

Software Studio

軟體設計與實驗

Server & Firebase

Hung-Kuo Chu

Department of Computer Science
National Tsing Hua University

CS2410



Codeblock Conventions

HTML5 Program

JavaScript Program

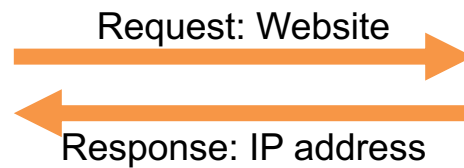
JSON or Rule



Basic Web Workflow



Client



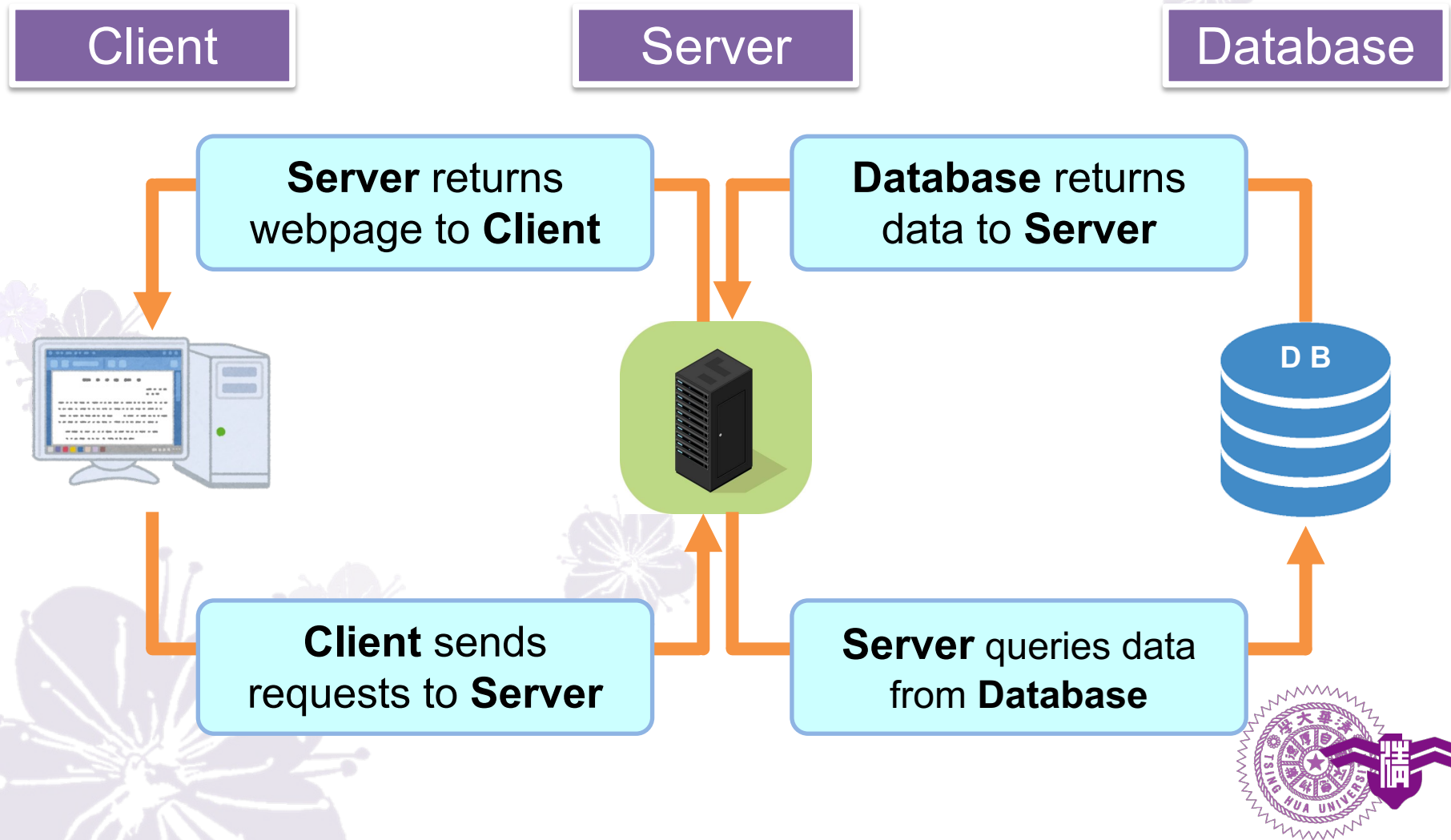
DNS Server



Main Server



Web Architecture



Client

- **Client (Frontend)**

- Send requests to Server.
- Process/Display data received from Server via Web Browser.
- HTML + CSS + Javascript can build a good interactive website
- We cannot keep any data
- We cannot collect data



Server

- **Server (backend)**
 - save data in server
 - do some private function
 - collect data and display to client
- **Database**
 - save some txt data
- **Storage**
 - save some multimedia data



What is Firestore?

- Firestore is an application cloud development platform built by Google.
- Help developers **quickly set up** backend services in the cloud which effectively shorten the application development time.



Firestore



Why Use Firebase?

- A powerful **real-time database** which makes it an excellent candidate to drive **multi-player games**.
- Full document storage, analytics, hosting, etc.
- Using **JavaScript** API and provide user side high security.



Who are Using Firebase?

The New York Times



venmo

The Economist

trivago



wattpad





What Will You Learn?

- Firebase has a lot of features and we will teach you





Hosting
</>





Realtime Database
iOS  </> C++ 



Cloud Storage
iOS  </> C++ 



Authentication
iOS  </> C++ 



Cloud Functions
iOS  </> C++ 



Step by step

1. Create a new Firebase Project
2. Create a new Web App on Firebase Console
3. Activate Firebase Features (Authentication, Realtime Database, Hosting, Storage)
4. Download React App
5. Import Firebase SDK
6. Install Required Packages
7. Firebase initialization
8. Run Your Webpage



1. Create a new Firebase Project

- Go to [Firebase Console](#) (need a Google account) and add a project

× 建立專案(步驟 3 之 1)

請先輸入您的 專案名稱[®]

專案名稱

Software Studio

software-studio-a2c21

☒ 我接受 [Firebase 條款](#)

繼續

This id will be used in database URL and webpage domain.
Please record it!



1. Create a new Firebase Project

- Then finish remaining steps.

× 建立專案(步驟 3 之 2)

適用於 Firebase 專案的 Google Analytics (分析)

Google Analytics (分析) 是一項沒有用量限制的免費資料分析解決方案，能讓您在使用 Firebase Crashlytics、雲端通訊、應用程式內通訊、遠端設定、A/B 測試、預測和 Cloud Functions 等功能時，執行指定目標和查看報表等動作。

Google Analytics (分析) 提供以下功能：

- A/B 測試 ①
- 所有 Firebase 產品中的使用者區隔和指定目標功能 ②
- 預測使用者行為 ③
- 未受當機情況影響的使用者 ④
- 以事件為基礎的 Cloud Functions 觸發條件 ⑤
- 沒有用量限制的免費報表功能 ⑥

☒ 啟用這項專案的 Google Analytics (分析) 功能建議選項

上一步 繼續

× 建立專案(步驟 3 之 3)

設定 Google Analytics (分析)

數據分析位置 ①

美國

資料共用設定和 Google Analytics (分析) 條款

- ☒ 使用預設的 Google Analytics (分析) 資料共用設定。 [Learn more](#)
 - ✓ 將 Analytics (分析) 資料提供給 Google，協助我們改善產品和服務
 - ✓ 將 Analytics (分析) 資料提供給 Google，以便啟用基礎化功能
 - ✓ 提供 Analytics (分析) 資料給 Google 以取得技術支援
 - ✓ 將 Analytics (分析) 資料提供給 Google 帳戶專家
- ☒ 我接受《[評估服務控管者與控管者的資料保護條款](#)》，並確認本人必須遵守《[歐盟地區使用者同意授權政策](#)》。如想提供 Google Analytics (分析) 資料來改善 Google 產品和服務，您必須勾選這個核取方塊。 [瞭解詳情](#)
- ☒ 我接受 [Google Analytics \(分析\) 的條款](#)

建立專案時，系統會建立新的 Google Analytics (分析) 資源，並將該資源連結至您的 Firebase 專案。連結完成後，資料就能在產品之間流通，從 Google Analytics (分析) 資源匯出到 Firebase 的資料需符合 Firebase《[服務條款](#)》的規定；而匯入 Google Analytics (分析) 的 Firebase 資料則需符合 Google Analytics (分析)《[服務條款](#)》的規定。 [瞭解詳情](#)。

上一步 建立專案


2. Create a new Web App on Firebase Console



Initialization Configuration

× 將 Firebase 新增至您的網路應用程式

1 註冊應用程式

應用程式暱稱 

SoftwareStudioExample-1

Pick a name for your application

☐ 一併為這個應用程式設定 **Firebase 代管功能**。瞭解詳情 

您也可以之後再設定代管功能。您隨時可以開始免費使用這項服務。

註冊應用程式

2 新增 Firebase SDK



Initialization Configuration

2 新增 Firebase SDK

☒ 使用 npm  ☐ 使用 <script> 標記 

如果你已使用 [npm](#) 和 [webpack](#) 或 [Rollup](#) 等模組整合工具，則可執行下列指令來安裝最新版 SDK：

```
$ npm install firebase
```



請初始化 Firebase，接著即可開始將 SDK 套用至要使用的產品。

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
const firebaseConfig = {
```

注意事項：這個選項會使用[模組 JavaScript SDK](#)，因此 SDK 的大小得以縮減。

如要進一步瞭解適用於網頁應用程式的 Firebase，請查看下列資源：[開始使用](#)、[Web SDK API 參考資料](#)、[使用範例](#)

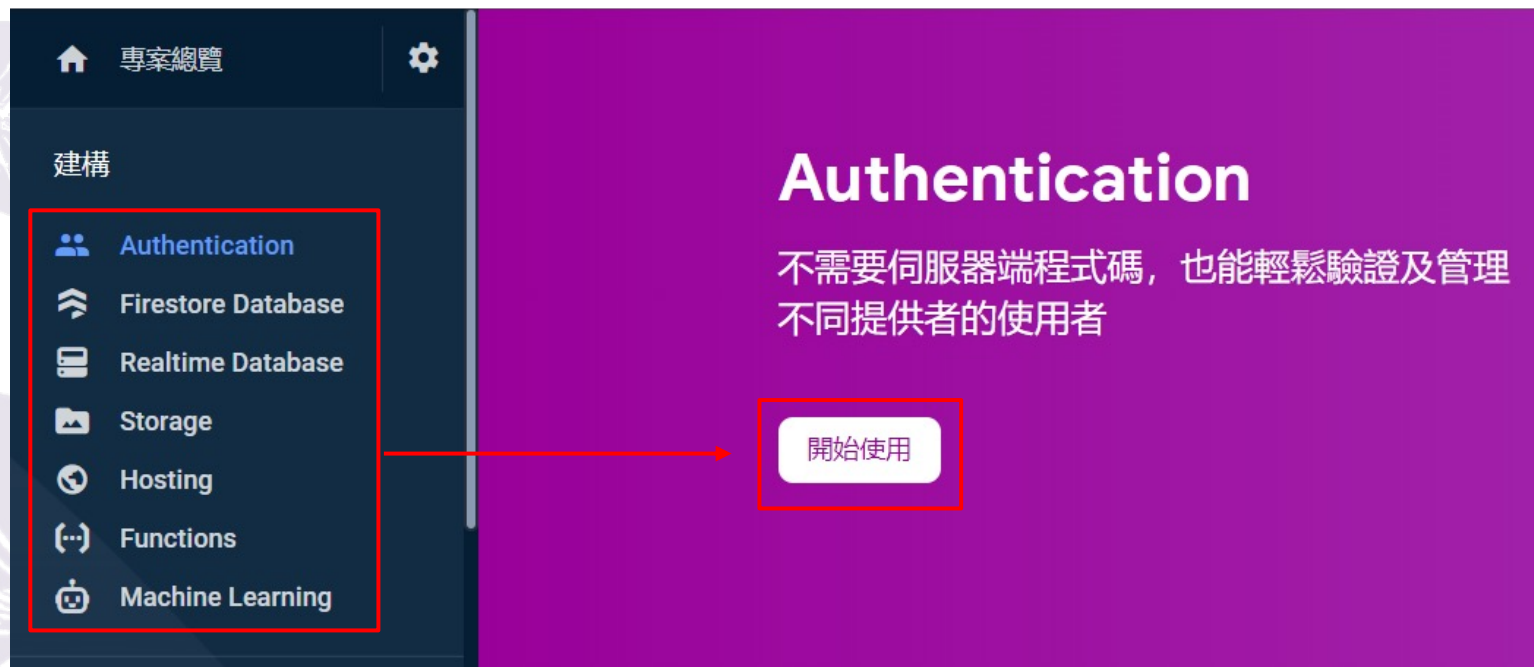
前往控制台

We will import Firebase SDK later



3. Activate Firebase Features

- Activate the firebase features you want to use one by one. (Authentication, Database, Hosting...)



3. Activate Firebase Features

Hosting

Deploy web and mobile web apps in seconds using a secure global content delivery network

Get started

1 Install Firebase CLI

To host your site with Firebase Hosting, you need the Firebase CLI (a command line tool).

Run the following [npm](#) command to install the CLI or update to the latest CLI version.

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

Doesn't work? Take a look at the [Firebase CLI reference](#) or change your [npm permissions](#)

- ☐ Also show me the steps to add the Firebase JavaScript SDK to my web app
The SDK includes Cloud Firestore, Authentication, Performance Monitoring and more. It can be added now or later.

Next

3. Activate Firebase Features

2 Initialize your project

Open a terminal window and navigate to or create a root directory for your web app

Sign in to Google

```
$ firebase login
```



Initiate your project

Run this command from your app's root directory:

```
$ firebase init
```

Next

3 Deploy to Firebase Hosting

When you're ready, deploy your web app

Put your static files (e.g., HTML, CSS, JS) in your app's deploy directory (the default is "public"). Then, run this command from your app's root directory:

```
$ firebase deploy
```



After deploying, view your app at testing-3b6e2.web.app

Need help? Check out the [Hosting docs](#)

Continue to console



3. Activate Firebase Features

Authentication

Authenticate and manage users from a variety of providers without server-side code

Get started

Sign-in providers

Get started with Firebase Auth by adding your first sign-in method

Native providers

Email/Password

Phone

Anonymous

Additional providers

Google

Game Center

Microsoft

Facebook

Apple

Twitter

Play Games

GitHub

Yahoo

Custom providers

OpenID Connect

SAML

We'll be back later



3. Activate Firebase Features

Realtime Database

Store and sync data in real time

Create Database

Set up database



Database options

2

Security rules

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

[Learn more](#)



Start in locked mode

Your data is private by default. Client read/write access will only be granted as specified by your security rules.



Start in test mode

Your data is open by default to enable quick setup. However, you must update your security rules within 30 days to enable long-term client read/write access.

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



All third party reads and writes will be denied

Cancel

Enable

3. Activate Firebase Features

Storage

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

Get started

Set up Cloud Storage

- 1 Secure rules for Cloud Storage
- 2 Set Cloud Storage location

After you define your data structure, you will need to write rules to secure your data.

[Learn more](#)

☒ Start in **production mode**

Your data is private by default. Client read/write access will only be granted as specified by your security rules.

☐ Start in **test mode**

Your data is open by default to enable quick setup. However, you must update your security rules within 30 days to enable long-term client read/write access.

```
rules_version = '2';
service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      allow read, write: if false;
    }
  }
}
```

i All third party reads and writes will be denied

Next

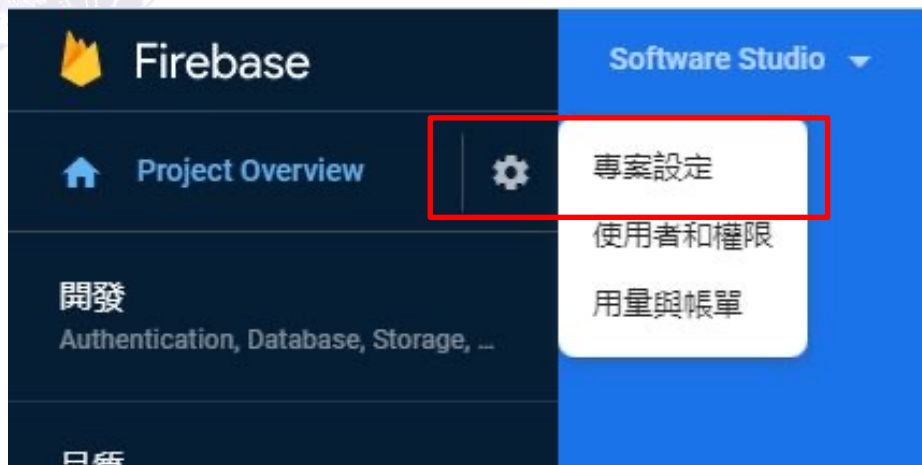
4. Download React App

- Use the link below to download the zip file
https://drive.google.com/file/d/1OVIjUzIkf4NgPneC-0UYybeWwPR7oisA/view?usp=drive_link
- Extract the zip file



5. Import Firebase SDK

- Go to Settings to get the firebaseConfig.



SDK 設定和配置

☒ npm [?] ☐ CDN [?] ☐ 設定 [?]

如果你已使用 [npm](#) 和 [webpack](#) 或 [Rollup](#) 等模組整合工具，則可執行下列指令來安裝最新版 SDK：

```
$ npm install firebase
```

請初始化 Firebase，接著即可開始將 SDK 套用至要使用的產品。

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
const firebaseConfig = {
```



5. Import Firebase SDK

- Add Firebase Config to your **config.js**
- **Make sure there is databaseURL.**

```
// Initialize Firebase
// TODO: Replace with your project's customized code snippet
const firebaseConfig = {
  apiKey: "<API_KEY>",
  authDomain: "<PROJECT_ID>.firebaseapp.com",
  databaseURL: "https://<DATABASE_NAME>.firebaseio.com",
  projectId: "<PROJECT_ID>",
  storageBucket: "< PROJECT_ID >.appspot.com",
  messagingSenderId: "<SENDER_ID>",
  appId: "<APPLICATION_ID>",
  measurementId: "<AnalyticsSDK>"
};
const app = initializeApp(firebaseConfig);
```



5. Import Firebase SDK

- If databaseURL is missing. You can find it in firebase console.

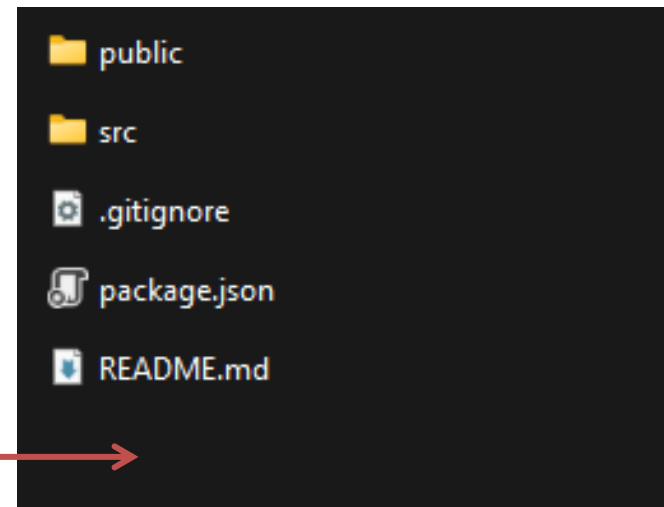


6. Install the Required Packages

- Download [node.js](#), 16.0.0 LTS or higher
- Run the command at your project directory and complete all settings

1. `npm install`
2. `npm run build`
3. `firebase login`
4. `firebase init`

Run at this directory!!



“`npm install`” will install all packages in `package.json` file



7. Firebase Initialization

- Press <space> to select Firebase features.

```
? Are you ready to proceed? Yes
? Which Firebase features do you want to set up for this directory? Press
  space to select, <a> to toggle all, <i> to invert selection, <enter> to
  proceed)
  ( ) Functions: Configure a Cloud Functions directory and its files
  (*) Hosting: Configure files for Firebase Hosting and (optionally) set
  up GitHub Action deploys
  ( ) Hosting: Set up GitHub Action deploys
> (*) Storage: Configure a security rules file for Cloud Storage
  ( ) Emulators: Set up local emulators for Firebase products
  ( ) Remote Config: Configure a template file for Remote Config
  (*) Realtime Database: Configure a security rules file for Realtime Database
(Move up and down to reveal more choices)
```



7. Firebase Initialization

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

- On default Firebase project selection, select your project
 - If you could not see the project you just create on firebase's console, you can directly add after initialization, so just select [don't setup a default project]
 - Run “**firebase projects:list**” and select one **ID**
 - Use “**firebase use [ID]**” to link your project



7. Firebase Initialization

- Use default setting in other selections
- Notice that if your application wants to use any firebase features, it need to be activated first. Otherwise, you can't set it in ***firebase init.***



7. Firebase Initialization

=== Project Setup

First, let's associate this project directory with a Firebase project. You can create multiple project aliases by running **firebase use --add**, but for now we'll just set up a default project.

```
? Please select an option: Use an existing project
? Select a default Firebase project for this directory: fir-react-4bb57 (FirebaseReact)
+ database: required API firebase.database.googleapis.com is enabled
```

Firebase Realtime Database Security Rules allow you to define how your data should be structured and when your data can be read from and written to.

```
? What file should be used for Realtime Database Security Rules? database.rules.json
+ Database Rules for fir-react-4bb57-default-rtdb have been written to database.rules.json.
Future modifications to database.rules.json will update Realtime Database Security Rules when you run
firebase deploy.
```



7. Firebase Initialization

=== 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? build
? Configure as a single-page app (rewrite all urls to /index.html)? Yes
? Set up automatic builds and deploys with GitHub? No
? File build/index.html already exists. Overwrite? No
i Skipping write of build/index.html
```

If you accidentally choose “**public**” as public directory, you can change it later in **firebase.json** file after completing initialization process. Details in next slide.



7. Firebase Initialization

firebase.json will show up
after initialization

Change **public** → **build**

```
> build
> node_modules
> public
> src
🔥 .firebaserc
📄 .gitignore
{} database.rules.json
🔥 firebase.json
{} package-lock.json
{} package.json
📄 README.md
📄 storage.rules
```

```
firebase.json ×
firebase.json > ...
1  {
2    "database": {
3      "rules": "database.rules.json"
4    },
5    "hosting": {
6      "public": "build",
7      "ignore": [
8        "firebase.json",
9        "**/.*",
10       "**/node_modules/**"
11     ],
12     "rewrites": [
13       {
14         "source": "**",
15         "destination": "/index.html"
16       }
17     ]
18   },
```

7. Firebase Initialization

=== Storage Setup

Firebase Storage Security Rules allow you to define how and when to allow uploads and downloads. You can keep these rules in your project directory and publish them with **firebase deploy**.

? What file should be used for Storage Rules? `storage.rules`

+ Wrote `storage.rules`

i Writing configuration info to `firebase.json`...

i Writing project information to `.firebaserc`...

+ Firebase initialization complete!



8. Run Your Webpage

- You can run a local server for testing
- Run the command at your project directory

– firebase serve

– npm start

Both have the
same function

- Your webpage will run on

<http://localhost:5000> or

<http://localhost:3000>

Firestore Example

[Authentication](#)

[Realtime Database](#)

[Cloud Storage](#)

Your webpage should
look like this



8. Run Your Webpage

- You can also deploy it using Firebase Hosting and obtain its link by running “**firebase deploy**” command.

+ Deploy complete!

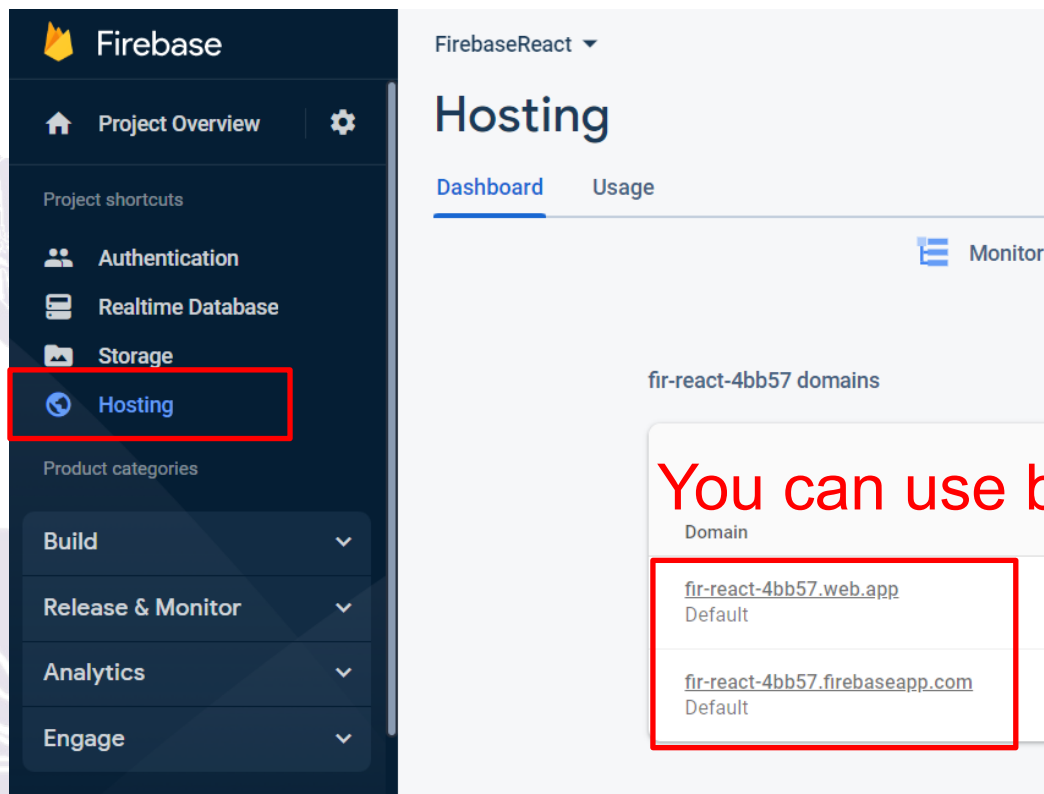
Project Console: <https://console.firebase.google.com/project/fir-react-4bb57/overview>

Hosting URL: <https://fir-react-4bb57.web.app>



8. Run Your Webpage

- Or you can obtain the link in firebase console.



**Take a
Break!**



Hosting

- Firebase Hosting has some key capabilities
 - Served over a secure connection
 - Fast content delivery
 - Rapid deployment
 - One-click rollbacks



Hosting

- Run the command at your project directory
 - `firebase deploy`
- Your app will be deployed to the domain <YOUR-FIREBASE-APP>.firebaseapp.com



Authentication

- You can use Firebase Authentication to allow users to sign in your app using one or more sign-in methods

新增第一種登入方式，即可開始使用 Firebase 驗證


原生供應商


✉ 電子郵件/密碼

☎ 電信業者

👤 匿名


其他供應商


 Google


 Game Center

 Microsoft


 Facebook

 Apple

 Twitter

 Play 遊戲

 GitHub

 Yahoo

Authentication

1. Add your web page domain to Authorized Domains. (You can skip this step if you are using firebase hosting.)
2. Enable the sign-in method
3. Implement authentication with `firebase.auth()`



Authorized Domains

- Before adding Firebase Authorized to your web page, you need to add your web page domain in firebase console.
- For example, your gitlab website url.

Authentication

Users

Sign-in method

Templates

Usage

授權網域 ⓘ

新增網域

已授權網域

類型

localhost

Default

softwarestudio-32b32.firebaseio.com

Default

softwarestudio-32b32.web.app

Default

Email/Password Sign-In

- Enable the Email/password sign-in method

The screenshot shows the 'Authentication' settings page. The 'Sign-in method' tab is selected and highlighted with a red box. A red arrow points from this tab to the '新增供應商' (Add Provider) button, which is also highlighted with a red box. Another red arrow points from the '新增供應商' button to the '電子郵件/密碼' (Email/Password) provider entry. In this entry, the toggle switch is turned on and highlighted with a red box. Below this, there is a description of the provider and another toggle switch for '電子郵件連結' (Email Link), which is also turned on.

Authentication

Users **Sign-in method** Templates Usage

登入供應商

新增供應商

識別資訊提供者

- ✉ 電子郵件/密碼
- Google

✉ 電子郵件/密碼

可讓使用者以自己的電子郵件地址和密碼註冊。我們的 SDK 也提供電子郵件地址驗證、密碼救援和電子郵件地址變更等基本功能。[瞭解詳情](#)

電子郵件連結 (不需要密碼即可登入)

Email/Password Sign-Up

- Sign up new users

```
import { getAuth, createUserWithEmailAndPassword } from "firebase/auth";

const auth = getAuth();
createUserWithEmailAndPassword(auth, email, password)
  .then((userCredential) => {
    // Signed in
    const user = userCredential.user;
    // ...
  })
  .catch((error) => {
    const errorCode = error.code;
    const errorMessage = error.message;
    // ..
  });
```



Email/Password Sign-In

- Sign in existing users

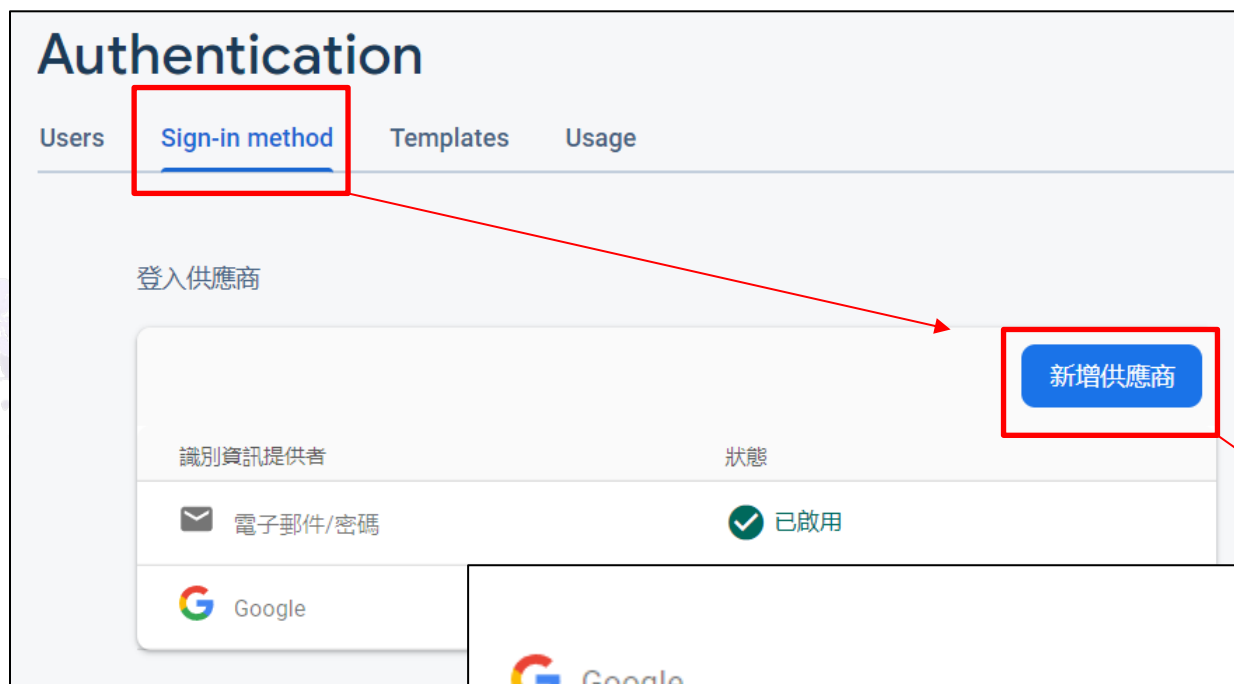
```
import { getAuth, signInWithEmailAndPassword } from "firebase/auth";

const auth = getAuth();
signInWithEmailAndPassword(auth, email, password)
  .then((userCredential) => {
    // Signed in
    const user = userCredential.user;
    // ...
  })
  .catch((error) => {
    const errorCode = error.code;
    const errorMessage = error.message;
  });
```



Google Sign-In

- Enable the Google sign-in method



Google

系統會自動為已連結的 Apple 和網頁應用程式設定 Google 登入機制。如要為 Android 應用程式設定 Google 登入機制，你必須在[專案設定](#)中為每個應用程式新增 [SHA1 指紋](#)。

Google Sign-In

- Create an instance of the Google provider object

```
import { GoogleAuthProvider } from "firebase/auth";  
const provider = new GoogleAuthProvider();
```



Google Sign-In

- Sign in with a pop-up window

```
import { getAuth, signInWithPopup, GoogleAuthProvider } from "firebase/auth";

const auth = getAuth();
signInWithPopup(auth, provider)
  .then((result) => {
    // The signed-in user info.
    const user = result.user;
  }).catch((error) => {
    // Handle Errors here.
    const errorCode = error.code;
    const errorMessage = error.message;
    // The email of the user's account used.
    const email = error.customData.email;
    // The AuthCredential type that was used.
    const credential = GoogleAuthProvider.credentialFromError(error);
  });
```

Google Sign-In

- Sign in by redirecting to the sign-in page

```
import { getAuth, getRedirectResult, GoogleAuthProvider } from "firebase/auth";

const auth = getAuth();
getRedirectResult(auth)
  .then((result) => {
    // The signed-in user info.
    const user = result.user;
    // ...
  }).catch((error) => {
    // Handle Errors here.
    const errorCode = error.code;
    const errorMessage = error.message;
    // The email of the user's account used.
    const email = error.customData.email;
    // The AuthCredential type that was used.
    const credential = GoogleAuthProvider.credentialFromError(error);
  });
```

Facebook Sign-In

- On the [Facebook for Developers](#) site, create a new App



The screenshot displays the Facebook for Developers website. On the left, there is a photograph of a group of people in a workshop setting. The main content area features the 'Facebook for Startups' banner with the text '參加 Facebook 新創公司計畫 助您打造優質的產品並拓展您 創公司規模。' and a '深入瞭解' button. On the right, a navigation menu is visible with the option '建立應用程式' (Create New App) highlighted by a red rectangular box. Other menu items include '要求', '開發人員設定', '公司設定', and '登出 Facebook'. At the bottom, there are three news items: 'Introducing Early Testing of Data Use Checkup', 'Winners of our first Facebook Online Hackathon announced', and 'Pausing Individual Verification', each with a '瞭解詳情' (Learn More) link.

facebook for developers

產品 文件 更多 ▾ 我的應用程式 ▾

Facebook for Startups

參加 Facebook 新創公司計畫
助您打造優質的產品並拓展您
創公司規模。

深入瞭解

建立應用程式

要求
開發人員設定
公司設定
登出 Facebook

2020 年 F8 大會重要更新：鑑於 2019 年新型冠狀病毒 (COVID-19) 疫情持續擴大，我們決定取消今年 F8 大會的現場實體活動。更多內容

Introducing Early Testing of Data Use Checkup
瞭解詳情

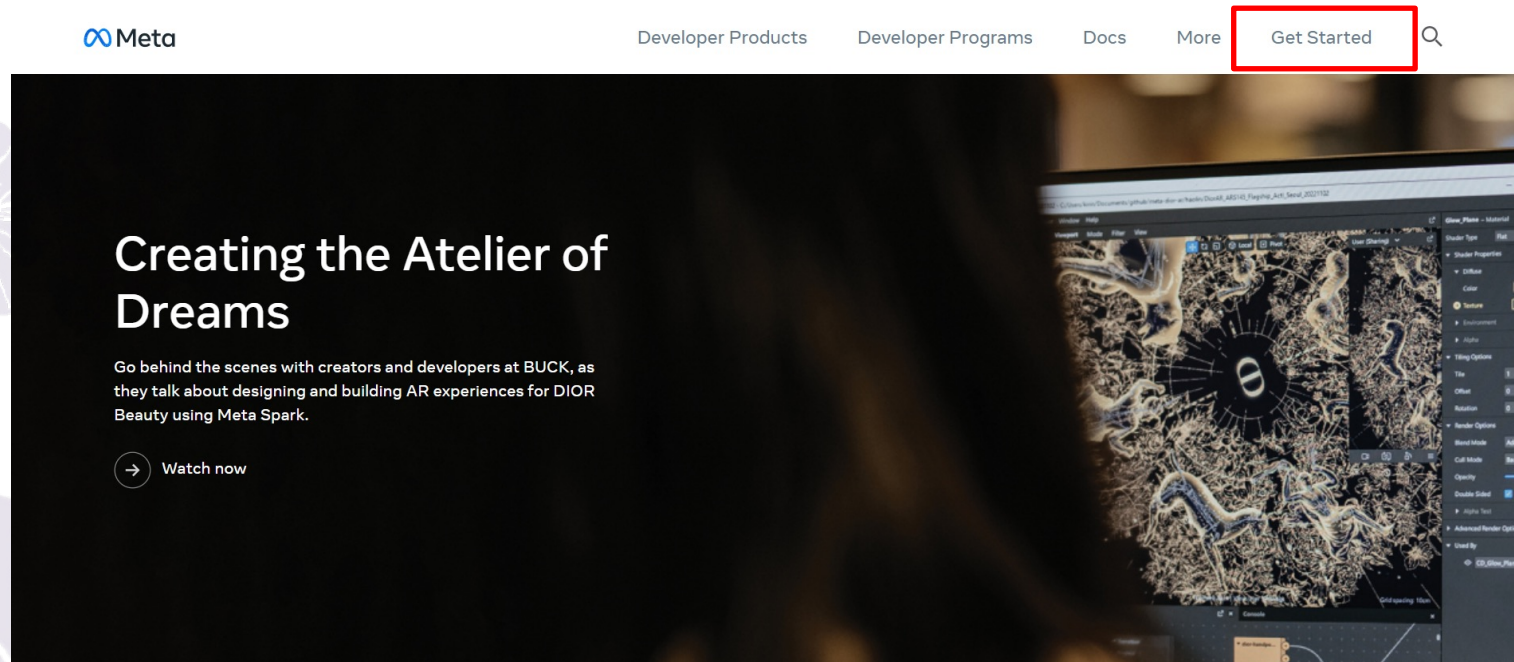
Winners of our first Facebook Online Hackathon announced
瞭解詳情

Pausing Individual Verification
瞭解詳情



Facebook Sign-In

- On the [Facebook for Developers](#) site, create a new App



Facebook Sign-In

- Get the **App ID** and an **App Secret** for your app

The screenshot shows the Facebook Developer console interface. At the top, there's a header with 'Software Studio 2018' and 'APP ID: [redacted]'. A status bar indicates 'Status: In Development' with a toggle switch set to 'OFF' and a 'View Analytics' button. A blue banner message states: 'We are not reviewing apps at this time due to changes we are making to the Facebook Platform. [Learn more](#) ×'. The left sidebar contains navigation links: Dashboard, Settings (Basic, Advanced), Roles, Alerts, App Review, and PRODUCTS (+). The main content area displays app details for 'Software Studio 2018'. A red rectangular box highlights the 'App ID' and 'App Secret' fields, which are both redacted. Other visible fields include 'Display Name' (Software Studio 2018), 'Namespace', 'App Domains', 'Contact Email' (roger2200@gmail.com), '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)' (placeholder image), and 'Category' (Choose a Category dropdown). At the bottom, there's a '+ Add Platform' button and the 'facebook for developers' logo.

(in Facebook for Developer)



Facebook Sign-In

- Enable the Facebook sign-in method

Authentication

USERS SIGN-IN METHOD TEMPLATES USAGE

Sign-in providers

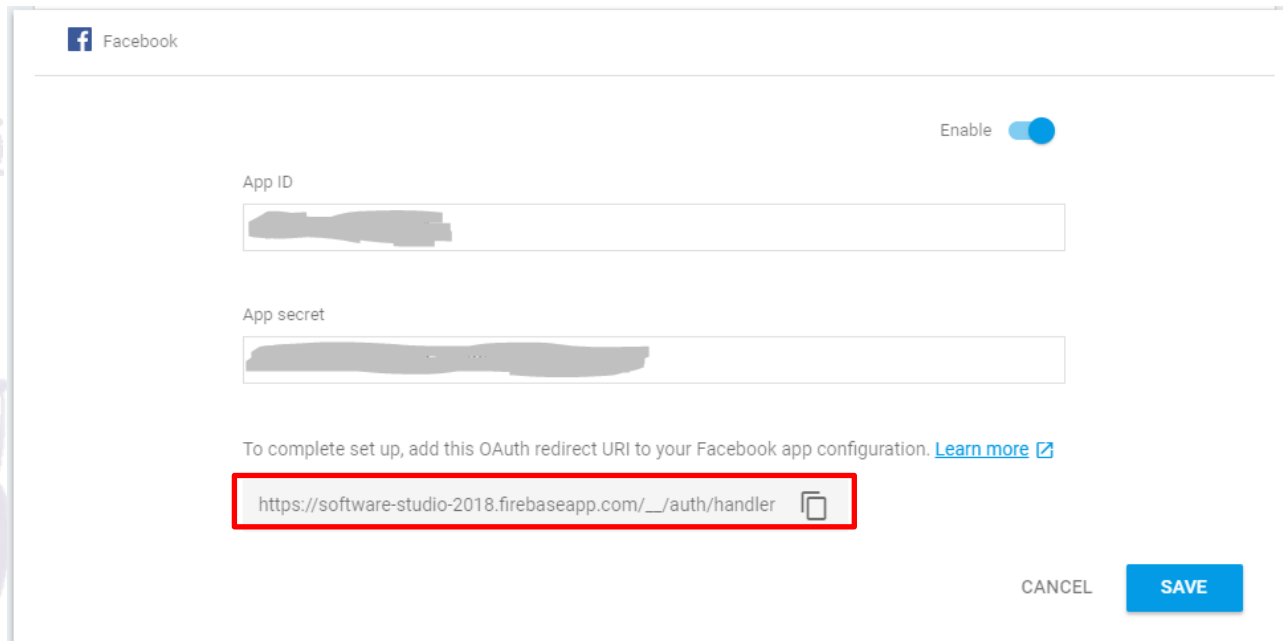
Provider	Status
Email/Password	Enabled
Phone	Disabled
Google	Enabled
Play Games	Disabled
Facebook	Enabled
Twitter	Disabled
GitHub	Disabled
Anonymous	Disabled

(in firebase page)



Facebook Sign-In

- Add OAuth redirect URL (find in firebase console) in **Product > Facebook Login** on the [Facebook for Developers](#)



Facebook

Enable ☒

App ID

App secret

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

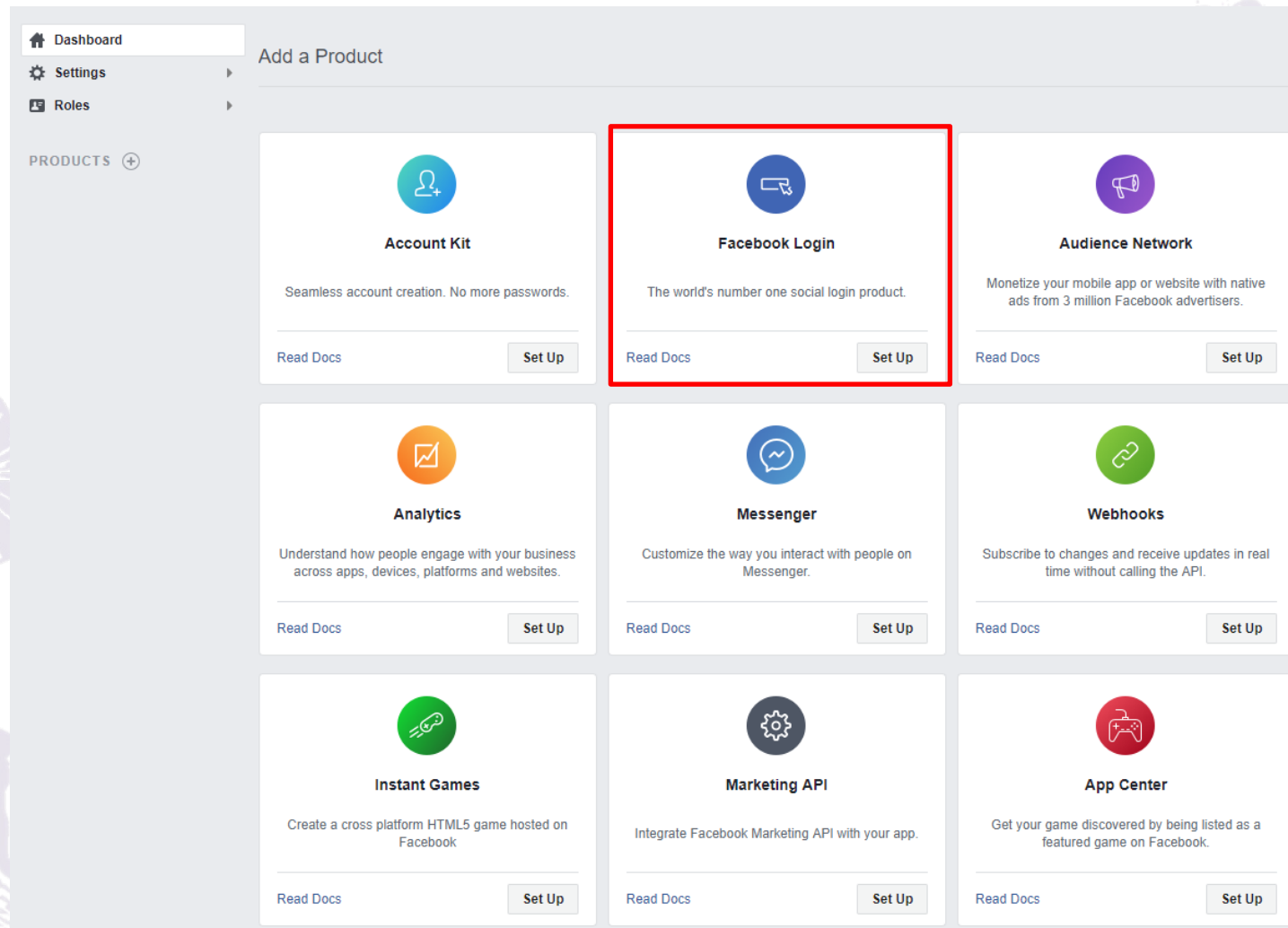
https://software-studio-2018.firebaseio.com/_/auth/handler

CANCEL SAVE

(in firebase page)



Facebook Sign-In



(in Facebook for Developer)



Facebook Sign-In

Dashboard
Settings
Roles
Alerts
App Review

PRODUCTS (+)
Facebook Login
Settings
Quickstart

Activity Log

Client OAuth Settings

☒ **Client OAuth Login**
Enables the standard OAuth client token flow. Secure your application and prevent abuse by locking down which token redirect URIs are allowed with the options below. Disable globally if not used. [?]

☒ **Web OAuth Login**
Enables web-based Client OAuth Login. [?]

☒ **Enforce HTTPS**
Enforce the use of HTTPS for Redirect URIs and the JavaScript SDK. Strongly recommended. [?]

☐ **Force Web OAuth Reauthentication**
When on, prompts people to enter their Facebook password in order to log in on the web. [?]

☐ **Embedded Browser OAuth Login**
Enable webview Redirect URIs for Client OAuth Login. [?]

☒ **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. [?]

Valid OAuth Redirect URIs

☐ **Login from Devices**
Enables the OAuth client login flow for devices like a smart TV [?]

(in Facebook for Developer)



Facebook Sign-In

- Create an instance of the Facebook provider object

```
import { FacebookAuthProvider } from "firebase/auth";  
const provider = new FacebookAuthProvider();
```



Facebook Sign-In

- Sign in with popup

```
import { getAuth, signInWithPopup, FacebookAuthProvider } from "firebase/auth";

const auth = getAuth();
signInWithPopup(auth, provider)
  .then((result) => {
    // The signed-in user info.
    const user = result.user;
  })
  .catch((error) => {
    // Handle Errors here.
    const errorCode = error.code;
    const errorMessage = error.message;
    // The email of the user's account used.
    const email = error.customData.email;
    // The AuthCredential type that was used.
    const credential = FacebookAuthProvider.credentialFromError(error);
  });
```

Facebook Sign-In

- Sign in with redirect

```
import { getAuth, signInWithRedirect } from "firebase/auth";  
  
const auth = getAuth();  
signInWithRedirect(auth, provider);
```

- The **getRedirectResult()** function is shared with google



Manage Users

- You can get the current user is by setting an observer on the Auth object

```
import { getAuth, onAuthStateChanged } from "firebase/auth";

const auth = getAuth();
onAuthStateChanged(auth, (user) => {
  if (user) {
    // User is signed in, see docs for a list of available properties
    // https://firebase.google.com/docs/reference/js/firebase.User
    const uid = user.uid;
  } else {
    // User is signed out
  }
});
```


Manage Users

- You can also get the currently signed-in user by using the currentUser property

```
import { getAuth } from "firebase/auth";

const auth = getAuth();
const user = auth.currentUser;

if (user) {
  // User is signed in, see docs for a list of available properties
  // https://firebase.google.com/docs/reference/js/firebase.User
} else {
  // No user is signed in.
}
```

Sign-Out

- To sign-out a user, call `signOut()`

```
import { getAuth, signOut } from "firebase/auth";

const auth = getAuth();
signOut(auth).then(() => {
  // Sign-out successful.
}).catch((error) => {
  // An error happened.
});
```



Reference

- <https://firebase.google.com/docs/auth/web/start>



Realtime Database

- All Firebase Realtime Database data is stored as JSON objects
- Unlike a SQL database, there are no tables or records

```
{  
  "users": {  
    "alovelace": {  
      "name": "Ada Lovelace",  
      "contacts": { "ghopper": true },  
    },  
    "ghopper": { ... },  
    "eclarke": { ... }  
  }  
}
```

database

Read and Write Data

- Get a database reference

```
import { getDatabase } from "firebase/database";  
  
const database = getDatabase();
```



Write Data

- There are 4 ways to write data to the database
 - **set()**: Write or overwrite specify reference data
 - **push()**: Adding list data, every call to push() will generate a unique ID
 - **update()**: Give a key, and update the data of this key will not cover the entire data
 - **transaction()**: Updating complex data while updating will cause errors



Write Data

- Use **ref()** to reference the specific path.
- Use **set()** to write data and replace any existing data at that path.

```
import { getDatabase, ref, set } from "firebase/database";

function writeUserData(userId, name, email, imageUrl) {
  const db = getDatabase();
  set(ref(db, 'users/' + userId), {
    username: name,
    email: email,
    profile_picture : imageUrl
  });
}
```


Write Data

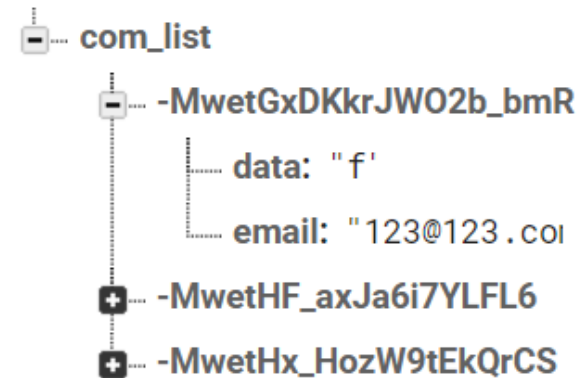
- Use **push()** to push data.
- Every **push()** generate a unique key.
- Use **.key** to access it.



Write Data

```
import { getDatabase, ref, child, push, update }  
from "firebase/database";  
  
function writeNewPost(uid, username, picture, title, body) {  
  const db = getDatabase();  
  
  // A post entry.  
  const postData = ({data: data, email: email});  
  
  // Get a key for a new Post.  
  const newPostKey = push(child(ref(db), 'com_list')).key;  
  
  // Write the new post's data simultaneously in the posts list and the user's post list.  
  const updates = {};  
  updates['/com_list/' + newPostKey] = postData;  
  
  return update(ref(db), updates);  
}
```

software-5290a-default-rtdb



Read Data

- To read data at a path and listen for changes, use the **onValue()** methods of reference to observe events or use query
- Listen for change, including changes to children



Read Data

- **onValue()** receive data as a DataSnapshot object.
- Use **val()** to get the content of the snapshot after you received data.

```
import { getDatabase, ref, onValue } from "firebase/database";

const db = getDatabase();
const starCountRef = ref(db, 'posts/' + postId + '/starCount');
onValue(starCountRef, (snapshot) => {
  const data = snapshot.val();
  updateStarCount(postElement, data);
});
```



Read Data

- Add **onlyOnce: true** to read data once

```
import { getDatabase, ref, onValue } from "firebase/database";
import { getAuth } from "firebase/auth";

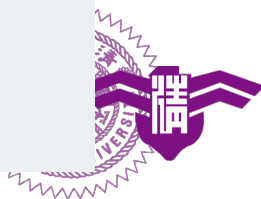
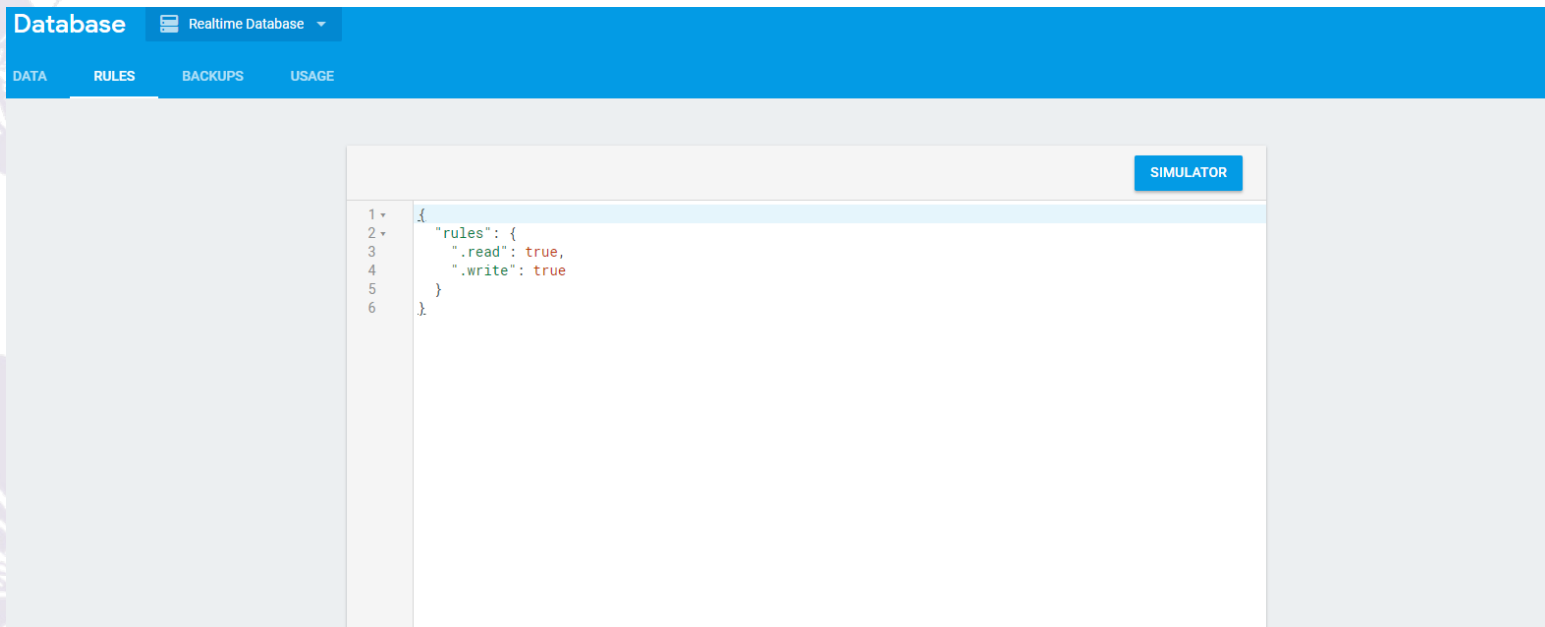
const db = getDatabase();
const auth = getAuth();

const userId = auth.currentUser.uid;
onValue(ref(db, '/users/' + userId), (snapshot) => {
  const username = (snapshot.val() && snapshot.val().username) || 'Anonymous';
}, {
  onlyOnce: true
});
```



Database Rules

- Firebase Realtime Database Rules determine who has read and write access to your database, how your data is structured, and what indexes exist



Database Rules

- Rule Types
 - **.read**
 - **.write**
 - **.validate**: Defines what a correctly formatted value will look like, whether it has child attributes, and the data type
 - **.indexOn**: Specifies a child to index to support ordering and querying



Database Rules

- For example, here's a set of security rules that allows anyone to read the path /foo/, but no one to write to it

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



Database Rules

- The Firebase Database Rules include [built-in variables](#) and functions

```
{  
  "rules": {  
    "users": {  
      "$uid": {  
        ".write": "$uid === auth.uid"  
      }  
    }  
  }  
}
```

rule

```
{  
  "users": {  
    "James": { // id 1  
      "name": "",  
      ""  
    },  
    "John": { // id 2  
      "name": "",  
      ""  
    }  
  }  
}
```

Database data

Database Rules

- The rules language includes a **.validate** rule which allows you to apply validation logic using the same expressions

```
{  
  "rules": {  
    "foo": {  
      ".validate": "newData.isString() && newData.val().length < 100"  
    }  
  }  
}
```



Database Rules

- When you changed database.rules.json locally, you must use **firebase deploy** to update the rule. Otherwise, your changes won't apply.
- Or you can change the rules directly on firebase console.



Database Rules

- More information and usage about database:
- <https://firebase.google.com/docs/reference/security/database>
- <https://firebase.google.com/docs/database/web/start#web-version-9>
- <https://firebase.google.com/docs/database/web/read-and-write>
- <https://firebase.google.com/docs/database/web/start>



Cloud Storage

- Cloud Storage for Firebase lets you upload and share user generated content, such as images and video, which allows you to build rich media content into your apps



Initialization

Storage

不需要伺服器端程式碼，也能儲存及擷取使用者產生的圖片、音訊與影片等檔案

開始使用

設定 Cloud Storage

1 Cloud Storage 安全性規則 ——— 2 設定 Cloud Storage 位置

根據預設，您的規則允許通過驗證的使用者執行所有讀寫作業。

定義資料結構之後，您需要撰寫規則來保護資料。[瞭解詳情](#)

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

取消

下一步

設定 Cloud Storage

✓ Cloud Storage 安全性規則 ——— 2 設定 Cloud Storage 位置

您在位置設定中指定的位置即為 Cloud Storage 預設值區的所在位置，值區資料都會儲存在該處。

⚠ 位置一經設定即無法變更。此位置設定會成為 Cloud Firestore 的預設位置。

[瞭解詳情](#)

Cloud Storage 位置

nam5 (us-central)

Blaze 方案客戶可為額外的值區選擇其他位置

取消

完成

Storage Reference

- Similar to Database, Cloud Storage need a reference to upload/download files

```
import { getStorage, ref } from "firebase/storage";

// Get a reference to the storage service, which is used to create references in
// your storage bucket
const storage = getStorage();
// Create a storage reference from our storage service
const storageRef = ref(storage);

// Create a child reference, imagesRef now points to 'images'
const imagesRef = ref(storage, 'images');

// Child references can also take paths delimited by '/'
// spaceRef now points to "images/space.jpg"
const spaceRef = ref(storage, 'images/space.jpg');
```

Upload File

- Once you've created an appropriate reference, you then up call the put() method to up;

```
import { getStorage, ref } from "firebase/storage";

// Create a root reference
const storage = getStorage();
// Create a reference to 'mountains.jpg'
const mountainsRef = ref(storage, 'mountains.jpg');

var file = ... // use the Blob or File API to get the file
uploadBytes(storageRef, file).then((snapshot) => {
  console.log('Uploaded a blob or file!');
});
//now the file's reference in database is '/mountain.jpg'
```


Upload File (Cont'd)

```
// Create a root reference
var storageRef = firebase.storage().ref();

// Create a reference to 'images/mountains.jpg'
var mountainImagesRef = storageRef.child('images/mountains.jpg');

var file = ... // use the Blob or File API to get the file
mountainImagesRef.put(file).then(function(snapshot) {
  console.log('Uploaded a blob or file!');
});

//now the file's reference is '/images/mountains.jpg'
```



Download File

- Use **getDownloadURL()** to get file URL, your can download file directly by this URL or inserted into a HTML element



Download File

- Directly download the file

```
import { getStorage, ref, getDownloadURL } from "firebase/storage";

const storage = getStorage();
getDownloadURL(ref(storage, 'images/stars.jpg'))
  .then((url) => { // `url` is the download URL for 'images/stars.jpg'
    // This can be downloaded directly:
    const xhr = new XMLHttpRequest();
    xhr.responseType = 'blob';
    xhr.onload = (event) => {
      const blob = xhr.response;
    };
    xhr.open('GET', url);
    xhr.send();
  })
  .catch((error) => { // Handle any errors });
```

Download File (Cont'd)

- Insert file into html element

```
import { getStorage, ref, getDownloadURL } from "firebase/storage";

const storage = getStorage();
getDownloadURL(ref(storage, 'images/stars.jpg'))
  .then((url) => {
    // `url` is the download URL for 'images/stars.jpg'

    // Inserted into an <img> element
    const img = document.getElementById('myimg');
    img.setAttribute('src', url);
  })
  .catch((error) => {
    // Handle any errors
  });
```

Reference

- <https://firebase.google.com/docs/storage/web/create-reference>



Cloud Storage Rules

- Storage Security Rules manage the complexity for you by allowing you to specify path-based permissions

```
// Rules can optionally specify a condition  
allow write: if <condition>;
```

Storage Rules

```
service firebase.storage {  
  // The {bucket} wildcard indicates we match files in all Cloud Storage buckets  
  match /b/{bucket}/o {  
    // Match filename  
    match /filename {  
      allow read: if <condition>;  
      allow write: if <condition>;  
    }  
  }  
}
```

Storage Rules

Cloud Storage Rules

- You can also specify the file size

// Rules can specify conditions that consider the request

Storage Rules

```
service firebase.storage {  
  match /b/{bucket}/o {  
    match /images/{imageId} {  
      // Only allow uploads of any image file that's less than 5MB  
      allow write: if request.resource.size < 5 * 1024 * 1024  
        && request.resource.contentType.matches('image/.*');  
    }  
  }  
}
```



Cloud Storage Rules

- More information and usage about cloud storage rules:
- <https://firebase.google.com/docs/storage/security>

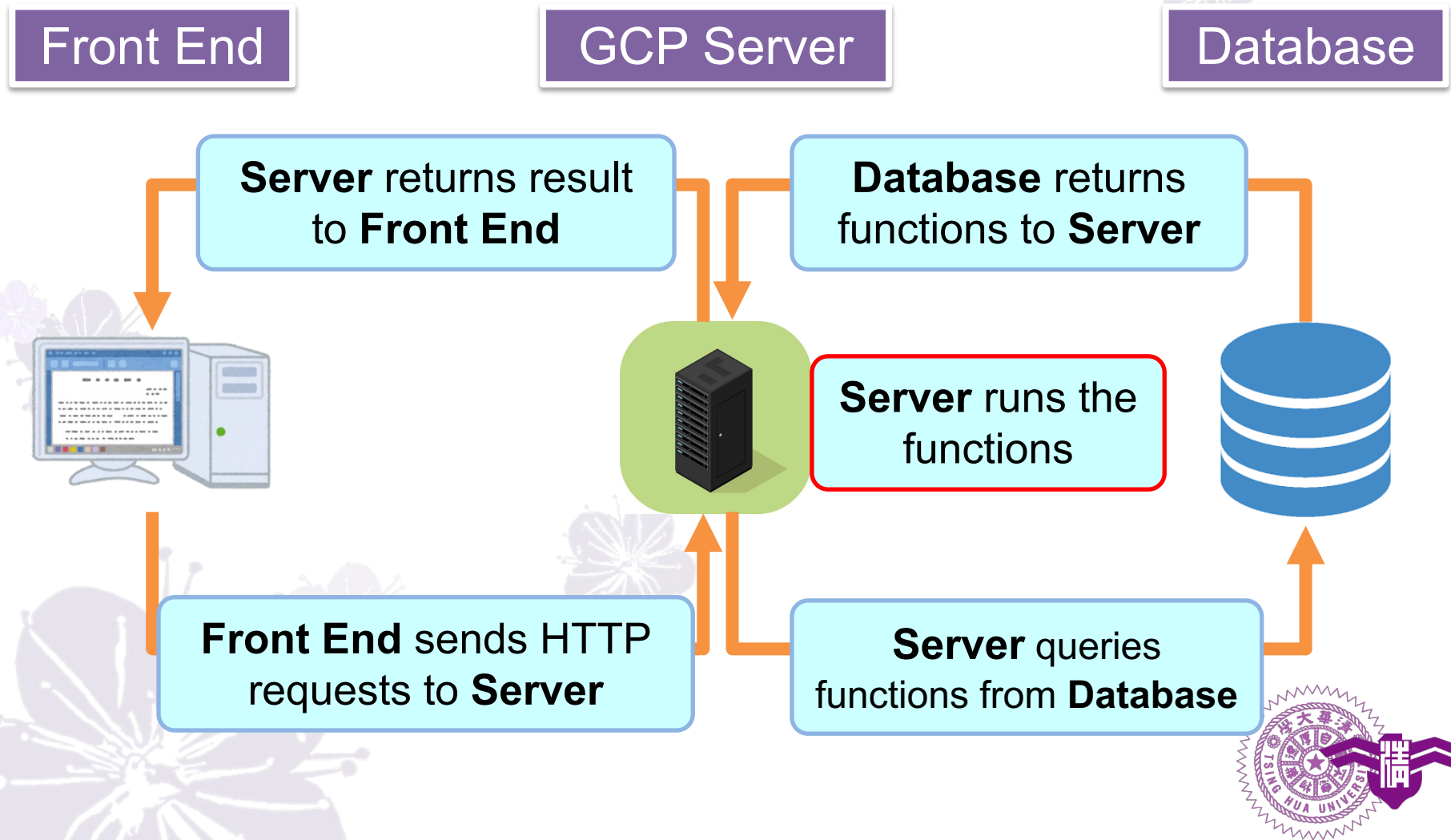


Cloud Function

- Cloud Functions let you automatically run backend code in response to events triggered by Firebase features and HTTPS requests.
- Your code is stored in Google's cloud and runs in a managed environment. There's no need to manage and scale your own servers.



Cloud Function



Cloud Function

- First, enable cloud functions to your project.

Command : `firebase init functions`

```
? Are you ready to proceed? Yes

=== Project Setup

First, let's associate this project directory with a Firebase project.
You can create multiple project aliases by running firebase use --add,
but for now we'll just set up a default project.

? Please select an option: Use an existing project
? Select a default Firebase project for this directory: software-studio-6698b (Software Studio)
i Using project software-studio-6698b (Software Studio)

=== 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
+ Wrote functions/.gitignore
? Do you want to install dependencies with npm now? Yes

> protobufjs@6.8.9 postinstall D:\大三嗨起來\助教人生\軟體實驗\Lecture Program\Firebase\answer\functions\node_modules\protobufjs
> node scripts/postinstall

npm notice created a lockfile as package-lock.json. You should commit this file.
added 258 packages from 281 contributors and audited 930 packages in 73.817s

30 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

i Writing configuration info to firebase.json...
i Writing project information to .firebaserc...
i Writing gitignore file to .gitignore...

+ Firebase initialization complete!
```



Cloud Function

- Then you will get the following folder in your project.
- Now, we are going to edit **index.js** to create our cloud functions.

```
└─ functions
   └─ node_modules
      .gitignore
      JS index.js
      {} package-lock.json
      {} package.json
```



Get Firebase admin

- Just like what we did **initializeApp(config)** in our .js files. We initialize firebase again in index.js.
- After that we can use the firebase functions, such as database or storage.

```
const admin = require('firebase-admin');  
admin.initializeApp();
```

index.js



Create a New Function

- This function is an HTTP endpoint. Any request to the endpoint results in ExpressJS-style Request and Response objects passed to the `onRequest()` callback. <https://expressjs.com/zh-tw/>

```
exports.addMessage = functions.https.onRequest(async (req, res) => {  
  // Grab the text parameter.  
  const original = req.query.text;  
  // Push the new message into Firestore using the Firebase Admin SDK.  
  const writeResult = await admin.firestore().collection('messages').add({original:  
original});  
  // Send back a message that we've successfully written the message  
  res.json({result: `Message with ID: ${writeResult.id} added.`});  
});
```

Server code

```
// This client-end function is equivalent to the previous cloud function  
function addMessageLocal(text){  
  const writeResult = await admin.firestore().collection('messages')  
  .add({original: original});}
```

Client code

Setup Local Testing Server

- Once you finish editing your cloud functions, you can start up a local server to test the correctness of your functions.

Command : `firebase emulators:start --only functions`

```
$ firebase emulators:start --only functions
i  emulators: Starting emulators: functions
!  Your requested "node" version "8" doesn't match your global version "12"
+  functions: Emulator started at http://localhost:5001
i  functions: Watching "C:\Users\penguin\Desktop\firebase\functions" for Cloud Functions...
+  functions[addMessage]: http function initialized (http://localhost:5001/software-studio-6698b/us-central1/addMessage).
+  All emulators started, it is now safe to connect.
```



Use Cloud Functions (local)

- After initialize firebase in your js code, you can get the functions you created.
- Remember that if you are using **local server** to call cloud functions, you have to use **connectFunctionsEmulator()** to redirect to localhost server.

// connect to localhost server

Client code

```
import { getApp } from "firebase/app";  
import { getFunctions, connectFunctionsEmulator } from "firebase/functions";  
  
const functions = getFunctions(getApp());  
connectFunctionsEmulator(functions, "localhost", 5001)
```



Deploy Cloud Functions

- Deploy cloud function to server require upgrading firebase service to **Blaze**

8

將功能部署到生產環境

一旦您的功能在仿真器中可以正常工作，您就可以繼續在生產環境中部署，測試和運行它們。請記住，要部署到建議的 Node.js 12 運行時環境，您的項目必須處於 Blaze 即用即付 [計費計劃](#) 中。請參閱 [雲功能定價](#)。

Firebase 計費方案

Spark

免費 每月 \$0 美元

資料庫、Firestore、儲存、函式、電話驗證、代管和 Test Lab 有使用量配額

✕ 你的專案將能使用 Google Cloud 功能

✓ 所有方案皆包含
數據分析、通知、當機報告和支援等服務

[查看完整方案詳情](#)

目前的方案

Blaze

用多少，付多少

包括免費用量，每天會計算一次。超出免費用量上限後，您只需針對專案的使用量付費。

你的專案將能使用 Google Cloud 功能

✓ 所有方案皆包含
數據分析、通知、當機報告和支援等服務

[查看完整方案詳情](#)

選取方案



Deploy Cloud Functions

- Deploy function to server make sure we don't need to run server by ourselves

Command : `firebase deploy --only functions`

```
$ firebase deploy --only functions

=== Deploying to 'software-studio-6698b'...

i deploying functions
i functions: ensuring necessary APIs are enabled...
+ functions: all necessary APIs are enabled
i functions: preparing functions directory for uploading...
i functions: packaged functions (26.79 KB) for uploading
+ functions: functions folder uploaded successfully
i functions: creating Node.js 8 function addMessage(us-central1)...
+ functions[addMessage(us-central1)]: Successful create operation.
Function URL (addMessage): https://us-central1-software-studio-6698b.cloudfunctions.net/addMessage
+ Deploy complete!

Project Console: https://console.firebase.google.com/project/software-studio-6698b/overview
```



Use Cloud Functions

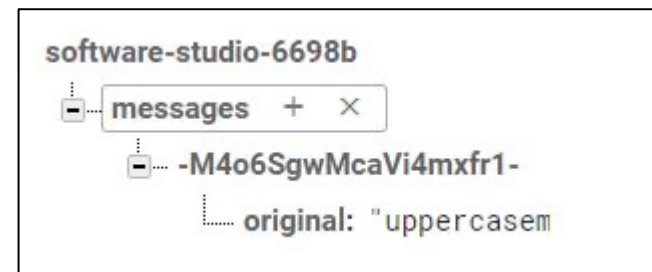
- After deploy firebase cloud functions, you can use the functions you created.

```
import { getFunctions, httpsCallable } from "firebase/functions";  
  
const functions = getFunctions();  
const addMessage = httpsCallable(functions, 'addMessage');  
addMessage({ text: messageText })  
  .then((result) => {  
    // Read result of the Cloud Function.  
    /** @type {any} */  
    const data = result.data;  
    const sanitizedMessage = data.text;  
  });fb_functions_call_add_message.js
```

Client code

Demo

- Enter the following url in your browser.
- [https://us-central1-\[MY_PROJECT\].cloudfunctions.net/addMessage?text=uppercasemetoo](https://us-central1-[MY_PROJECT].cloudfunctions.net/addMessage?text=uppercasemetoo)
- Then go to firebase console and check your real-time database. If cloud functions setup successfully, you will have following structure.



Reference

- [Firestore Documentation](#)
- [Firestore Get Started – Web](#)
- [Firestore on the Web - Tutorials](#)



thank
you!

Question

