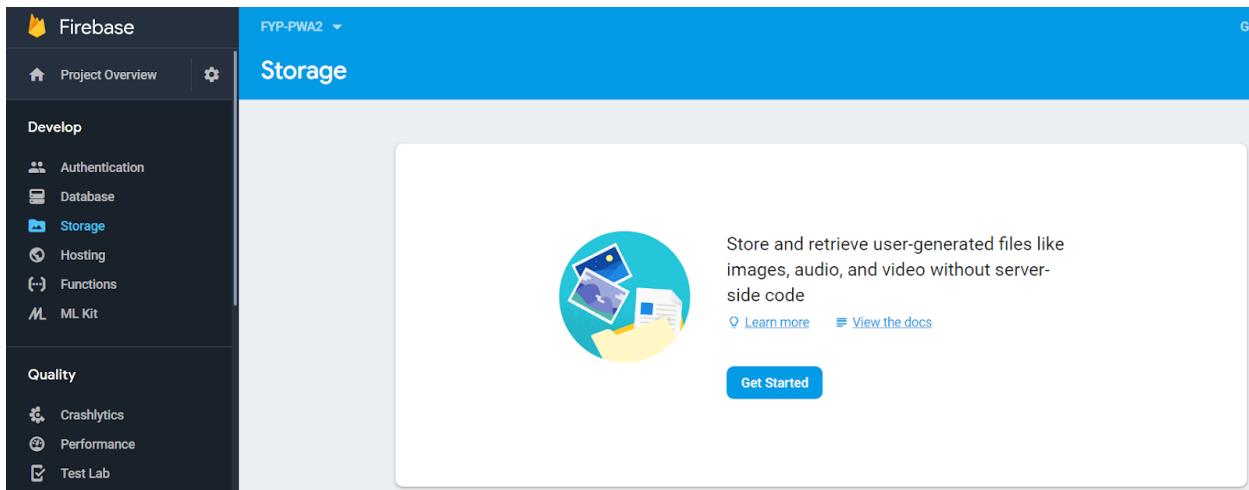
**Step 4** - You should see something like this once you have imported successfully. Although you cannot see the structure of the database, you can see it once a route has been completed/failed, or when a user has been created.

The screenshot shows the Firebase Realtime Database interface for project 'FYP-PWA2'. The left sidebar is identical to the previous screenshot. The main area now shows a node 'fyp-pwa2' with a child node 'Counter' containing the value '0'. The interface includes standard database management tools like add, edit, and delete icons.



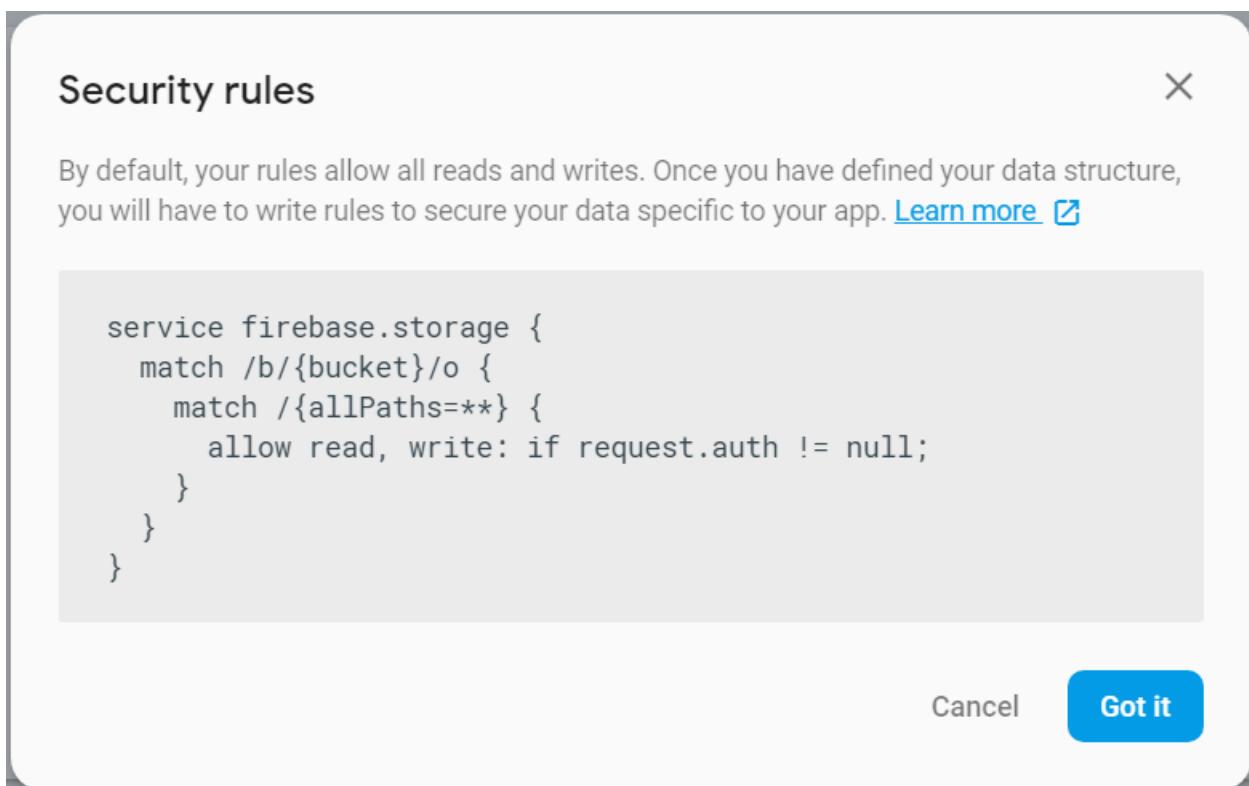
## 2.3 Setting up Firebase Storage

**Step 1** - Navigate to Storage under the menu and click on 'Get Started'.



The screenshot shows the Firebase console interface. On the left, there's a sidebar with 'Develop' and 'Quality' sections. Under 'Develop', 'Storage' is selected. The main area is titled 'Storage' and contains a large circular icon with a document and a photo. Below the icon, the text reads: 'Store and retrieve user-generated files like images, audio, and video without server-side code'. It includes links to 'Learn more' and 'View the docs'. A blue 'Get Started' button is at the bottom right of the main area.

**Step 2** - A pop-up will appear. Click on 'Got it'.



The pop-up window has a title 'Security rules' and a close button 'X' in the top right corner. Inside, it says: 'By default, your rules allow all reads and writes. Once you have defined your data structure, you will have to write rules to secure your data specific to your app.' It includes a 'Learn more' link. Below this is a code block showing Firebase security rules:

```
service firebase.storage {  
  match /b/{bucket}/o {  
    match /{allPaths=**} {  
      allow read, write: if request.auth != null;  
    }  
  }  
}
```

At the bottom right of the pop-up are 'Cancel' and 'Got it' buttons, with 'Got it' being highlighted in blue.



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

The screenshot shows the Firebase Storage console interface. At the top, there's a URL bar with 'gs://fyp-pwa2.appspot.com'. To the right of the URL is an 'Upload file' button with an upward arrow icon. Below the URL bar is a header row with columns for 'Name', 'Size', 'Type', and 'Last modified'. A message at the bottom of the list states: 'Default security rules require users to be authenticated' with 'Learn more' and 'Dismiss' links. The main content area below the header shows a message: 'There are no files here yet'.

**Step 4** - Make a directory something like this:

images/  
images/facilities/  
images/ProfilePicture/  
images/units/

Your root folder will have something like this:

The screenshot shows the Firebase Storage console interface. At the top, there's a URL bar with 'gs://pwa-firebase-hosting.appspot.com'. To the right of the URL is an 'Upload file' button with an upward arrow icon. Below the URL bar is a header row with columns for 'Name', 'Size', 'Type', and 'Last modified'. Under the 'Name' column, there is a single entry: 'images/' which is a folder. The 'Type' column indicates it is a 'Folder'.

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.

```
Command Prompt
Microsoft Windows [Version 10.0.17134.165]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\$Users\$Index>npm install -g firebase-tools
```

**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.

```
Command Prompt
-
X
C:\$Users\$Index>npm install -g firebase-tools
C:\$Users\$Index\$AppData\$Roaming\$npm\$firebase -> C:\$Users\$Index\$AppData\$Roaming\$npm\$node_modules\$firebase-tools\$bin\$fireba
se
+ firebase-tools@4.0.2
updated 1 package in 4.767s
C:\$Users\$Index>firebase login
```

**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.

```
D:\Users\Index\Documents\GitHub\firebase2>firebase init
#####
##  ## ###### ## ###### ###### ## ## ###### #####
##### ## ##### ## ##### ###### ##### ## #####
## ## ## ## ## ## ## ## ## ## ## ## ## ## #####
## ##### ## ##### ###### ##### ## #####
You're about to initialize a Firebase project in this directory:
D:\Users\Index\Documents\GitHub\firebase2
? Are you ready to proceed? Yes
? 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
```

**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!

```
? Select a default Firebase project for this directory: FYP-PWA2 (fyp-pwa2)
== 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? JavaScript
? Do you want to use ESLint to catch probable bugs and enforce style? No
+ Wrote functions/package.json
+ Wrote functions/index.js
? Do you want to install dependencies with npm now? No
== Hosting Setup
Your public directory is the folder (relative to your project directory) that
will contain Hosting assets to be uploaded with firebase deploy. If you
have a build process for your assets, use your build's output directory.
? What do you want to use as your public directory? public
? Configure as a single-page app (rewrite all urls to /index.html)? No
+ Wrote public/404.html
+ Wrote public/index.html
i Writing configuration info to firebase.json...
i Writing project information to .firebaserc...
+ Firebase initialization complete!
```



**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.

```
Administrator: Command Prompt
[grpc] Success: "D:\Users\Index\Documents\GitHub\firebase\functions\node_modules\grpc\src\node\Extension_binary\node-v57-win32-x64-unknown\grpc_node.node" is installed via remote

> protobufjs@6.8.8 postinstall D:\Users\Index\Documents\GitHub\firebase\functions\node_modules\google-gax\node_modules\protobufjs
> node scripts/postinstall

> protobufjs@6.8.8 postinstall D:\Users\Index\Documents\GitHub\firebase\functions\node_modules\google-proto-files\node_modules\protobufjs
> node scripts/postinstall

> firebase-functions@2.0.2 postinstall D:\Users\Index\Documents\GitHub\firebase\functions\node_modules\firebase-functions
> node ./upgrade-warning

===== WARNING! =====

This upgrade of firebase-functions contains breaking changes if you are upgrading from a version below v1.0.0.

To see a complete list of these breaking changes, please go to:
https://firebase.google.com/docs/functions/beta-v1-diff

added 541 packages in 34.187s

D:\Users\Index\Documents\GitHub\firebase\functions>
```



## 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!

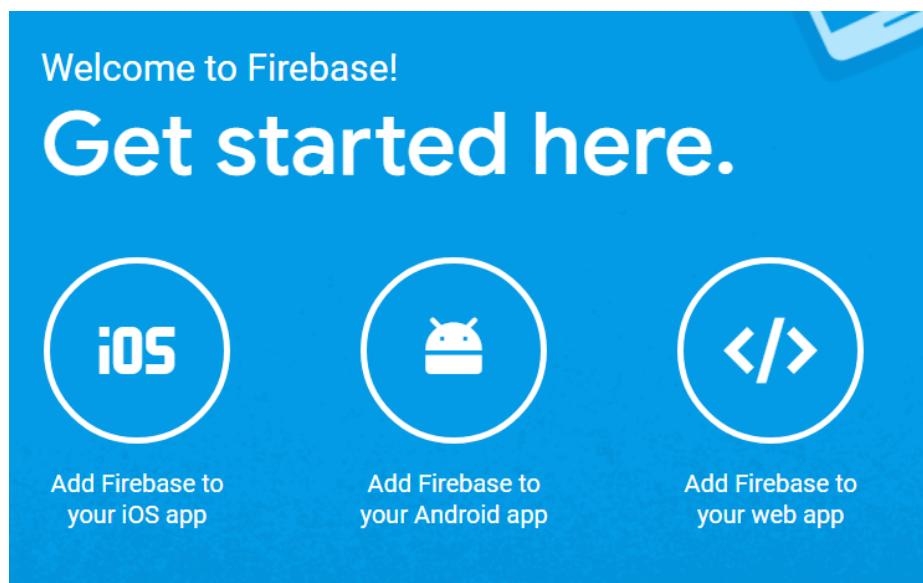
Project Console: https://console.firebaseio.google.com/project/pwa-firebase-hosting/overview
Hosting URL: https://pwa-firebase-hosting.firebaseioapp.com
```



## 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.firebaseio.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: "AIzaSyCnrvmbLM",
    authDomain: "fyp-com",
    databaseURL: "https://fyp-.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.

#### Service accounts for project "FYP-PWA2"

A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App Engine apps, or systems running outside Google. [Learn more](#)

<input type="checkbox"/>	Email	Name	Key ID	Delete key	Key creation date	Actions
<input type="checkbox"/>	fyp-pwa2@appspot.gserviceaccount.com	App Engine default service account	No keys			

**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: "AIzaXXXXXXXXXXGa-CmBHidWrOGV2RSBI",
    authDomain: "pwa-firebase.firebaseio.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.firebaseio.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-sln4-999999999.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:

The screenshot shows the Directie mobile application interface. On the left is a vertical navigation menu with a user icon and the word "Admin" at the top, followed by "Home", "Compass", "Edit Profile", and "Log out". To the right of the menu is a large dark blue and green area representing the indoor environment. On the far right is a white configuration panel titled "compass bearing". It contains four dropdown menus: "Previous Beacon" (set to "Please select a beacon"), "Current Beacon" (set to "Please select a beacon"), "Next Beacon" (set to "Please select a beacon"), and "Direction" (set to "Please select a direction"). Below these is a text input field labeled "Compass Bearing" containing the value "113". A small "Note" section is present below the bearing value. At the bottom of the panel is a blue button labeled "SEND COMPASS BEARING".

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.