

03--FlutterEngine

下载引擎代码

工具准备

下载引擎

关于升级

编译引擎代码

配置项目代码

iOS工程

Flutter工程

检查二进制是否含有调试信息

lipo命令

LLDB检查是否含有调试信息

使用python列出模块的所有编译单元的完整路径

下载引擎代码

工具准备

- Chromium提供的部署工具[depot_tools](#)

Bash | [复制代码](#)

```
1  git clone https://chromium.googlesource.com/chromium/tools/depot_tools.git
```

- gitHub配置SSH

错误提示

Bash [复制代码](#)

```
1 git@github.com: Permission denied (publickey).
2 fatal: Could not read from remote repository.
3
4 Please make sure you have the correct access rights
5 and the repository exists.
```

这个是因为你GitHub的KEY过期了。长期不clone代码导致的。
进入~/.ssh 目录。删除 known_hosts 里面过期的git账号

Bash [复制代码](#)

```
1 $vi ~/.ssh/known_hosts
```

然后生成一对新的公钥、私钥。说白了这就是SSH免密登录。

Bash [复制代码](#)

```
1 $ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

然后注意！在Mac下。我们要做这个操作。将SSH 添加到ssh-agent!

1、在后台启动ssh-agent。

Bash [复制代码](#)

```
1 $ eval "$(ssh-agent -s)"
2 > Agent pid 18995
```

2、修改~/.ssh/config文件（这个是Mac系统大于 10.12.2 的）

Bash [复制代码](#)

```
1 $vi ~/.ssh/config
```

然后编辑内容,你私钥的名字和路径。

Bash | [复制代码](#)

```
1 Host *
2   AddKeysToAgent yes
3   UseKeychain yes
4   IdentityFile ~/.ssh/github_hank
```

3、将您的SSH私钥添加到ssh-agent并将密码短语存储在钥匙串中

Bash | [复制代码](#)

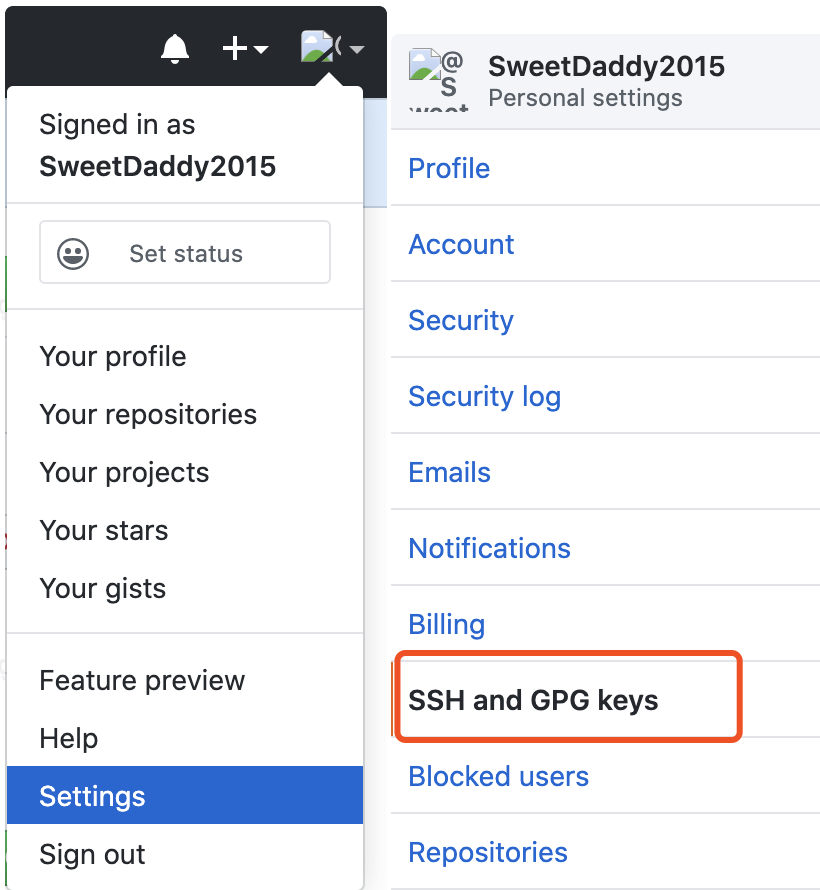
```
1 $ssh-add -K ~/.ssh/id_rsa
```

4、最后一步，Copy 我们的公钥内容，这里也可以用命令

Bash | [复制代码](#)

```
1 $pbcopy < ~/.ssh/github_hank.pub
```

点击头像设置，然后从左往右操作。



SSH keys / Add new

Title

Hank_Mac

Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQCAQEA
nzwOCeASOyQBgb7lel
emnFFQBFYY1rkvvk
x6clRqk3W5TnuGebf6xzjN
ZMvwnsPmUgspkYydzNL936-union
a.com
```

Add SSH key

- 配置工具的环境变量

(我放在/opt/目录下，这里面如果你需要放就需要给权限)

```

#Flutter镜像
export PUB_HOSTED_URL=https://pub.flutter-io.cn
export FLUTTER_STORAGE_BASE_URL=https://storage.flutter-io.cn
export DEPOT_TOOLS=/opt/depot_tools/

#Android Studio 配置
#export ANDROID_HOME=/Users/hank/Library/Android/sdk
#模拟器
#安卓 tools路径
#安卓平台工具

#写入环境变量
export PATH=$MonkeyDevPath/bin:$HKbin:$FLUTTER:$DEPOT_TOOLS:$PATH

```

- 安装最后一个工具

```
1 brew install ant
```

Bash | [复制代码](#)

下载引擎

- 新建目录（注意，我们的路径不能有中文，否则后续Down下载的源码会有问题）

```
1 mkdir engine
```

Bash | [复制代码](#)

- 创建gclient文件。（我们需要通过gclient下载源码）

```
1 touch .gclient
```

Bash | [复制代码](#)

- 然后配置文件。（尤其要确定CommitID 保持一致）

Bash | 复制代码

```
1 solutions = [  
2   {  
3     "managed": False,  
4     "name": "src/flutter",  
5     "url":  
6       "git@github.com:flutter/engine.git@6bc433c6b6b5b98dcf4cc11aff31cdee90849f32",  
7     "custom_deps": {},  
8     "deps_file": "DEPS",  
9     "safesync_url": "",  
10  ]
```

- 执行 `gclient sync`（这个操作将会 fetch Flutter 所有的依赖。这里有10G文件，需要点时间，请保持网络！该操作需要翻墙）

Bash | 复制代码

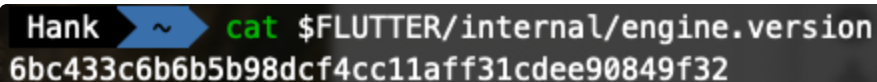
```
1 gclient sync
```

关于升级

当我们升级了Flutter的SDK，我们想要升级引擎代码。直接更新 `.gclient` 文件。

Bash | 复制代码

```
1 $cat $FLUTTER/internal/engine.version
```



```
Hank ~ cat $FLUTTER/internal/engine.version  
6bc433c6b6b5b98dcf4cc11aff31cdee90849f32
```

然后将这个修改到 `.gclient` 文件中

```
1 solutions = [  
2   {  
3     "managed": False,  
4     "name": "src/flutter",  
5     "url": "git@github.com:flutter/engine.git@6bc433c6b6b5b98dcf4cc11aff31cdee90849f32",  
6     "custom_deps": {},  
7     "deps_file": "DEPS",  
8     "safesync url": "",
```

然后进入 `src/flutter` 目录。

- 1、进行一次 `git pull` 然后 `git reset --hard commitID`

```
1 git pull
2 git reset --hard commitID
```

这个命令就是告诉Git 我下次下载就下载这个commitID的

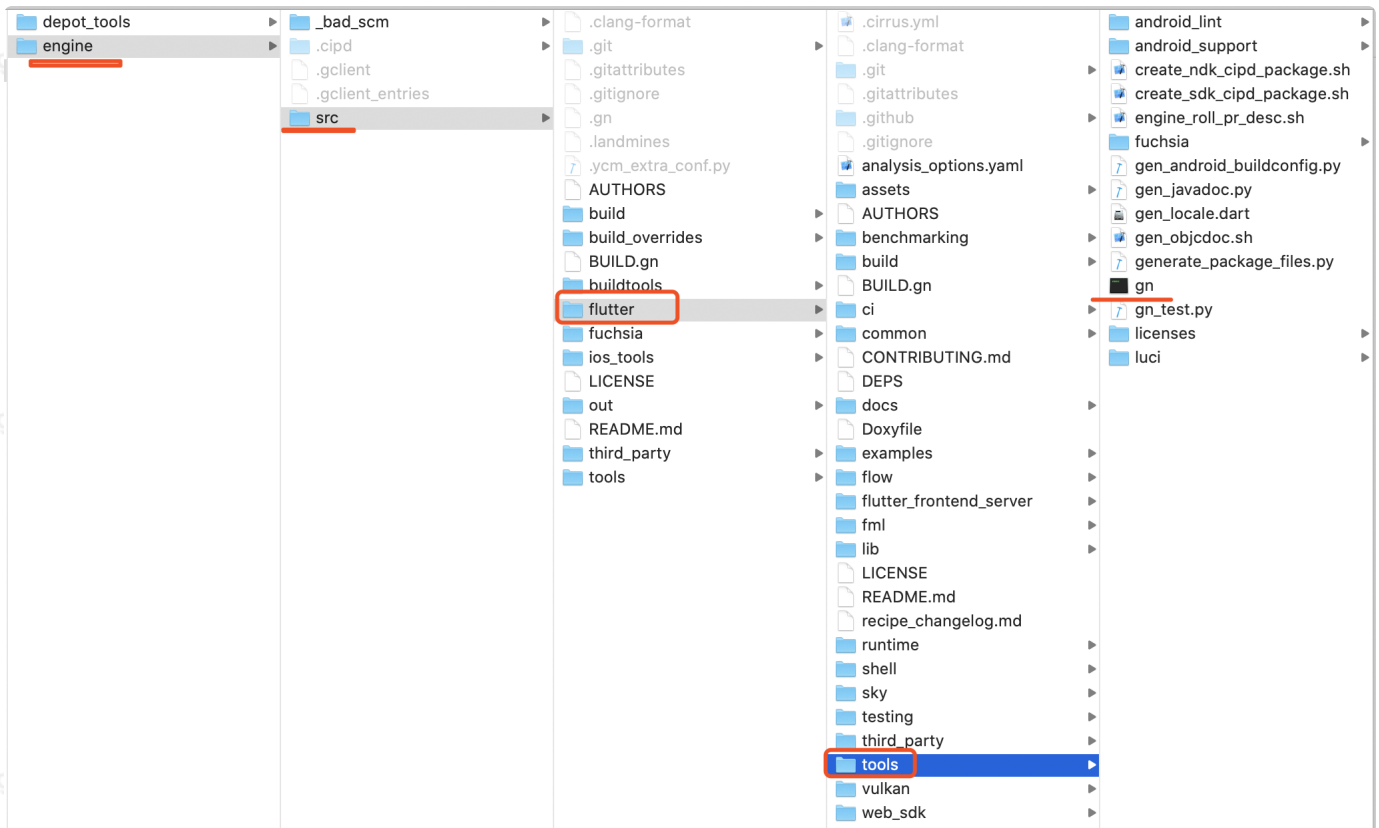
```
git reset --hard 6bc433c6b6b5b98dcf4cc11aff31cdee90849f32
```

回到engine 目录，也就是.gclient文件所在的目录

```
1 $gclient sync --with_branch_heads --with_tags --verbose
```

编译引擎代码

我们先要使用 GN：这个哥们是一个生成Ninja构建文件的元构建系统，最后我们还是用Ninja编译！

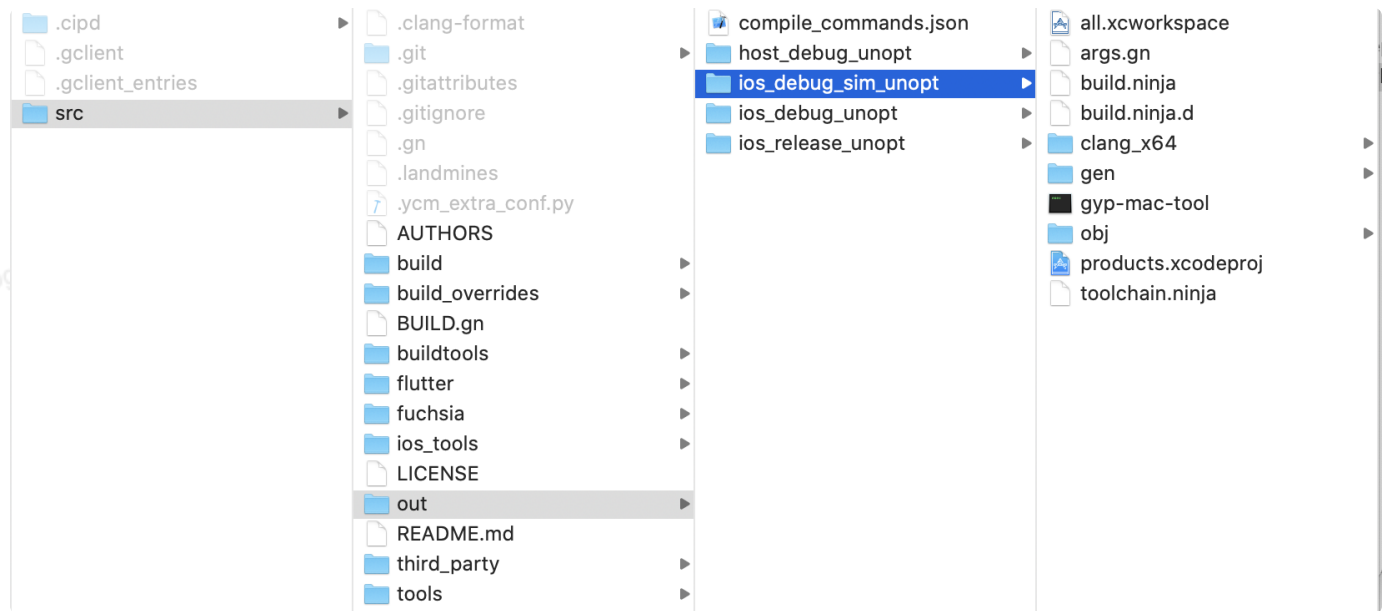


- 构建

Bash | 复制代码

```
1 #构建iOS设备使用的引擎
2 #真机debug版本
3 ./gn --ios --unoptimized
4 #真机release版本(日常开发使用,如果我们要自定义引擎)
5 ./gn --ios --unoptimized --runtime-mode=release
6 #模拟器版本
7 ./gn --ios --simulator --unoptimized
8 #主机端(Mac)构建
9 ./gn --unoptimized
```

构建完成会有四个Xcode工程



- 使用ninja编译工程(这里也是一个耗时操作)

Bash | 复制代码

```
1 ninja -C host_debug_unopt && ninja -C ios_debug_sim_unopt && ninja -C ios_debug_unopt && ninja -C ios_release_unopt
```

配置项目代码

项目配置以iOS为例。

iOS工程

在iOS工程中，找到Generated.xcconfig文件。在里面添加两个环境变量

Bash | [复制代码](#)

```
1 FLUTTER_ENGINE=你存放引擎代码的路径/engine/src
2 #使用的引擎对应版本（这里是iOS-debug模式下-模拟器的版本）
3 LOCAL_ENGINE=ios_debug_sim_unopt
```

Flutter工程

如果是使用AS配置，那么在

检查二进制是否含有调试信息

lipo命令

Bash | [复制代码](#)

```
1 #可以查看包含的架构
2 $lipo -info xxx
3 #拆分架构
4 $lipo xxx -thin armv7 -output armv7_xxx
5 #合并多架构
6 $lipo -create xxx.a xxx.a -output xxx.a
```

LLDB检查是否含有调试信息

Bash | [复制代码](#)

```
1 $lldb --file Flutter_arm64
2 (lldb) target create "Flutter_arm64"
3 Current executable set to 'Flutter_arm64' (arm64).
4 (lldb) script lldb.target.module['Flutter_arm64'].GetNumCompileUnits()
5 1
6 (lldb)
```

使用python列出模块的所有编译单元的完整路径

```
1 (lldb) target create "Flutter_arm64"
2 Current executable set to 'Flutter_arm64' (arm64).
3 (lldb) script
4 Python Interactive Interpreter. To exit, type 'quit()', 'exit()' or Ctrl-
  D.
5 >>> m = lldb.target.module['Flutter_arm64']
6 >>> for i in range(m.GetNumCompileUnits()):
7 ...     cu = m.GetCompileUnitAtIndex(i).file.fullpath
8 ...     print(cu)
9 ...
10 None
11 >>>
```