**異動記錄**

|  |  |
| --- | --- |
| 日期 | 內容 |
| 2021/04/23 | 初版定版 |
| 2021/05/06 | 打包設定的說明異動 |
| 2021/06/17 | 1. config.json 的說明修正 2. 環境準備更新 |

[環境準備 2](#_Toc74822451)

[Xcode 2](#_Toc74822452)

[Command Line Tools 2](#_Toc74822453)

[RVM 2](#_Toc74822454)

[Ruby 2.6.3 3](#_Toc74822455)

[Gemset 3](#_Toc74822456)

[Bundler 3](#_Toc74822457)

[執行 3](#_Toc74822458)

[指令格式 3](#_Toc74822459)

[Lanes 3](#_Toc74822460)

[打包設定 4](#_Toc74822461)

[情境 5](#_Toc74822462)

[全部打包 5](#_Toc74822463)

[部分打包 6](#_Toc74822464)

[打包結果 6](#_Toc74822465)

[遇到問題 6](#_Toc74822466)

# 環境準備

## Xcode

用途：fastlane 的環境準備可以看成是在安裝 Ruby on Rails, 這個是必不可少的前置準備

動作：從 App Store 搜尋 Xcode 之後安裝

## Command Line Tools

用途： 某些版本的 Xcode 不會自動安裝這個工具，又或者版本太舊可能會衍生其他問題

動作：

|  |
| --- |
| $ xcode-select –install |

安裝完成後

|  |
| --- |
| $ xcode-select –p |

如果出現

/Applications/Xcode.app/Contents/Developer

就是完成了

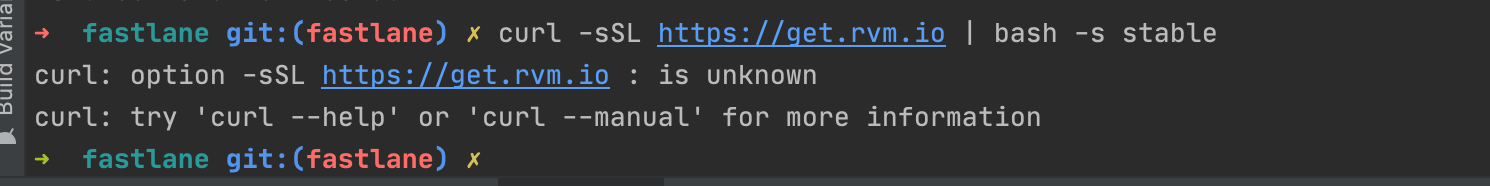
## RVM

用途：管理 Ruby 版本，讓專案內使用的 Ruby 可以不受外部影響

動作：

|  |
| --- |
| $ curl -sSL [https://get.rvm.io](https://get.rvm.io/) | bash -s stable |

若遇到問題如下圖



可參考 <https://rvm.io/rvm/install> 說明

試試看

|  |
| --- |
| $ curl https://raw.githubusercontent.com/rvm/rvm/master/binscripts/rvm-installer | bash -s stable |

## Ruby 2.6.3

用途：成套搭配，目前使用 2.6.3

動作：

|  |
| --- |
| $ rvm install 2.6.3 |

## Gemset

用途：創立一個乾淨的 gem，後續 bundler 在裝套件時比較不會有問題

動作：

這裡不需要特別動作，尤其對於 gemset 怎麼取名字沒有特別要求時，

只要在 terminal 下切換到專案資料夾時就會自動切換成 2.6.3@android。

2.6.3 是先前下載好的版本，android 是我們在這個路徑預設使用的 gemset 對象。

若之後想改 ruby 版本，可修改 {project root}/.ruby-version

若想改 gemset 的使用對象，可修改 {project root}/.ruby-gemset

## Bundler

用途：搭配已預先導入到專案內的 Gemfile ，安裝獨立的套件組給專案用

動作：

|  |
| --- |
| {project root} $ bundle install --path vendor/bundler |

# 執行

## 指令格式

bundle exec fastlane {lane\_name} argu1:value1...

lane\_name 是想要執行的動作

argu1:value1 argu1 是參數名，value1是值的內容，看到實際範例時會更清楚。

## Lanes

Fastlane 所使用的 Fastfile，他的設計意圖是讓我們可以定義多種情境下相似但細微差異的動作。

每個動作的單位就是 lane，按照我們的需求，我在 Fastfile 裡定義了以下 lanes：

1. build\_all\_app

依照目前需求 Android 打包只有一個 Lane，

詳細的參數使用後面會說明。

# 打包設定

使用 config.json 取代原本的 build.gradle 來設定打包。

換句話說，使用 fastlane 進行包版時，會使用 config.json 的資料。

而不會依賴到 build.gradle 裡的設定。

所以之後使用 fastlane 包新平台時，需要把資料填進 config.json 裡。

Config.json 結構如下：

|  |
| --- |
| {  "dictionary\_version": "1.0.12",  "short\_version": "1",  "platform": [  {  "app\_code": "spplat1",  "appid": "com.happysport.sl.test",  "app\_name": "1bo sports",  "jpush\_appKey": "65670407e46038b18ee836ed"  }  ]  } |

|  |  |
| --- | --- |
| Property Name | Description |
| dictionary\_version | APP 版本號，相當於gradle 裡的 versionName |
| short\_version | APP 版本號，相當於gradle 裡的versionCode |
| app\_code | 渠道 ID |
| appid | APP ID |
| app\_name | APP 安裝後顯示名稱 |
| jpush\_appKey | 極光推送的 key |

打包前先改好 config.json，就可得到想要的結果。

# 情境

## 全部打包

因為是使用 bundle 管理，所以現在會變成開頭先 bundle exec ，

後面再用 fastlane 去指定一個 lane 執行

|  |
| --- |
| {project root} $ bundle exec fastlane build\_all\_app |

build\_all\_app 在不指定平台時，會產出所有的 IPA 檔。

## 部分打包

可以在 lane 後加入參數 platforms:{平台集合}

來指定要打包的平台。

舉例：

|  |
| --- |
| {project root} $ bundle exec fastlane build\_all\_app platforms:plat1,vnbf,vnyw |

這裡有加一點容錯機制，遇到 332ncd 這種不對的 app\_code 會自動被略過。

要注意的地方是逗號需相連，不相連 fastlane 馬上就噴 error 給你看。

如果想奔放一點來指定打包的平台，就在前後加上 double qoute。

例如：

|  |
| --- |
| {project root} $ bundle exec fastlane build\_all\_app platforms:"plat1 , vnbf , vnyw" |

就是可以正常執行的方式。

## 打包結果

跟之前用 gradle 打包一樣，放在 {project root}/app/build/outputs/releaseAPK

# 遇到問題

雖然我為了避免每個人環境差異或版本不同會帶來奇怪的結果，

所以可以先設定的，先導入的管理行為都已經執行了。

當然我想像中的順利不一定就是你的現實，

如果遇到問題，可以使用各種方式找到我。

eddy\_ios