

# Lab15

## Cloud Messaging

本節目的：

- 了解何謂 Notification、Cloud Messaging。
- 認識 Firebase 雲端開發平台。
- 使用 Firebase 實現 Cloud Messaging。

## 15.1 推播

當手機連上網路時，就會收到一些來自網路的訊息，例如：Line 的聊天訊息、臉書的好友邀請...等，你有想過這些訊息是怎麼出來的嗎？

「推播」（Notification），意指手機上的訊息通知，允許裝置在沒有啟動應用程式的情況下，從網路推送訊息給使用者，而這些來自網路的訊息，我們稱為「雲端訊息」（Cloud Messaging），開發者將訊息/資料透過雲端的伺服器推送到使用者的裝置上，使用者不必下載或開啟應用程式，就可以獲得最新的訂單資訊、叫車進度與對話通知，如圖 15-1 所示。



圖 15-1 Cloud Messaging

### 說明

推播除了作為提供使用者即時訊息的管道，也常運用於商業行銷，開發者能透過雲端訊息向使用者推播具有客製化或主題的資料內容，提供更符合使用者且更有價值的訊息。

## 15.1.1 Firebase

要在手機上實現應用程式的推播功能，通常需要架設一個 Web Service Server 以及一個 Notification Server，用作身份註冊與訊息推送，如下圖 15-2 所示。這樣的架構對於一位開發者而言，無疑是一道難以跨越的門檻。

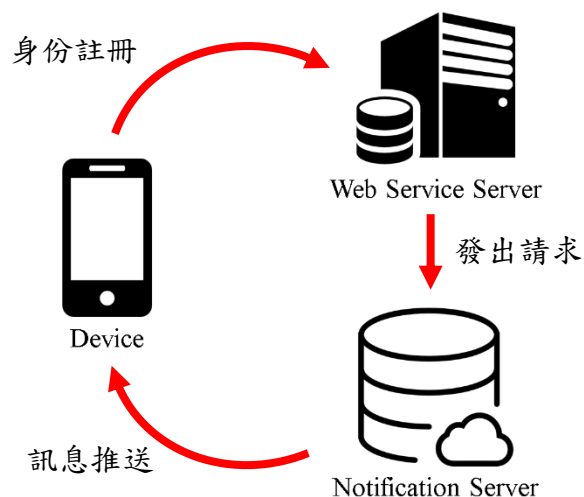


圖 15-2 推播架構

Google 在 2016 年的 I/O 開發者大會中，發表了新版的 Firebase，Firebase 是一個行動應用程式的開發平台，提供即時資料庫、資料分析與雲端訊息推播...等服務，協助開發者在雲端快速建置後端服務，同時支援 Android、iOS 與 Web 三大平台，讓開發者可以更專注於前端的優化上。

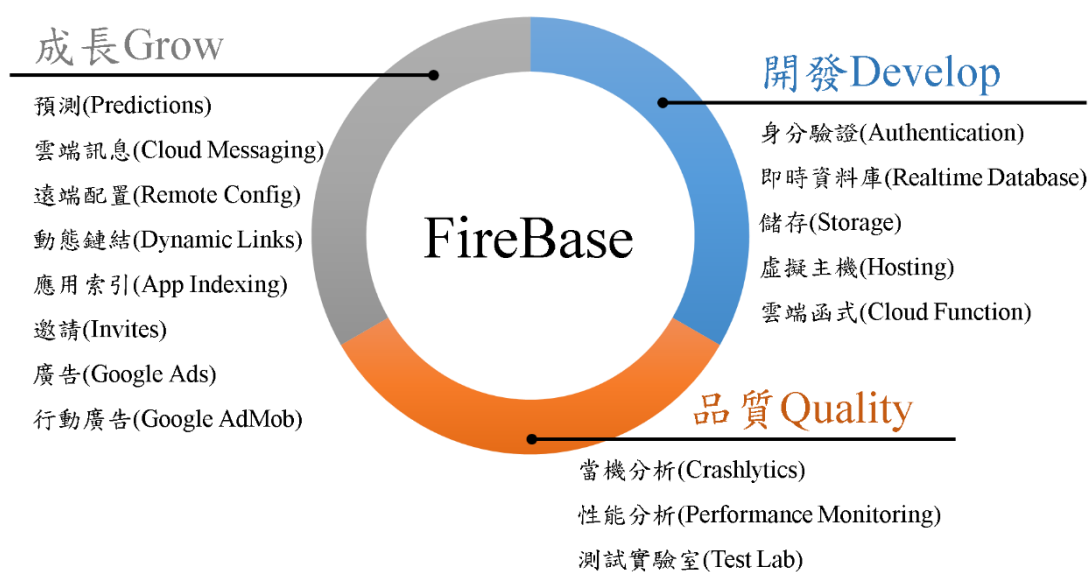


圖 15-3 Firebase Cloud Service

圖 15-3 為 Firebase 提供的雲端服務內容，分為開發、成長與品質三階段，並整合分析工具，提供事件紀錄、使用者分析與 APP 品質管理...等多項服務。其中雲端訊息（Cloud Messaging）**現階段完全免費**，部分項目在流量或功能的限制下也提供免費使用，讓新創團隊與輕度使用者能節省開發與維護成本。

#### 開發 Develop：

- 即時資料庫（Realtime Database）：即時資料庫是一個雲端資料託管服務，資料以 JSON 的格式保存，並即時同步到每台連結的終端裝置。
- 雲端函式（Cloud Function）：Cloud Functions 可以自動運行後端程式，以響應由 Firebase 功能和 HTTPS 請求觸發的事件。程式存儲在 Google 的雲端上，無需管理和調節自己的伺服器。
- 身分驗證（Authentication）：身分驗證提供應用程式對使用者的身份進行驗證，支持密碼、電話號碼與社群媒體帳號（如 Google、Facebook、Twitter...等）。
- 虛擬主機（Hosting）：開發人員的 Web 內容託管，可以快速部署 Web 應用程式，並將靜態和動態內容提供給全局 CDN（內容託管網絡）。
- 儲存（Storage）：提供 Google 安全品質的文件上傳下載服務。可以使用 SDK 來存儲圖片、音頻、視頻或其他由使用者生成的內容。

#### 成長 Grow：

- 雲端訊息（Cloud Messaging）：**Firebase Cloud Messaging** 是一種跨平台的訊息傳遞，讓開發者可以免費可靠地傳遞訊息，通知終端裝置應用程式同步電子郵件或其他資料。
- 遠端配置（Remote Config）：遠端配置可讓開發者更改應用的行為和外觀，而無需使用者下載應用更新。
- 動態鏈結（Dynamic Links）：在行動裝置上開啟動態鏈接時，可以直接轉到開發者應用程式中的鏈接內容。如果在桌面瀏覽器中打開相同的鏈接，則可以轉到網站上的同等內容。
- 應用索引（App Indexing）：App Indexing 可幫助使用者在應用程式上查找公開內容和個人內容，甚至提供查詢自動填充功能以幫助他們更快地找到所需內容，從而重新吸引這些使用者的關注。
- 預測（Predictions）：預測會將機器學習應用於您的分析資料，從而根據應用程式中預測的使用者行為創建動態使用者群集。
- 邀請（Invites）：邀請是一款即開即用的服務，支持通過電子郵件或簡

訊進行應用程式推薦和分享。

- 廣告（Google Ads）：透過線上廣告吸引潛在客群，提升應用程式安裝量、深入分析廣告轉化情況，並利用 Google Analytics for Firebase 對群體投放有針對性的廣告系列來擴大使用群。
- 行動廣告（Google AdMob）：Google AdMob 是一種移動廣告平台，可用於從您的應用程式獲得額外營收。

品質 Quality：

- 性能分析（Performance Monitoring）：性能監控服務可幫助開發者深入了解應用程式的性能特徵。開發者可以使用 SDK 收集應用的性能資料，然後在 Firebase 控制台中查看和分析。
- 當機分析（Crashlytics）：Crashlytics 是一個輕量級的即時崩潰報告，幫助開發者對影響應用品質的穩定性問題進行追蹤、確定優先順序並加以修復。
- 測試實驗室（Test Lab）：測試實驗室只需一項操作，就能測試應用程式在各種設備上和設備配置下的表現，並可以在 Firebase 控制台查看測試結果（包含日誌、影片與螢幕截圖）。

## 15.1.2 Firebase Cloud Messaging (FCM)

Firebase Cloud Messaging (FCM) 的前身為 Google Cloud Messaging (GCM)，現已完全被取代，Firebase Cloud Messaging 支援網頁控制台，同時提供有 Android、IOS 與 Web 的跨平台訊息通知服務。

### ● 新增通知

FCM 的訊息分為 Notification Message 與 Data Message 兩種格式，支援 key-value 格式的資料。Notification Message 包含了 title、body、icon... 等預先定義好的鍵值資料，並會自動向終端裝置顯示訊息，而 Data Message 則只有自定義鍵值的資料，不會向終端裝置顯示消息，目的是讓前端專注於資料的處理。

目前網頁控制台僅支援 Notification Message，下方為發送 Notification Message 的操作示範：

**Step1** 開發者可以透過「新增通知」創建新訊息，對前端發送通知，如圖 15-4 所示。

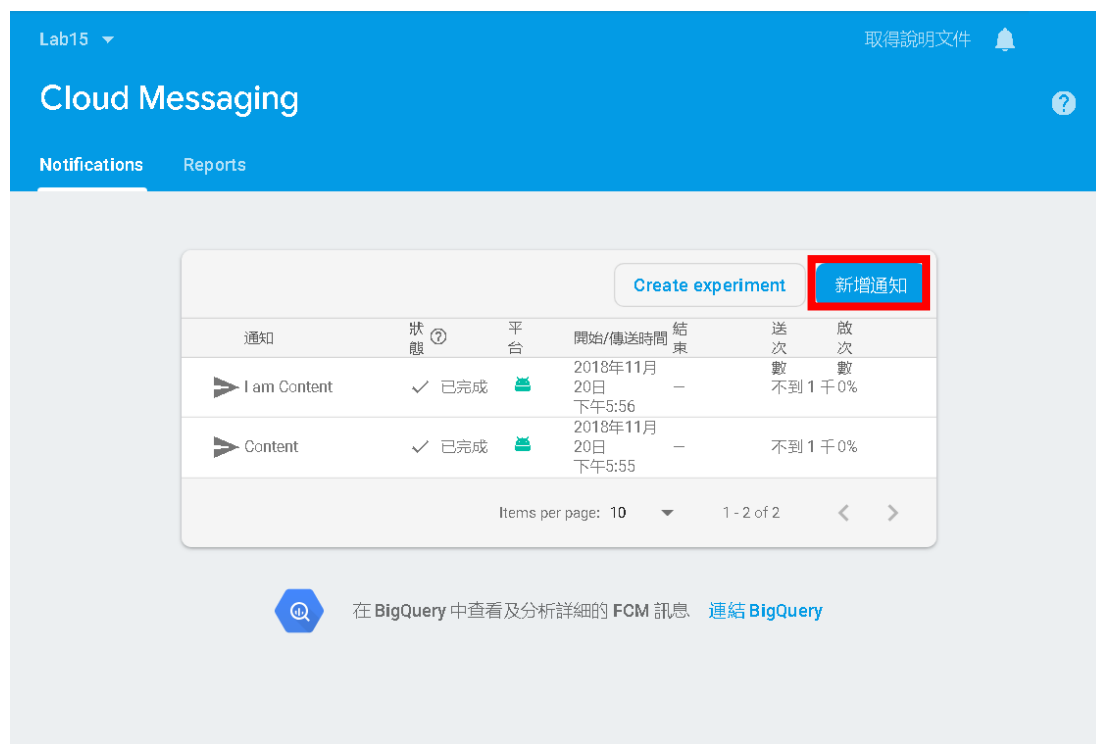


圖 15-4 Firebase Cloud Messaging 控制台

**Step2** 設定 Notification Message 的標題與文字內容，並且可以加入訊息標籤以便後續追蹤之用，設定完成後按下「下一步」，如圖 15-5 所示。

1 建立訊息

訊息標題 (選填) ?

訊息文字

訊息標籤 (選填) ?

[在裝置上進行測試](#)

下一步

圖 15-5 建立訊息

**Step3** 推播方式分為圖 15-5 的在裝置上進行測試 (Token) 以及圖 15-6 的使用者區隔 (應用程式 ID) 與主題 (Topic) 三種，前者可以針對單一的使用者，而後者則可針對不同的使用客群推播指定的內容。

建立訊息  
Content


2 指定目標

使用者區隔

主題

如果符合下列條件，則指定使用者：

應用程式

 bluenet.com.lab15

▼

且

Target another app

這項活動將指定 19 名應用程式目標使用者 (佔潛在使用者中的 100%)。 ?

下一步

圖 15-6 指定目標

**Step4** 使用者區格除了應用程式外，還可額外加入版本、語言、行為...等屬性進一步劃分使用客群，此處設定 Lab15 即可，完成後按下「下一步」，如圖 15-7 所示。

版本  
應用程式版本 (可能因平台而不同)。

語言  
使用者選擇的語言和語言代碼。

使用者目標對象  
使用者所屬的目標對象。

使用者屬性  
比對使用者屬性的值

預測  
依其在接下來 7 天會有特定互動行為的可能性來鎖定使用者

與應用程式的最近一次互動  
根據最近一次與您應用程式互動的時間來

bluenet.com.lab15

Target another app

這項活動將指定 20 名應用程式目標使用者 (佔潛在使用者中的 100%)。?

下一步

圖 15-7 使用者區格

**Step5** FCM 支援排程作業可以定時發送推播，舉凡特殊節日或是周年活動等皆可事先做好內容，等待時間到自動推送給使用者，此處使用預設 Now 並按下「下一步」，如圖 15-8 所示。

指定目標  
符合「1 項指定條件」的使用者區隔

3 排定時間

傳送給符合資格的使用者

Now

下一步

圖 15-8 推播排程



**Step6** 轉換事件需搭配 FCM 數據分析的「Events」功能使用，此處不多作介紹，按下「下一步」即可，如圖 15-9 所示。

指定目標  
符合「1 項指定條件」的使用者區隔

排定時間  
立即傳送

4 設定轉換事件 (選用)

已傳送 | 已開啟

下一步

5 其他選項 (選用)

Conversion events

- first\_open
- in\_app\_purchase
- ecommerce\_purchase

圖 15-9 轉換事件

**Step7** 其他選項中，可以額外設定 Android 通知通道、自訂資料、優先權、音效與時效，如圖 15-10 所示，在自訂資料中以 key value 的方式夾帶自訂的資料到通知中，完成後按下「發佈」即可發送推播。

其他選項 (選用)

所有欄位皆為選填欄位

Android 通知管道

自訂資料

鍵	值
key1	value1

優先順序 音效

高 已啟用

到期時間

1 天

儲存為草稿 發佈

圖 15-10 自訂資料

## 15.2 設計重點：

- 連動 Android Studio 專案與 Firebase。
- 練習使用 Firebase 網頁控制台發送 Cloud Messaging。

Firebase 連結如下：<https://firebase.google.com/>

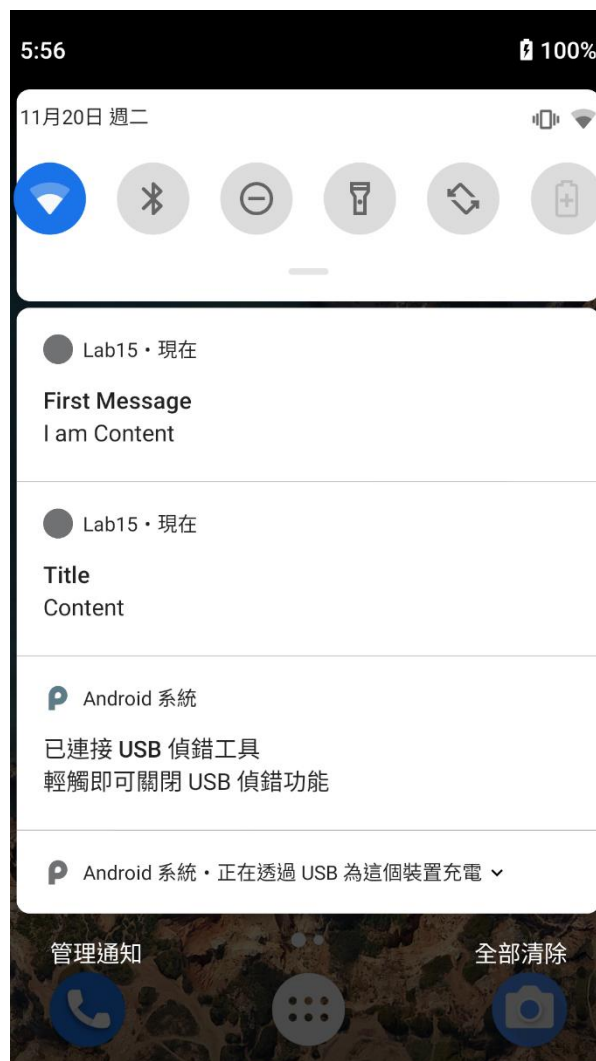


圖 15-11 接收推播通知

## 15.2.1 連動 Firebase Cloud Messaging

**Step1** 建立新專案，點選「Tools→Firebase」開啟 Firebase Assistant，如圖 15-12 所示。

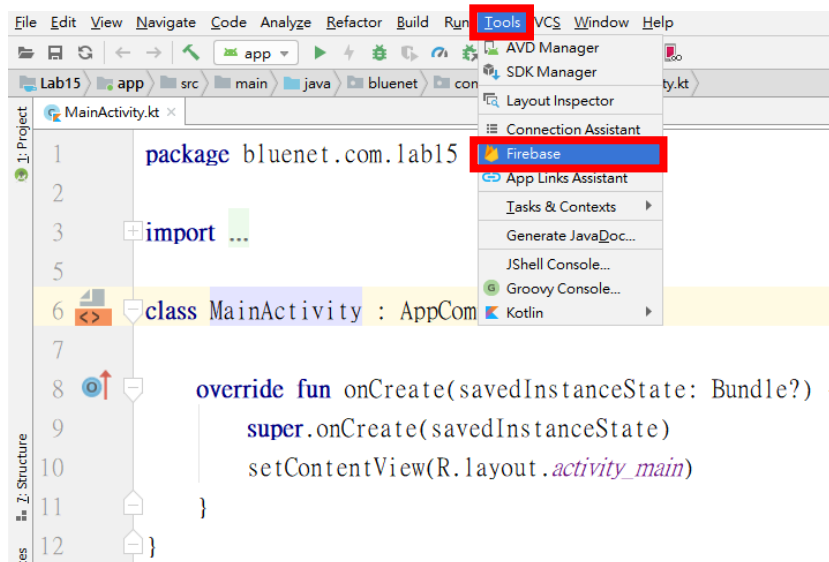


圖 15-12 開啟 Firebase Assistant

**Step2** 選擇「Cloud Messaging」，並按下「Set up Firebase Cloud Messaging」，連動 Google 帳戶與 Android Studio，如圖 15-13 所示。

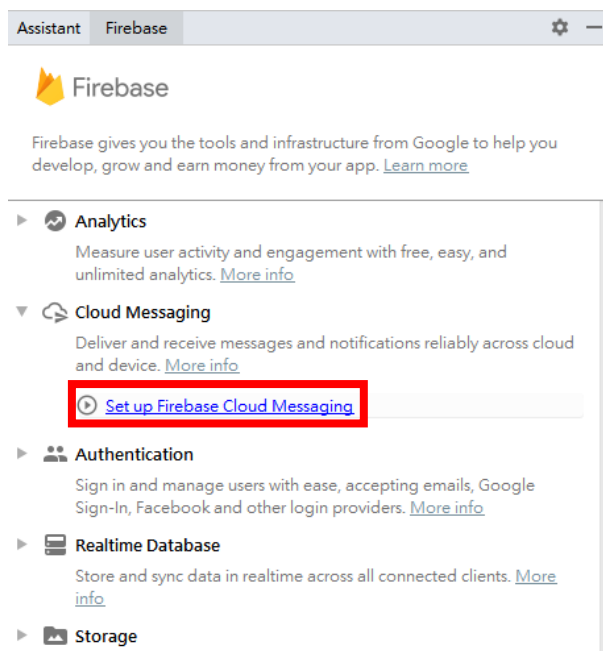


圖 15-13 點選 Set up Firebase Cloud Messaging

**Step3** 連動成功後會進入設定頁面，點擊「Connect to Firebase」，等待連線至 Firebase，如圖 15-14 所示。

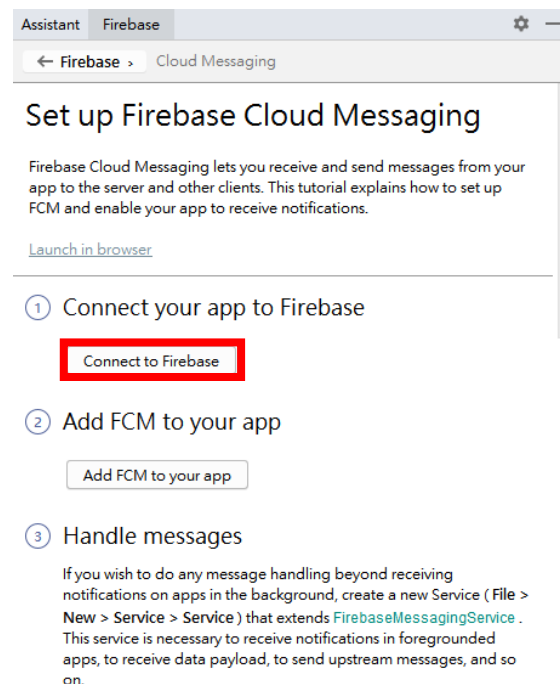


圖 15-14 Connect to Firebase

**Step4** 選擇「Create new Firebase project」並按下「Connect to Firebase」，如圖 15-15 所示。

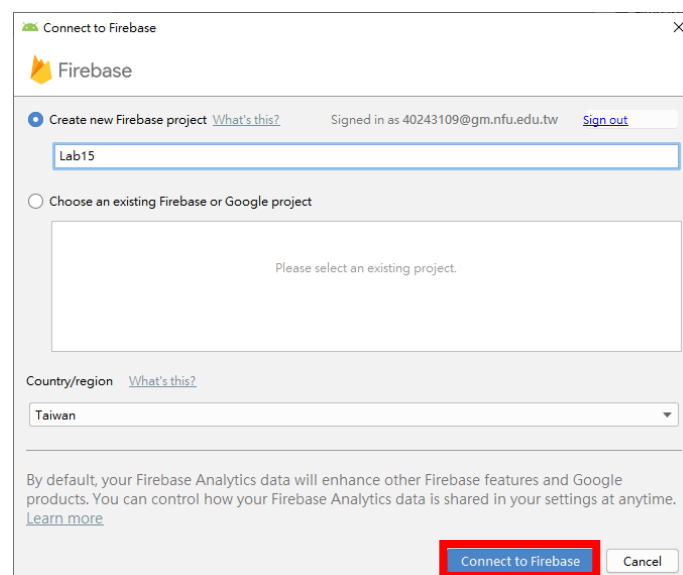


圖 15-15 Create new Firebase project

**Step5** 連動成功後可以看到圖 15-16 中步驟一的「Connected」字樣與右下角的提示訊息，接著點擊步驟二「Add FCM to your app」，如圖 15-16 所示。

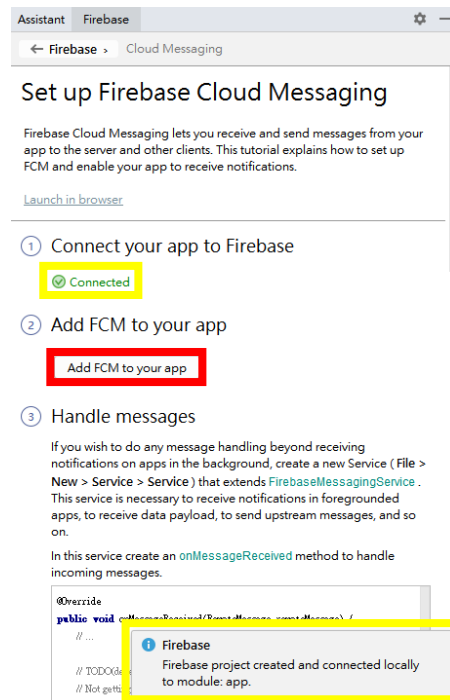


圖 15-16 連動成功

**Step6** 點選「Accept Changes」自動添加 google-service 與 firebase-messaging library (之後需要手動指定 library 版本)，如圖 15-17 所示。

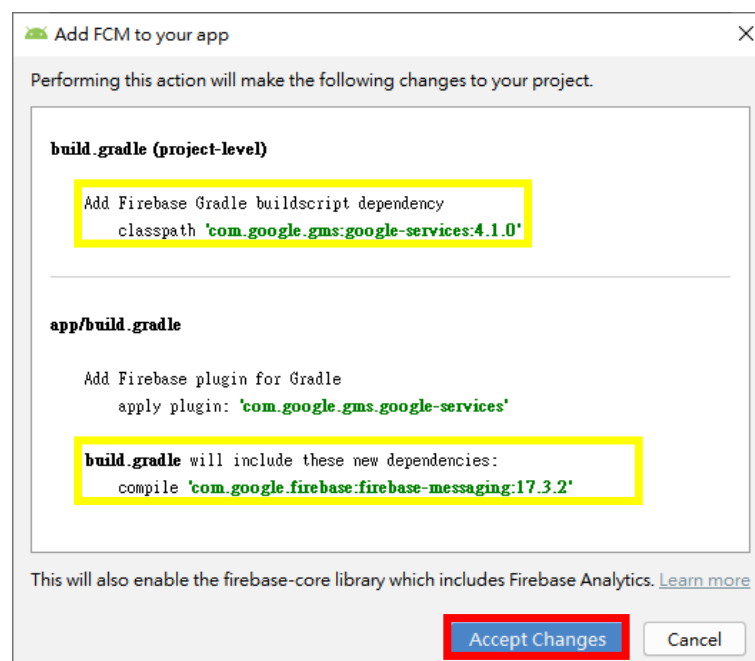


圖 15-17 Add library

**Step7** 修改 build.gradle(Project: Lab15)中的 **google-services** 版本為 4.2.0。

```
buildscript {  
    ext.kotlin_version = '1.3.11'  
    repositories {  
        google()  
        jcenter()  
    }  
    dependencies {  
        classpath 'com.android.tools.build:gradle:3.3.0'  
        classpath  
        "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"  
        classpath 'com.google.gms:google-services:4.2.0'  
    }  
}
```

**Step8** 修改 build.gradle(Module: app)中的 **firebase-messaging** 版本為 17.3.4，並加入 **firebase-core** 版本為 16.0.6。

```
dependencies {  
    implementation fileTree(dir: 'libs', include: ['*.jar'])  
    ...  
    implementation 'com.google.firebase:firebase-core:16.0.6'  
    implementation 'com.google.firebase:firebase-messaging:17.3.4'  
}
```

**Step9** 完成後必須按下畫面上方的同步按鈕讓系統將其匯入，如圖 15-18 所示。

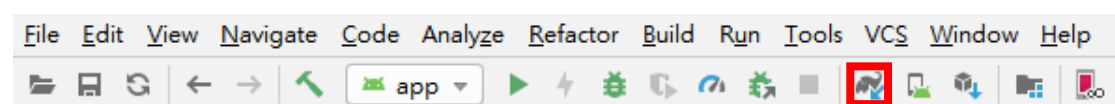


圖 15-18 修改 firebase-messaging 版本為 17.3.4

**Step10** 在專案中創建一個 Class 命名為 MyMessagingService，繼承 FirebaseMessagingService 類別，複寫 onNewToken()與 onMessageReceived()函式，程式碼如下：

```
class MyMessagingService : FirebaseMessagingService(){
    //取得新 token 時呼叫，通常是在第一次啟動 APP 時會自動與 Firebase 註冊
    override fun onNewToken(token: String?) {
        super.onNewToken(token)
        Log.e("Firebase", "onNewToken $token")
    }
    //APP 在前景時收到 Notification Message 會呼叫
    override fun onMessageReceived(msg: RemoteMessage?) {
        super.onMessageReceived(msg)
        Log.e("Firebase", "onMessageReceived")
        //判斷收到的 msg 不為 null
        msg?.let {
            Log.e("Firebase", it.from)
            //透過 for loop 將 msg 夾帶的資料輸出
            for(entry in it.data.entries)
                Log.e("message", "${entry.key}/${entry.value}")
        }
    }
}
```

**Step11** 在 AndroidManifest.xml 中加入 MyMessagingService 定義。

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".MainActivity">
        <intent-filter>
            <action
```

```

        android:name="android.intent.action.MAIN"/>

        <category
            android:name="android.intent.category.LAUNCHER"/>
        </intent-filter>
    </activity>

    <service android:name=".MyMessagingService">
        <intent-filter>
            <action
                android:name="com.google.firebase.MESSAGING_EVENT" />
            </intent-filter>
        </service>
</application>

```

**Step12** 將 Lab15 安裝至模擬器或實機上，並透過「Logcat」追蹤 Firebase Token，取得 Token 代表 APP 已成功註冊 Firebase Messaging 服務（若未看到 Token，請檢查網路是否可用並重啟 APP），如圖 15-19 所示。

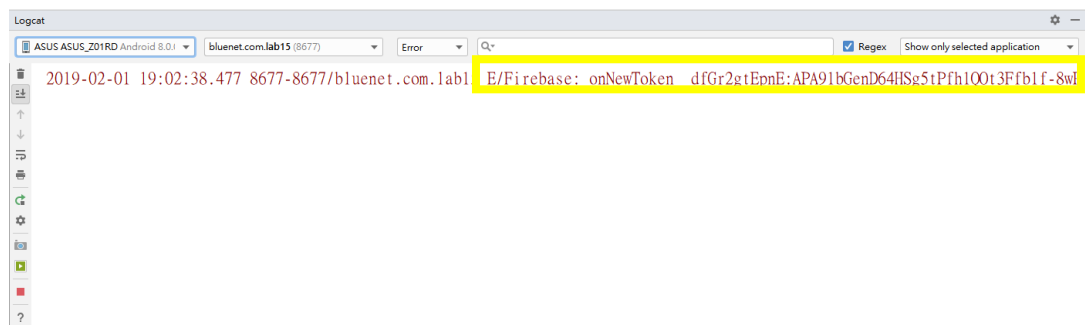


圖 15-19 顯示 Firebase Token



## 15.2.2 發送 Cloud Messaging

**Step1** 開啟 Firebase 網址：<https://firebase.google.com>，點選「GET STARTED」並選擇 Lab15 進入專案控制台，如圖 15-20 所示。

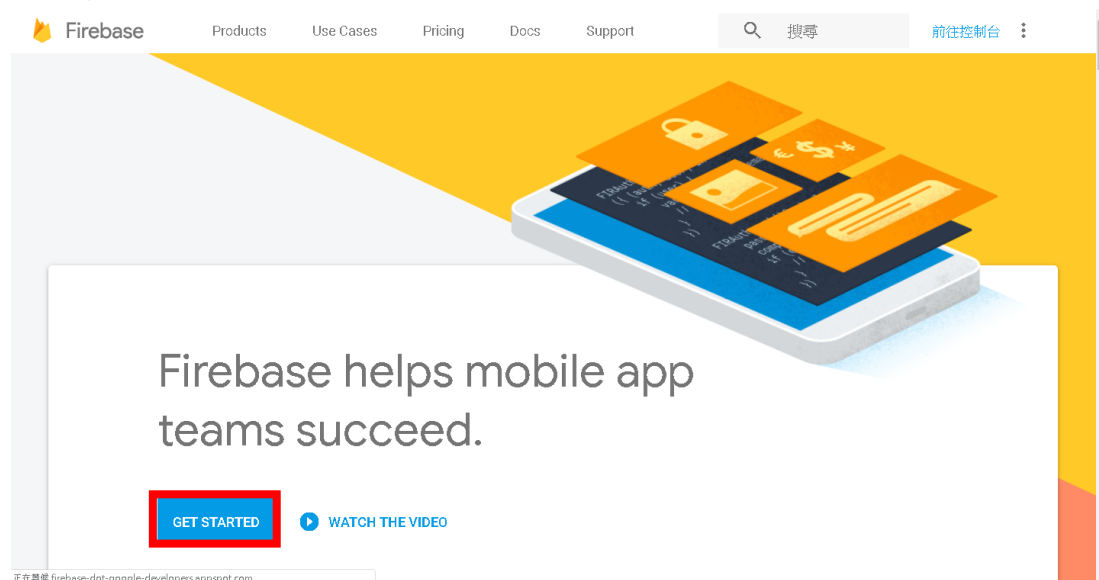


圖 15-20 Firebase 首頁

**Step2** 在控制台右側功能列中找到「Cloud Messaging」，並點選「Send your first message」，如圖 15-21 所示。

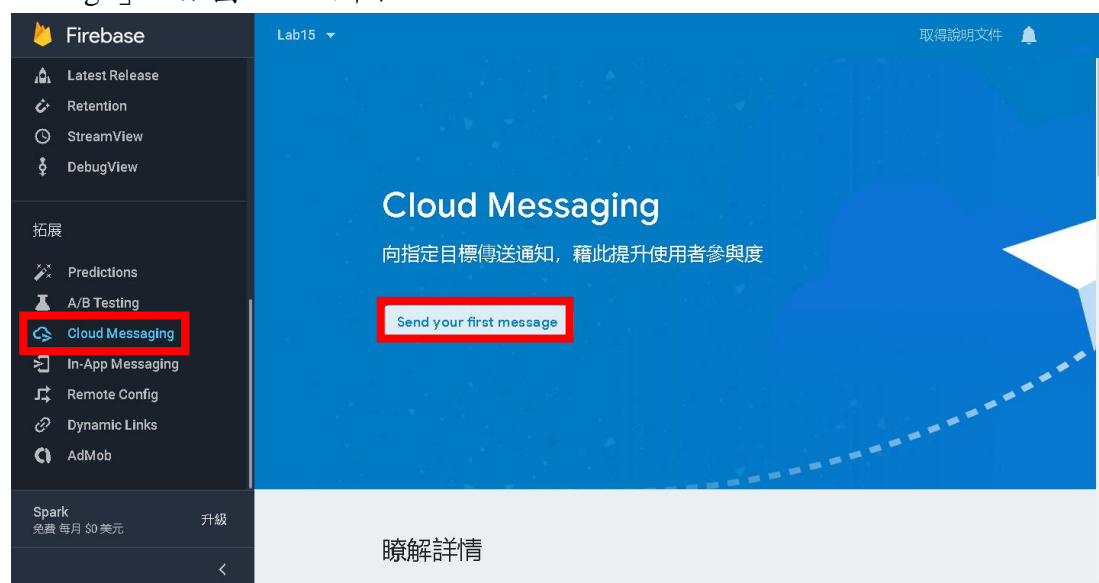


圖 15-21 Cloud Messaging 控制台

**Step3** 填入訊息標題、文字，並按下「在裝置上進行測試」，如圖 15-22 所示。

1 建立訊息

訊息標題 (選填) ?

訊息文字

訊息標籤 (選填) ?

在裝置上進行測試

下一步

圖 15-22 建立訊息並在裝置上進行測試

**Step4** 輸入 **Logcat** 取得的 Token 貼上後按下「+」號加入，並按下「測試」，如圖 15-23 所示。

### 在裝置上進行測試

在下方輸入或選取開發裝置的 [Firebase 執行個體 ID 憑證](#)，即可測試這項活動。



未設定測試裝置

取消

測試

圖 15-23 在裝置上進行測試

**Step5** 當 APP 在前景時，會觸發 `onMessageReceived()`，如圖 15-24 所示。在背景時則會在上方狀態欄留下訊息通知（圖 15-11）。

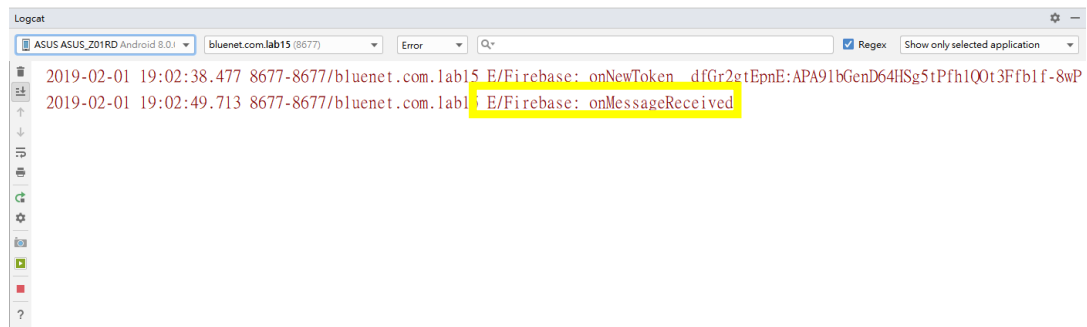


圖 15-24 觸發 `MessageReceived`

#### 說明

測試時要注意部分裝置在靜音或省電模式會啟動勿打擾功能，會導致收到的訊息隱藏且不會發出音效！