在 Android 8.0.0 (Oreo) 及更高版本中，编译版本号的格式 **PVBB.YYMMDD.bbb[.Cn]**，其中：

* **P** 表示平台版本代号的第一个字母，例如 **O** 表示 Oreo。
* **V** 表示支持的行业。按照惯例，**P** 表示主要平台分支。
* **BB** 是由字母和数字组成的代码，Google 可通过该代码识别相应细分版本所属的确切代码分支。
* **YYMMDD** 表示相应版本从开发分支细分出来或与开发分支同步的日期。它并不一定是细分版本的确切构建日期，因为 Google 常常会在现有细分版本中增加细微的更改，并在新细分版本中重复使用与现有细分版本相同的日期代码。
* **bbb** 表示具有相同日期代码的不同版本，从 **001** 开始。
* **Cn** 是可选的字母数字，表示在现有 PVBB.YYMMDD.bbb 细分版本之上编译的修补程序，从 **A1** 开始。

<https://juejin.im/post/5cc80360f265da037c7cf20e>

匹配度很高哦

<http://blog4jimmy.com/2018/02/418.html>

<https://c55jeremy-tech.blogspot.com/2019/04/aosppixel-2-romrom.html>

https://c55jeremy-tech.blogspot.com/2019/04/aosppixel-2-romrom.html窗体顶端

窗体底端

Android平台Platform及Framework相關的技術分享。也包含AOSP相關的技術研究分折。包含Gerrit及Git Repo的使用心得。

### 完整指引如何編譯AOSP (Build Android P)，整合GMS及刷機 (Pixel 2)

- [4月 12, 2019](https://c55jeremy-tech.blogspot.com/2019/04/aosppixel-2-romrom.html)

## 前言

本文是說明一個Android Framework研發人員，如何利用Google的Pixel手機進行系統程式的客製化。本文的工作如下：

* 解鎖Pixel2
* 取得AOSP原始碼
* 編譯AOSP （Build AOSP）
* 將Pixel 2刷機自己編譯的ROM
* 修改Framework後，重編譯，再刷機，完成！
* 安裝GMS

本文針對Pixel 2進行刷機，理論上Pixel2 XL/Pixel 3/Pixel 3 XL應該都適用。

## 前置工作

### 解鎖Pixel 2，讓手機允許刷機

首先必須知道，官方釋出的手機的開機載入程序（Bootloader）會阻止你刷入任何客製ROM。所以所謂「解鎖」就是解鎖Bootloader，讓它允許你可以進行刷機動作。  
其次要注意是在刷機後，所有使用者資料（User data）都會消失。所以刷機前要先備份重要資料。  
  
網路上有很多教解鎖Pixel手機的文章，都可以參考，實際上在Android官網已經有教導你如何解鎖Pixel 2的Bootloader，我個人是比較習慣直接看官網來做：  
  
<https://source.android.com/setup/build/running>  
  
如果你懶得看一大堆英文，根據官網的描述，以下的命令就可以讓你解鎖Bootloader。

adb reboot bootloader

#等重開機後會看到一個躺著且肚子打開的機器人

fastboot flashing unlock

#進入清除資料階段，螢幕顯示Erasing data...

Copy

所以解鎖其實只有兩個命令：  
  
1. adb reboot bootloader重開機以進入“開機載入程序”，畫面如下

|  |
| --- |
| [https://1.bp.blogspot.com/-EZkD22jRwy4/XK1NGW5I-FI/AAAAAAACt_c/ou3s6PQLwCoRLynx0MTBVAGegq4FKIqUgCLcBGAs/s320/1-1%2BBootloader%2BMode.jpg](https://1.bp.blogspot.com/-EZkD22jRwy4/XK1NGW5I-FI/AAAAAAACt_c/ou3s6PQLwCoRLynx0MTBVAGegq4FKIqUgCLcBGAs/s1600/1-1+Bootloader+Mode.jpg) |
| adb reboot bootloader後的實機畫面 |

2. fastboot flashing unlock進行Bootloader解鎖。

|  |
| --- |
| [Pixel2手機解鎖畫面](https://1.bp.blogspot.com/-ArGSL5Pt7PY/XK1Sp6U0FxI/AAAAAAACt_o/w8tP1yczo5gNesPKpkuq-kTnDiuzVg01gCLcBGAs/s1600/1-2+Flashing+unlock.jpg) |
| fastboot flashing unlock後的畫面，按音量上鍵後按電源鍵，確認解鎖 |

手機進入解鎖畫面後，按音量鍵選擇「UNLOCK THE BOOTLOADER」,進行解鎖。  
之後手機進行資料清除後重開。  
  
很多文章寫了一大堆步驟，主要是為了要取得Root權限。但在本文中這是不必要的，因為等你刷入了自己Build出來的ROM，自然可以取得Root權限。

### 刷入官網的ROM

先刷一次官網的ROM，目的有兩個：1. 確定解鎖後的Pixel2可以刷機。 2. 刷特定版本的Pixel2 官方ROM，以確保Kernel和之後我們要編譯的系統映像（System Image）是在同一版本。  
  
官網的ROM的位置在：  
<https://developers.google.com/android/images#walleye>  
  
Walleye就是Pixel2的代號，這很重要，因為會牽涉到之後取得AOSP原始碼時所要選擇的原始碼分枝 （Branch）。在本文中，下載的版本為：

**PQ2A.190205.002**

|  |
| --- |
| [選擇PQ2A.190205.002](https://3.bp.blogspot.com/-9XiV8GottlM/XK2xIufYMEI/AAAAAAACuAw/Weu_U3cKuDIp5gegsbi6_TkLlAuj6HAVgCLcBGAs/s1600/1-2+Official+Pixel2+ROM.png) |
| 選擇PQ2A.190205.002做為官方ROM刷機 |

記得你要刷機的版本，之後一致的版本可以讓你少走一些冤枉路。之後你想要取得更新的版本當然也沒問題，只要確保 **官方ROM**和**AOSP Branch原始碼版本**及**Binary Driver 的版本**一致就可以。  
  
  
下載後解開tgz，一樣進入Bootloader模式 (adb reboot bootloader)，執行fastboot.sh (或在Windows下執行fastboot.bat），則Script開始執行刷機的動作。

adb reboot bootloader

./flash-all.sh

< waiting for any device >

target reported max download size of 536870912 bytes

Sending 'bootloader\_a' (38680 KB)...

OKAY [ 1.421s]

Writing 'bootloader\_a'...

(bootloader) Updating: partition:0 @00002000 sz=0000B000

(bootloader) Updating: partition:1 @0000D000 sz=0000B000

(bootloader) Updating: partition:2 @00018000 sz=0000B000

(bootloader) Updating: partition:3 @00023000 sz=0000B000

...

Sending 'userdata' (4497 KB)...

OKAY [ 0.163s]

Writing 'userdata'...

OKAY [ 0.000s]

Rebooting...

Finished. Total time: 166.501s

Copy

順利的話，完成刷機動作，系統重新開機。此時你進入 [系統]->[關於手機] 的畫面

|  |
| --- |
| [https://4.bp.blogspot.com/-siB77MCeiJU/XLASDxBc0sI/AAAAAAACuFE/vP7m0W0lDbYF9eFv9aFjkvs__lnOSAoTgCLcBGAs/s320/1-2%2BAbout%2BPhone%2BR32.png](https://4.bp.blogspot.com/-siB77MCeiJU/XLASDxBc0sI/AAAAAAACuFE/vP7m0W0lDbYF9eFv9aFjkvs__lnOSAoTgCLcBGAs/s1600/1-2+About+Phone+R32.png) |
| |  | | --- | | 刷機後在系統》關於手機中，確認Build Number與下載ROM一致 | |

你可以看到版本號碼（Build number）為**PQ2A.190205.002**，這和我們從Android官網取得的官方ROM一致。

## 取得AOSP原始碼

### 確認AOSP Branch/Tag

AOSP有很多Branch，那一個Branch才是Android P Pixel2必須先找到。  
  
首先根據Android官網指令，把編譯環境設定好，確保你可以編譯Android AOSP：  
<https://source.android.com/setup/build/initializing>  
  
之後要選定AOSP版本及分枝，在以下官方連結中，列出了所有的分枝  
<https://source.android.com/setup/start/build-numbers.html#source-code-tags-and-builds>  
  
記得我們是選定**PQ2A.190205.002**這個版本來實作，所以可以找到對映的分枝：android-9.0.0\_r32

|  |
| --- |
| [https://1.bp.blogspot.com/-5Wza28iffwc/XK2wG9DWurI/AAAAAAACuAo/MYpcTJB7KlMN3PPldnaajIqqHMl5ulI1wCLcBGAs/s640/1-3%2BAOSP%2BBranch%2Bfor%2BPixel2%2BR32.png](https://1.bp.blogspot.com/-5Wza28iffwc/XK2wG9DWurI/AAAAAAACuAo/MYpcTJB7KlMN3PPldnaajIqqHMl5ulI1wCLcBGAs/s1600/1-3+AOSP+Branch+for+Pixel2+R32.png) |
| |  | | --- | | PQ2A.190205.002對映的Tag為android-9.0.0\_r32 | |

### 下載AOSP Android原始碼 （及私有函式庫） 所以找到的AOSP是android-9.0.0\_r32，現在我們就根據官網來下載這個原始碼： <https://source.android.com/setup/build/downloading> 順著官網執行設定工作後，最重要就是下載AOSP原始碼了。主要的不同也只有在這一行

repo init -u https://android.googlesource.com/platform/manifest -b android-9.0.0\_r32 #-b是branch取得android-9.0.0\_r32這個分枝

repo sync #開始下載

Copy

下載完成（視網路狀況，資料量約90Gbytes）後，接下來下載Pixel2 Driver。  
為什麼還有Driver? 因為AOSP是完成Open Source，但實機上有些原始碼是私有Proprietary的，受版權保護及有安全疑慮而不放開原始碼。例如和無線通訊／圖形加速晶片相關，這些是以Binary形態存在，所以必須也要下載，我們才能完整編譯出一個可以刷入實機的ROM。（否則只能編譯Emulator而已）。  
  
Pixel2 (Walleye) 的Binary Driver在： <https://developers.google.com/android/drivers#walleye>  
  
同樣我們必須保持版本一致。找到**PQ2A.190205.002**這個版本

|  |
| --- |
| [https://1.bp.blogspot.com/-UyVrlXIAEt0/XK2x04F3-RI/AAAAAAACuA8/UWbKiJVjcUkfazOpOY-XMhPWT9SkKoF2ACLcBGAs/s640/1-5%2BDriver%2BBinaries%2BR32.png](https://1.bp.blogspot.com/-UyVrlXIAEt0/XK2x04F3-RI/AAAAAAACuA8/UWbKiJVjcUkfazOpOY-XMhPWT9SkKoF2ACLcBGAs/s1600/1-5+Driver+Binaries+R32.png) |
| PQ2A.190205.002有兩個Binary包，都要下載 |

我們看到兩個Binary，一個是Vendor Image，另一個是Driver。這兩個都是必要的，都下載並且存放至你剛剛下載AOSP的目錄下。以下命令是使用wget取回Pixel2 DriverBinary。

wget https://dl.google.com/dl/android/aosp/google\_devices-walleye-pq2a.190205.002-a5b43ffa.tgz

--2019-04-10 17:07:08-- https://dl.google.com/dl/android/aosp/google\_devices-walleye-pq2a.190205.002-a5b43ffa.tgz

Resolving dl.google.com (dl.google.com)... 172.217.24.14, 2404:6800:4008:801::200e

Connecting to dl.google.com (dl.google.com)|172.217.24.14|:443... connected.

HTTP request sent, awaiting response... 200 OK

Length: 153576317 (146M) [application/x-gtar]

Saving to: ‘google\_devices-walleye-pq2a.190205.002-a5b43ffa.tgz’

100%[==============================================================================================================================================>] 153,576,317 31.6MB/s in 4.6s

2019-04-10 17:07:12 (31.9 MB/s) - ‘google\_devices-walleye-pq2a.190205.002-a5b43ffa.tgz’ saved [153576317/153576317]

wget https://dl.google.com/dl/android/aosp/qcom-walleye-pq2a.190205.002-d5cc3341.tgz

--2019-04-10 17:07:19-- https://dl.google.com/dl/android/aosp/qcom-walleye-pq2a.190205.002-d5cc3341.tgz

Resolving dl.google.com (dl.google.com)... 172.217.24.14, 2404:6800:4008:801::200e

Connecting to dl.google.com (dl.google.com)|172.217.24.14|:443... connected.

HTTP request sent, awaiting response... 200 OK

Length: 1196171 (1.1M) [application/x-gtar]

Saving to: ‘qcom-walleye-pq2a.190205.002-d5cc3341.tgz’

100%[==============================================================================================================================================>] 1,196,171 --.-K/s in 0.04s

2019-04-10 17:07:20 (26.3 MB/s) - ‘qcom-walleye-pq2a.190205.002-d5cc3341.tgz’ saved [1196171/1196171]

2019-04-10 11:30:02 (25.1 MB/s) - ‘qcom-walleye-pq1a.181205.002-c199ac9d.tgz’ saved [1196259/1196259]

Copy

接下來直接用tar解開，會解出兩個script：

tar xvfz google\_devices-walleye-pq2a.190205.002-a5b43ffa.tgz

extract-google\_devices-walleye.sh

tar xvfz qcom-walleye-pq2a.190205.002-d5cc3341.tgz

extract-qcom-walleye.sh

Copy

然後執行兩個script，解出Binary：

./extract-google\_devices-walleye.sh

./extract-qcom-walleye.sh

Copy

它會要你看完版權宣告後，輸入 I ACCEPT，才真正解開來，解出的Binary都存在vendor目錄下。

[](https://3.bp.blogspot.com/-XJZM5ftv-_g/XK2QVxvKwUI/AAAAAAACuAQ/H0ML4QxaZf8V7T1V7QDW1a6he0msFqloQCLcBGAs/s1600/1-5+Binary+Drivers.png)

可以看到google\_devices目錄中有個vendor.img，這是OEM廠商私有的映像檔，主要包含和硬體相關的程式及函式庫，以及硬體啟動時所需的設定檔。  
qcom表示Qualcomm，看起來這些Binary主要是和IMS （[IP Multimedia Subsystem](https://zh.wikipedia.org/wiki/IP%E5%A4%9A%E5%AA%92%E4%BD%93%E5%AD%90%E7%B3%BB%E7%BB%9F)）相關的函式庫以及框架層所需的JAR檔。  
  
完全不懂Binary Drivers也無所謂，知道這些檔案是必須的就行了。  
至此所需的檔案和原始碼已完全下載完畢。再來接下來就可以開始編譯AOSP了。

## 進行編譯AOSP

首先建立環境變數，就是告訴AOSP要如何編譯這整包原始碼，你可以把它編輯成模擬器Emulator也可以編譯成Pixel2的ROM。  
我們設定為編譯Pixel2 ROM

sh ./build/envsetup.sh

including device/generic/car/vendorsetup.sh

including device/generic/mini-emulator-arm64/vendorsetup.sh

...

including sdk/bash\_completion/adb.bash

lunch aosp\_walleye-userdebug

============================================

PLATFORM\_VERSION\_CODENAME=REL

...

OUT\_DIR=out

============================================

Copy

編譯環境的建立就兩個命令 sh ./build/envsetup.sh建立通用環境變數，而lunch aosp\_walleye-userdebug則是指定之後下make編譯時，編譯出符合Pixel2的映像檔（image）。  
  
那如果要編譯Pixel 3呢？你可以到官網：<https://source.android.com/setup/build/running#selecting-device-build>

|  |
| --- |
| [https://2.bp.blogspot.com/-DjflnqKhjEk/XK2VookI9oI/AAAAAAACuAc/I1SxabfgVoIvsOXD6ufv6z6EeQdYgp9IwCLcBGAs/s640/1.6%2BAOSP%2BBuild%2Bconfiguration.png](https://2.bp.blogspot.com/-DjflnqKhjEk/XK2VookI9oI/AAAAAAACuAc/I1SxabfgVoIvsOXD6ufv6z6EeQdYgp9IwCLcBGAs/s1600/1.6+AOSP+Build+configuration.png) |
| Google硬體對映Build configuration （編譯組態） |

所以如果你的硬體是Pixel 3，則命令就是lunch aosp\_blueline-userdebug。  
  
之後下make就開始編譯了。  
所以完整下編譯的命令為：

#建立通用環境

sh ./build/envsetup.sh

#指定編譯組態

lunch aosp\_walleye-userdebug

#開始編譯，時間視PC速度而定

make -j16

Copy

一切順利後完成編譯，輸出目錄在環境變數$ANDROID\_PRODUCT\_OUT所指定的位置。你可以直接下cd $ANDROID\_PRODUCT\_OUT到輸入目錄。  
  
再來檢視你編譯出來的結果：

cd $ANDROID\_PRODUCT\_OUT

ls -al \*.img

-rw-rw-r-- 1 user user 33554432 4月 10 14:26 boot.img

-rw-rw-r-- 1 user user 8388608 4月 10 12:04 dtbo.img

-rw-rw-r-- 1 user user 1569523 4月 10 14:26 ramdisk.img

-rw-rw-r-- 1 user user 8720084 4月 10 14:26 ramdisk-recovery.img

-rw-rw-r-- 1 user user 1175118152 4月 10 14:29 system.img

-rw-rw-r-- 1 user user 84722016 4月 10 14:28 system\_other.img

-rw-rw-r-- 1 user user 4952764 4月 10 14:20 userdata.img

-rw-rw-r-- 1 user user 4096 4月 10 14:29 vbmeta.img

-rw-r--r-- 1 user user 370045136 4月 10 12:04 vendor.img

Copy

可以比較一下你編譯出來的映像檔(\*.img)是否相同。 完成編譯後，接下來就是刷機了。

## 用自己編譯的ROM來刷機

最後就是刷機，刷機就簡單了，先下adb reboot bootloader重新將手機進入Bootloader，再執行fastboot flashall

#進入Bootloader Mode

adb reboot bootloader

#進行刷機

fastboot flashall -w

Copy

刷機完成後會完全清除使用者資料，如果想要保留資料，則之後刷機不加-w參數。  
刷機後開機，如果一切順利，你會發現，怎麼和官方ROM的畫面完全不同？？官方的ROM是包含GMS，而自己編譯的ROM則沒有GMS，所以自己編譯的ROM看起來就很像開發Android APP的模擬器。

|  |
| --- |
| [自行編譯Build Pixel2 AOSP ROM並刷機，沒有GMS](https://1.bp.blogspot.com/-7sHbU5R8cjQ/XK249xN7A8I/AAAAAAACuBI/QHML4WoepNsLK-RV6BwFKu3C6AH8wZkCACLcBGAs/s1600/1-6+My+ROM.jpg) |
| Pixel2刷自製ROM後，沒有GMS的主畫面 |

## 修改AOSP

### 實作耳機圖示

我們來做簡單的一個小功能（其實只是小改一下），功能為「插入耳機時，在狀態列Status Bar上顯示耳機圖示小Icon」,像這樣：

|  |
| --- |
| [Android P顯示耳機圖示](https://2.bp.blogspot.com/-yHSstimmj4I/XKRdvsdyLyI/AAAAAAACt4Y/xOJX24jn7YgIfndNK_OWaHWjcyL4QIasACLcBGAs/s1600/3-3+Headset+on+Status+bar.png) |
| 耳機插入小圖示 |

你可以看到在插入耳機後，在Wi-Fi信號左邊一個耳機圖示。這是在耳機插入時才會顯示。在Google的原生AOSP設計中，預設是不顯示這個Icon。  
  
在Android 8.0（Android Oreo）時，你可以下拉Status Bar後，長按設定鍵（齒輸小Icon），進入System Tuner UI，然後手動將耳機Headset圖示開啟。但Android 9.0 （Android Pie）之後，System Tuner的功能預設關掉了，所以我們的工作根本不用開發什麼，就是把它找回來就可以了！！  
  
首先開啟：  
frameworks/base/packages/SystemUI/src/com/android/systemui/statusbar/phone/SettingsButton.java  
把第三行的Code從false改為true：

public class SettingsButton extends AlphaOptimizedImageButton {

private static final boolean TUNER\_ENABLE\_AVAILABLE = true; //原始AOSP的值為false

private static final long LONG\_PRESS\_LENGTH = 1000;

private static final long ACCEL\_LENGTH = 750;;

Copy

再重新下make命令再Build一次，然後重刷一次機。完成後下拉Status Bar，長按右下角的設定鍵會發現小齒輪一直轉動（如下圖指標所示），放開後進入Settings畫面。

|  |
| --- |
| [開啟Android P System UI Tunner設定](https://3.bp.blogspot.com/-xiU-7PFIhZo/XKRwEIwq78I/AAAAAAACt4k/t97e_xNDkU4O8RTivwLLQTs8Nha-Kp1_wCLcBGAs/s1600/3-5+Setting+Button+rolling.png) |
| 開啟System UI Tuner，客製圖示顯示設定 |

進入Settings之後，進入System->Advanced->System UI Tunner->Status bar後，開啟Headset。  
再找個耳機插入USB Type-C孔，就會看到耳機圖示出現了。

### 把GSM整合進你的Codebase！

畢竟純AOSP對於一般使用者是不用夠，無法做為日常使用，所以必須有GMS的套件，才算完整。接下來我們就把GSM整合回來！

GMS就是Google Map/Chrome/Youtube等等Google軟體，是Google私有的套件。只有和Google有合約的手機廠商可以預載GMS。所以雖然我們編譯AOSP是Android官方釋出，並不會包含GMS預載。這造成了一個很矛盾的問題，第三方ROM，例如Lineage OS，很受歡迎，對於推廣Google服務很有幫助，但它並不是有合約的手機廠商，所以拿不到GMS? 可是他又有也有可以刷GMS，難道沒侵權啊？！  
  
理論上Google為了要同時保持開放性及服務可靠性，所以算是放水。它讓解鎖後的手機可對Android系統安裝GMS。而重點在「解鎖後」，所以Google不會告你侵權，只要不是預載在市售ROM就可以。於是Open GAPP的計劃就產生了，它讓第三方ROM可以使用GMS及其服務。Open Gapps官網：<https://opengapps.org/>  
  
一般像是LineageOS這種第三方ROM會教你如何刷GSM進去 （通常是TWRP加上OpenGapps）。不過要整進AOSP有點特別，AOSP預設「沒有」GMS。整個AOSP Codebase並沒有和GMS整合的相關編輯設定。如果直接用TWRP來刷GMS並不是不行，但在Android系統權限和WebView設定等等方面，都會出問題。  
  
OpenGapps也有教你如何把GMS整合進整個原始碼  
 <https://github.com/opengapps/aosp_build> (但這說明有些不完整，有幾個Issue我要Submit給作者，請他修正了)  
  
**1. 安裝lunzip**  
首先安裝lunzip，這是Open Gapps在整合AOSP所需的命令工具。

sudo apt-get install lunzip

Copy

**2. 修改manifest.xml**  
  
將Open Gapps的專案下載至你的AOSP Codebase。（這些Open Gapps專案很大，約20G，因為他是一個專案支援所有Android版本）  
  
在.repo/manifest.xml的最後，把以下的Open Gapps專案Projects加入

<remote name="opengapps" fetch="https://github.com/opengapps/" />

<remote name="nezor" fetch="https://gitlab.nezorfla.me/opengapps/" />

<project path="vendor/opengapps/build" name="aosp\_build" revision="master" remote="opengapps" />

<project path="vendor/opengapps/sources/all" name="all" clone-depth="1" revision="master" remote="nezor" />

<!-- arm64 depends on arm -->

<project path="vendor/opengapps/sources/arm" name="arm" clone-depth="1" revision="master" remote="nezor" />

<project path="vendor/opengapps/sources/arm64" name="arm64" clone-depth="1" revision="master" remote="nezor" />

<project path="vendor/opengapps/sources/x86" name="x86" clone-depth="1" revision="master" remote="nezor" />

<project path="vendor/opengapps/sources/x86\_64" name="x86\_64" clone-depth="1" revision="master" remote="nezor" />

Copy

接下來在.repo/mainfests/目前下，執行git commit -a -m "add open gapps"，將修改commit進git。  
  
**3. 修改aosp\_walleye.mk及device.mk**  
  
修改檔案：device/google/muskie/aosp\_walleye.mk，將：  
PRODUCT\_RESTRICT\_VENDOR\_FILES := **owner**  
改為  
PRODUCT\_RESTRICT\_VENDOR\_FILES := **false**  
  
否則在編輯時會發生

error: Error: Product "aosp\_walleye" cannot have overlay in vendor tree:...

再修改檔案：device/google/wahoo/device.mk，在檔案最後加入  
  
GAPPS\_VARIANT := stock  
$(call inherit-product, vendor/opengapps/build/opengapps-packages.mk)

編譯AOSP時，就會呼叫vendor/opengapps/build/opengapps-packages.mk將Open Gapps編入system.img中  
  
**4. 修改原始碼**  
修改frameworks/base/services/core/java/com/android/server/wm/WindowManagerService.java

--- a/services/core/java/com/android/server/wm/WindowManagerService.java

+++ b/services/core/java/com/android/server/wm/WindowManagerService.java

@@ -6001,8 +6001,8 @@ public class WindowManagerService extends IWindowManager.Stub

@Override

public void setShelfHeight(boolean visible, int shelfHeight) {

- mAmInternal.enforceCallerIsRecentsOrHasPermission(android.Manifest.permission.STATUS\_BAR,

- "setShelfHeight()");

+ //mAmInternal.enforceCallerIsRecentsOrHasPermission(android.Manifest.permission.STATUS\_BAR,

+ // "setShelfHeight()");

synchronized (mWindowMap) {

getDefaultDisplayContentLocked().getPinnedStackController().setAdjustedForShelf(visible,

shelfHeight)

Copy

把setShelfHeight所需要的權限移除，否則刷入ROM之後Pixel Launcher會因權限不足發生錯誤。  
  
**5. 重編譯及刷機**  
再下make之後，重新刷機，結果如下：

|  |
| --- |
| [自行媥譯AOSP Android P並刷機進Pixel2](https://3.bp.blogspot.com/-6fU6vSfIBwI/XLARB9ma8_I/AAAAAAACuE8/2TKj16WIzCUCPgCfCzdWULdNIM7dsGI3ACLcBGAs/s1600/1-8+Final+AOSP+with+GMS.jpg) |
| 刷入最後自行客製ROM，AOSP Android P + Open Gapps |

如此完成了一次，從取得AOSP，編譯，刷機，修改，加入GMS，再編譯，再刷機的完整工作！

## 最後一哩

有了完整的AOSP + GMS，其實還缺內核原始碼的部份。因為Pixel的內核在AOSP中採取預編內核，在device/google/wahoo-kernel下可以看到預編的內核（prebuilt kernel）。如果是內核開發者必須更動其原始碼，最必須也將內核核心整合進來。詳細方法可以參考：[如何編譯AOSP內核Kernel並整合進AOSP （Android P/Pixel 2)](https://c55jeremy-tech.blogspot.com/2019/04/aospkernelaosp.html)

[客製ROM](https://c55jeremy-tech.blogspot.com/search/label/%E5%AE%A2%E8%A3%BDROM) [解鎖手機](https://c55jeremy-tech.blogspot.com/search/label/%E8%A7%A3%E9%8E%96%E6%89%8B%E6%A9%9F) [編譯AOSP](https://c55jeremy-tech.blogspot.com/search/label/%E7%B7%A8%E8%AD%AFAOSP) [AOSP Build](https://c55jeremy-tech.blogspot.com/search/label/AOSP%20Build) [Opengapps](https://c55jeremy-tech.blogspot.com/search/label/Opengapps)

位置： [台灣台北](https://maps.google.com/maps?q=%E5%8F%B0%E7%81%A3%E5%8F%B0%E5%8C%97@25.0329694,121.56541770000001&z=10)

### 留言

1. https://lh3.googleusercontent.com/zFdxGE77vvD2w5xHy6jkVuElKv-U9_9qLkRYK8OnbDeJPtjSZ82UPq5w6hJ-SA=s35

[**Unknown**](https://www.blogger.com/profile/11110879170408655073)[*2019年5月15日 下午10:54*](https://c55jeremy-tech.blogspot.com/2019/04/aosppixel-2-romrom.html?showComment=1557932067979#c4496184782862537226)

跟下面的整合内核Kernel一起，这个帖子给我示范了一个完整的AOSP 编译与刷机过程。是个好教程。  
另外请教有个问题。我现在基于Android Preview(即Android Q的预览版)在写一个东西，但我已经忘记这个Q的源码是如何得到的，更不知道它对应于Walleye的Driver Binary在哪里，所以我现在遇到了一个十分棘手的问题：无法成功将我所编写的软件基于Q来刷Pixel 2的真机。请问，有办法吗？

[**回覆**](javascript:;)

* 1. https://lh3.googleusercontent.com/zFdxGE77vvD2w5xHy6jkVuElKv-U9_9qLkRYK8OnbDeJPtjSZ82UPq5w6hJ-SA=s35

[**Jeremy Chen**](https://www.blogger.com/profile/00985953479608252145)[*2019年5月16日 上午11:14*](https://c55jeremy-tech.blogspot.com/2019/04/aosppixel-2-romrom.html?showComment=1557976466411#c3929045793793357280)

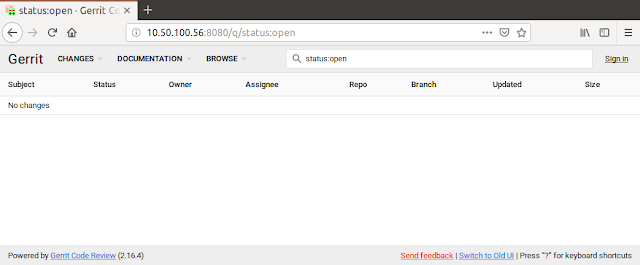
取得AOSP Q Preview源碼是容易的，如果沒錯，我看了一下，應該是用 "repo init -u https://android.googlesource.com/platform/manifest -b android-q-preview-2.5" 來取得。  
但Device Binary就麻煩了，官方還沒釋出，所以只能先刷官方的Android Q Preview之後（https://developer.android.com/preview/download#flash），比較一下AOSP System和官方釋出Binary Image刷機完之後的檔案差別。  
把官方Image多出來的檔案抓出來到vendor/qcom/walleye/proprietary下，官方的Binary Image解開後，原本就有vendor.img了，這部份就容易整合了。  
這只是理論，有空我會試一下，但這就很花時間，所以對我而言，玩AOSP在工作上沒這麼急迫，所以不如等官方發佈出來再來玩。

#### 張貼留言

### 這個網誌中的熱門文章

### [架設Gerrit Server : 架設本地AOSP Gerrit Server完整指引 PART1](https://c55jeremy-tech.blogspot.com/2019/04/gerrit-serveraosp-codebase-part-13.html)

*-*[*4月 08, 2019*](https://c55jeremy-tech.blogspot.com/2019/04/gerrit-serveraosp-codebase-part-13.html)

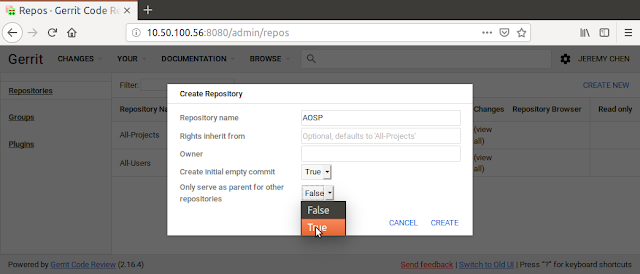
[](https://c55jeremy-tech.blogspot.com/2019/04/gerrit-serveraosp-codebase-part-13.html)

前言 本文引導入如何從無至有，建立一個專屬自己的Gerrit伺服器 (Gerrit Server)  
  
本文是系列文的第一篇，本系列在介紹如何建立本地自有 （localhost） Gerrit Server並且將AOSP （Android Open Source Project）的原始碼完整導入此Gerrit Server。  
  
文章分為三個部份，本文是第一部份，教導如何架設Gerrit Server，如果你已經很熟悉這個工作，可以直接跳至第二或第三部份。  
  
第一部份：架設Gerrit Server （本文）  
第二部份：[移轉整個AOSP至本地Gerrit Server](https://c55jeremy-tech.blogspot.com/2019/04/gerrit-serveraosp-codebase-part-23.html)。  
第三部份：[如何從本地Gerrit Server建立一個AOSP Branch](https://c55jeremy-tech.blogspot.com/2019/02/gerrit-serveraosp-codebase-part-33.html)，  
  
真正利用Gerrit來確保Code的品質並且保留企業內部技術。  
進行此AOSP導入工作的主要目的在於本人專業工作上需要修改AOSP原始碼以符合客製化的需求，但AOSP本身龐大的原始碼及代碼庫（Repository）不適合再導入公司內部原有已存在的Gerrit Server。於是乎，將AOSP導入一個新建的Gerrit Server成為一個合理的選擇。另外更重要的原因是，自身內部專案的AOSP原始碼必須能夠提供以下功能：  
  
進行Code Review，確保穩定性及紀錄留存以保留技術知識。同步Merge Android官方原始碼，確保官方修正的問題可以同步合入自身專案。  
在完成本文系統的三篇文章中所述的任務之後，你應該可以：  
  
建立基本Gerrit Server的能力抓取AOSP並導入自有Gerrit Server的能力同步AOSP並且合入官方修改的原始碼的能力建立自身AOSP Branch並且修改及進行Code Review  
本文在Ubuntu 18.04進行，基本上我在Ubuntu 16.04也執行過同樣任務，所以差別不大，所以你的系統是在Ubuntu 16.04，應該完全可以適用。  
所需軟體[Gerrit](http://gerrit-releases.storage.googleapis.com/index.html) (version 2.16.4)GitOpen JDK 8MySQL (非必要）Gerrit Delete Project Plug-in （非必要，但很好用）  
安裝及設定 因為Gerrit是以Java開發，而且AOSP也需要Java 8做為Java Compilter，我們首先安裝Open JDK 8。 安裝OpenJDK 8 U…

[**閱讀完整內容**](https://c55jeremy-tech.blogspot.com/2019/04/gerrit-serveraosp-codebase-part-13.html#more)

### [將AOSP加入Gerrit Server : 架設本地AOSP Gerrit Server完整指引 PART2](https://c55jeremy-tech.blogspot.com/2019/04/gerrit-serveraosp-codebase-part-23.html)

*-*[*4月 08, 2019*](https://c55jeremy-tech.blogspot.com/2019/04/gerrit-serveraosp-codebase-part-23.html)

[](https://c55jeremy-tech.blogspot.com/2019/04/gerrit-serveraosp-codebase-part-23.html)

前言 本文在引導你如何將AOSP （Android Open Source Project）完整導入企業內部或個人Gerrit伺服器。AOSP包含近800個專案，要如何有效快速無誤的將這些專案導入Gerrit對很多IT人員來說若不知方法會是很頭痛的問題。  
  
本文是系列文的第二篇，本系列在介紹如何建立本地自有 （local host） Gerrit Server並且將AOSP （Android Open Source Project）的原始碼完整導入此Gerrit Server。  
  
系列分為三個部份，本文是第二部份，教導如何建立AOSP Git Mirror並且同步導入自建的Gerrit Server，如果你還不知如何架設一個Gerrit Server，可以回到這系列的  
  
第一部份：[架設Gerrit Server](https://c55jeremy-tech.blogspot.com/2019/04/gerrit-serveraosp-codebase-part-13.html)，  
第二部份：將AOSP完整導入Gerrit Server (本文）。  
第三部份：[如何從本地Gerrit Server建立一個AOSP Branch](https://c55jeremy-tech.blogspot.com/2019/02/gerrit-serveraosp-codebase-part-33.html)。  
  
進行此AOSP導入工作的主要目的在於本人專業工作上需要修改AOSP原始碼以符合客製化的需求，但AOSP本身龐大的原始碼及代碼庫（Repository）不適合再導入公司內部原有已存在的Gerrit Server。於是乎，將AOSP導入一個新建的Gerrit Server成為一個合理的選擇。另外更重要的原因是，自身內部專案的AOSP原始碼必須能夠提供以下功能：  
  
進行Code Review，確保穩定性及紀錄留存以保留技術知識。同步Merge Android官方原始碼，確保官方修正的問題可以同步合入自身專案。  
在完成本文中所述的任務之後，你應該可以：  
  
建立基本Gerrit Server的能力抓取AOSP並導入Gerrit Server的能力同步AOSP並且合入官方修改的原始碼的能力建立自身AOSP Branch並且修改及進行Code Review  
本文在Ubuntu 18.04進行，基本上我在Ubuntu 16.04也執行過同樣任務，所以差別不大，所以你的系統是在Ubuntu 16.04，應該完全可以適用。  
所需軟體[Gerrit](http://gerrit-releases.storage.googleapis.com/index.html) (version 2.16.4)GitOpen JDK 8Apache2 （非必要）MySQL (非必要）Gerrit Delete Project Plug-in （非必要，但很好用）  
概念 AO…

[**閱讀完整內容**](https://c55jeremy-tech.blogspot.com/2019/04/gerrit-serveraosp-codebase-part-23.html#more)

[技術提供：Blogger](https://www.blogger.com/)

主題圖片來源：[Michael Elkan](http://www.offset.com/photos/394244)

[**JEREMY CHEN**](https://www.blogger.com/profile/00985953479608252145)

[瀏覽簡介](https://www.blogger.com/profile/00985953479608252145)

### 封存

### 標籤

### [檢舉濫用情形](https://www.blogger.com/go/report-abuse)

### prism

# 2、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、

<https://www.androidperformance.com/2018/11/01/android-system-develop-0/>

# Android 系统开发源码环境搭建

[Framework](javascript:void(0);) [Ubuntu](javascript:void(0);)

字数统计: 847阅读时长: 3 min

2018/11/01 1051 Shar

不管是 Android App 开发工程师还是 Android 系统开发工程师，对 Android 系统有一定的了解是很有必要的，正好我这几天在家里搭建了个开发环境，把过程分享出来，有需要的人可以看一下。

大概的步骤包含下面几个：

1. 安装 Ubuntu 系统
2. 配置 Ubuntu 系统
3. 安装必备的软件
4. 配置 VPN (可选)
5. 下载 AOSP 代码
6. 配置编译环境并编译 Pixel 的代码
7. 刷机
8. 修改和编译 Framework 、Service、Res

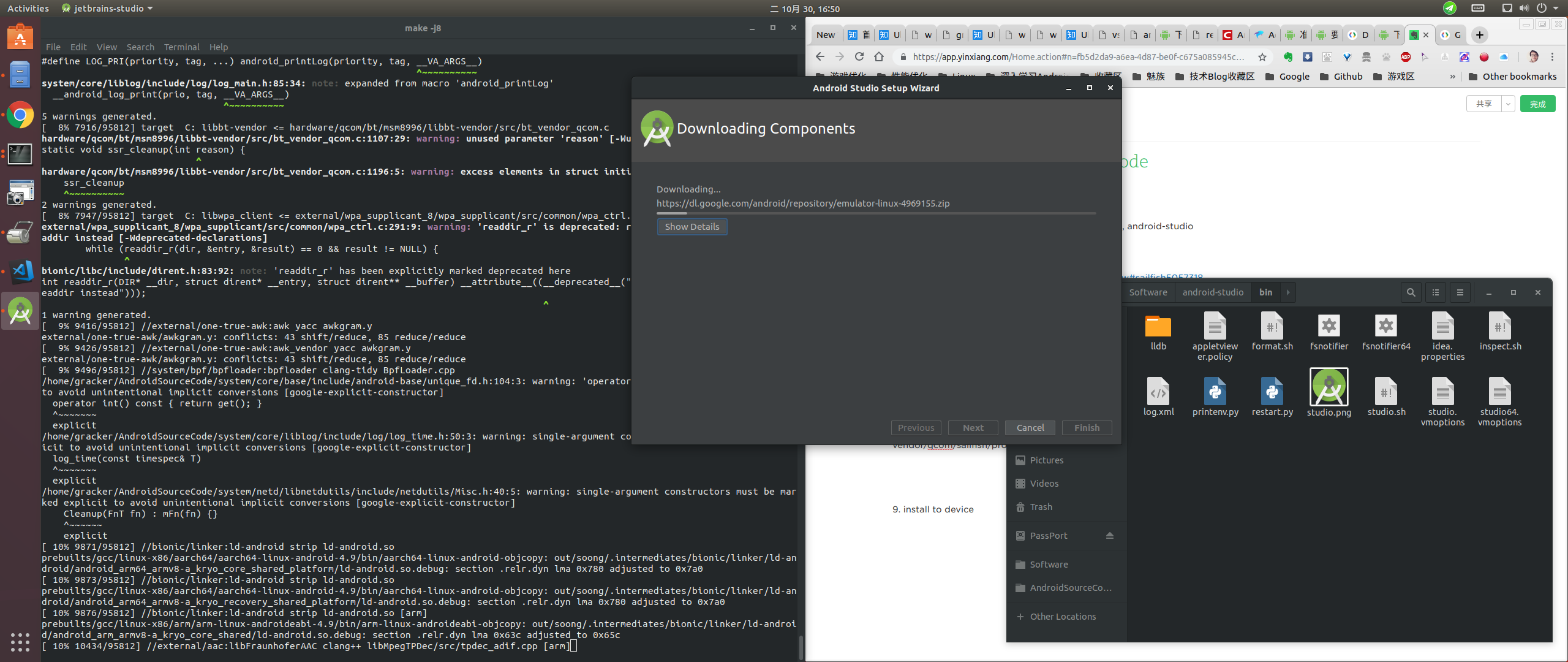
建议准备下面的硬件，当然没有也没关系，有了更好

1. PC 或者笔记本
2. 512GB 的 SSD
3. Pixel 手机一台

# 安装 Ubuntu 系统

Linux 这边我建议用 Ubuntu 系统，不建议用虚拟机，直接安装一个新的 Ubuntu 系统会比较好，Ubuntu 目前最新的 LTS 版本是 18.04，目前安装 Ubuntu 的步骤会比较简单：

1. 下载 Ubuntu18.04 版本:[ubuntu 18.04](http://mirrors.opencas.cn/ubuntu-releases/18.04.1/ubuntu-18.04.1-desktop-amd64.iso)
2. 使用 Ubuntu 推荐的工具做一个 U 盘启动盘
3. 使用 U 盘安装 Ubuntu 系统

[](https://www.androidperformance.com/images/android-system/mydesktop.png)

# 配置 Ubuntu 系统

1. 安装搜狗输入法
2. 安装 vim ：sudo apt install vim
3. 安装 adb
4. 安装 fastboot

# 安装必备的软件

1. VS Code
2. Android Studio
3. Meld
4. Wine
5. WPS

# 配置 VPN (可选)

1. ShadowSocks

# 下载 AOSP 代码

没有 v-p-n 的话，推荐使用清华的镜像站：<https://mirror.tuna.tsinghua.edu.cn/help/AOSP/>

## repo 下载

|  |  |
| --- | --- |
| 1 2 3 4 | mkdir ~/bin PATH=~/bin:$PATH curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo chmod a+x ~/bin/repo |

## 建立工作目录

|  |  |
| --- | --- |
| 1 2 | mkdir WORKING\_DIRECTORY cd WORKING\_DIRECTORY |

## 初始化仓库

|  |  |
| --- | --- |
| 1 | repo init -u https://aosp.tuna.tsinghua.edu.cn/platform/manifest |

## 下载代码（-c –no-tags 能下载更少的代码）

|  |  |
| --- | --- |
| 1 | repo sync -c --no-tags |

# 配置编译环境并编译 Pixel 的代码

## 配置编译环境

### 安装jdk

|  |  |
| --- | --- |
| 1 2 | sudo apt-get update sudo apt-get install openjdk-8-jdk |

### 安装相关依赖

|  |  |
| --- | --- |
| 1 | sudo apt-get install git-core gnupg flex bison gperf build-essential zip curl zlib1g-dev gcc-multilib g++-multilib libc6-dev-i386 lib32ncurses5-dev x11proto-core-dev libx11-dev lib32z-dev libgl1-mesa-dev libxml2-utils xsltproc unzip |

## 下载 Pixle 的驱动

编译 Android Master 的代码的话，需要下载对应的手机的驱动，在这个页面找到自己需要的驱动：  
<https://developers.google.cn/android/blobs-preview>  
[extract-qcom-sailfish](https://www.androidperformance.com/images/android-system/extract-qcom-sailfish.)

解压如下：

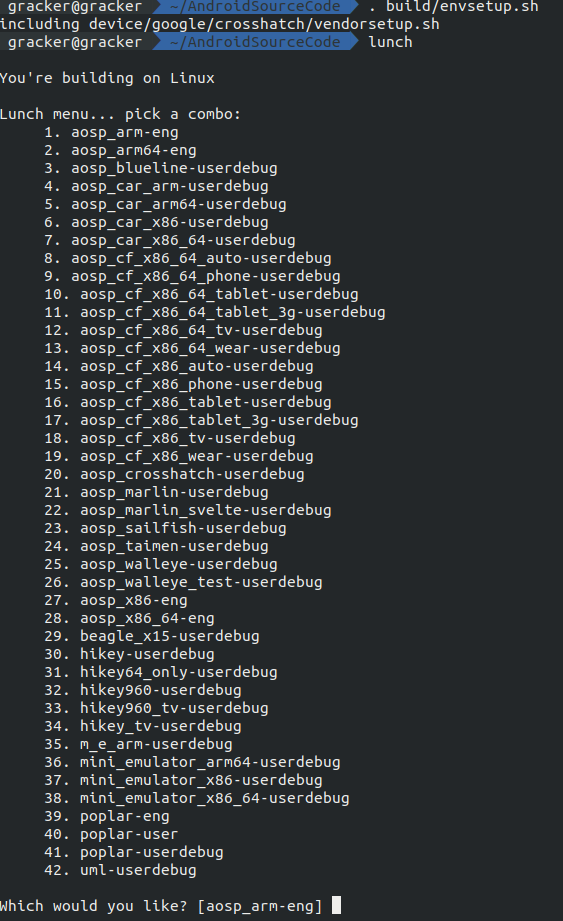
## 编译 Pixle 的系统镜像

在源码根目录下执行

|  |  |
| --- | --- |
| 1 | source build/envsetup.sh |

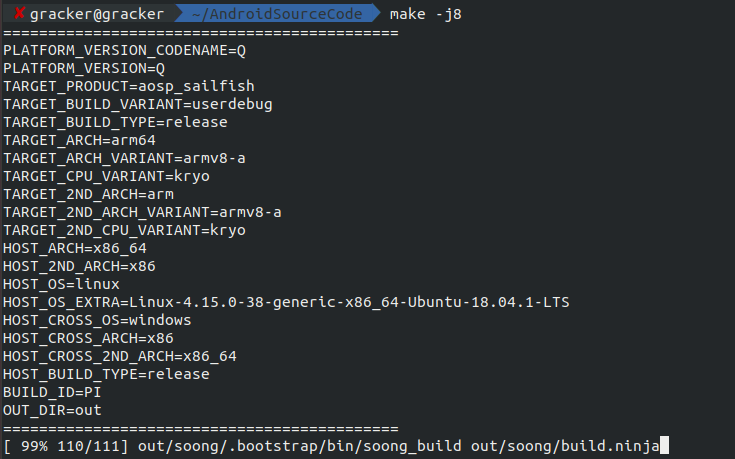
执行下面的命令选择要编译的手机型号和版本（user、userdebug、eng）

|  |  |
| --- | --- |
| 1 | lunch |

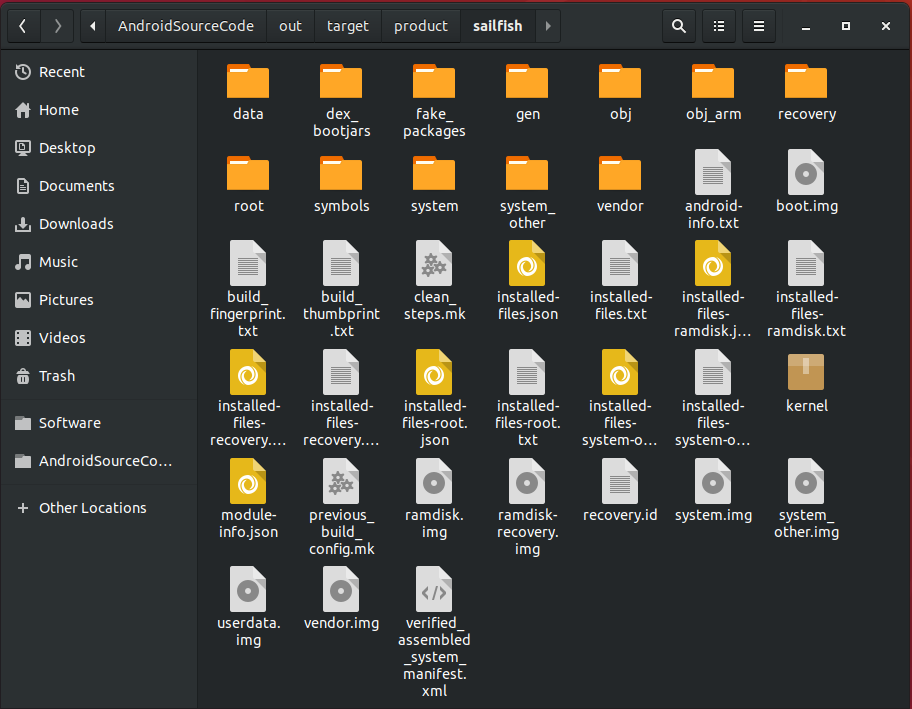
[](https://www.androidperformance.com/images/android-system/lunch.png)

选择好了之后，输入对应的数字或者数字后面的,执行 make 开始编译（可选择加 -j4，4带代表线程数，机器性能好的话可以8或者16，看cpu）：

|  |  |
| --- | --- |
| 1 | make -j8 |

[](https://www.androidperformance.com/images/android-system/make-j8.png)

## 编译成功

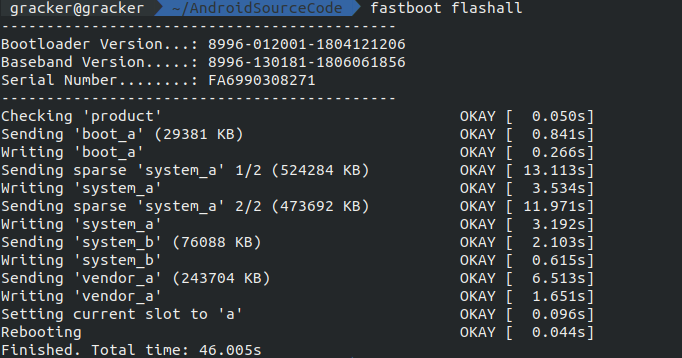
out 目录会生成对应的 image  
[](https://www.androidperformance.com/images/android-system/out_folder.png)

# 刷机

在源码根目录下，执行下面的命令,即可刷入对应的系统到

[fastboot](https://www.androidperformance.com/images/android-system/fastboot.png)

|  |  |
| --- | --- |
| 1 | fastboot flashall |

[](https://www.androidperformance.com/images/android-system/fastboot-flashall.png)

# 修改和编译 Framework 、Service、Res

以下命令都在源码根目录下执行

## 修改代码 IDE

### java代码

java 代码推荐使用 AndroidStudio 打开、编辑

### c/cpp代码

c/cpp 代码推荐使用 SourceInsight 、Eclipse、VS Code 打开、编辑

## 编译 Framework

|  |  |
| --- | --- |
| 1 | mmm framework/base |

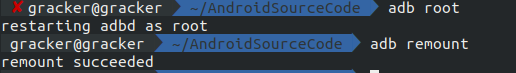
## 编译 Services

|  |  |
| --- | --- |
| 1 | mmm frameworks/base/services |

## 编译 res

|  |  |
| --- | --- |
| 1 | mmm frameworks/base/core/res |

## root && remount

[](https://www.androidperformance.com/images/android-system/root--remount.png)

## push

root && remount 之后，就可以把对应的 framework、Services、Res 等 push 进去，重启 shell 即可生效,或者直接 adb sync system 即可, sync system 会把 out目录下对应机型的 system 目录和手机的 system 目录进行同步，很是方便。

例子：

|  |  |
| --- | --- |
| 1 | adb root && adb remount && adb shell stop && adb sync system && adb shell start |