

CodeMagic journey

Suesi Tran



DISCLAIMER

- Using GitHub account
- No setup, no Flutter starter guide

Build trigger

- Default trigger setting
- NOTE: automatic build trigger will only run TEST

Build triggers

i ▼

Select which branches to track and when to trigger builds.

Include ▼

Add pattern

[Show pattern examples](#)

Automatic build triggering

☐ Trigger on every push

☒ Trigger on pull request update

Save

Cancel

Environment variables

- For private info that is not committed to Git
- FCI_CLONE_UNSHALLOW: useful for automated buildNumber using
 - **git rev-list --count**

Environment variables

Use environment variables to store information that you don't want to store in the repository. You can access these variables in your code by adding the `$` symbol in front of the variable name.

Variable name	=	Variable value	<input type="checkbox"/> Secure	Add
FIREBASE_CONFIG	=	*****		
FIREBASE_SERVICE	=	*****		
FCI_CLONE_UNSHALLOW	=	1		

SaveCancel

Dependency Caching

- Default setting

Dependency caching

i


Dependency caching allows to speed up builds by storing dependencies on Codemagic.
Specify the paths to be cached.

☐ Enable dependency caching

\$FCI_BUILD_DIR/dir/to/cache

Add

\$FCI_BUILD_DIR/build



Save

Cancel

Post clone script

- Use your Environment variable here
- Need to start with `#!/bin/sh`
- This script is run immediately after git clone success

Post-clone script

Run script after the clone phase has finished

```
#!/bin/sh

echo $FIREBASE_CONFIG > ./lib/firebase_config.dart
echo $FIREBASE_SERVICE > ./android/app/google-services.json
```

Pre-test script

Run script before the test phase has started

```
#!/bin/sh
set -e # exit on first failed command
set -x # print all executed commands to the log
```

Save

Cancel

Test

- Run flutter test
- Recommend to use flutter analyze here
- Tips: use --flutter-repo to analyze your flutter repo packages

Test

Assure the quality of your code by running tests and static code analysis.

Flutter analyze arguments

analyze

Flutter test target

test

☐ Enable Flutter analyzer

☒ Enable Flutter test

☒ Stop build if tests fail

Save

Cancel

Build (part 1)

- Flutter channel
- Platform
- Mode
 - Need Code Sign for Release Mode
- Xcode version

Specify how you want Codemagic to build your app.

Flutter version

channel Stable ▼

Build for platforms

☒ Android ☒ iOS ☐ Web ☐ Run tests only

☐ Build Android App Bundles

Mode

☐ Debug ☒ Release ☐ Profile


Xcode version

10.2 ▼

Build - Build arguments


- You can still change your build mode here.
- FCL_CLONE_UNSHALLOW is used in this step if you use the command 'git rev-list HEAD --count'
- Need Code Signing for release mode

Build arguments




--release ▼

--build-number=\$(git rev-list HEAD --count) --split-per-abi



--release ▼

--verbose



flutter packages pub global run webdev build

Save

Cancel

Code Signing - Setup Android (1)

- Keystore
- Use key.properties for release

Steps

- Create key.properties file
- Get signing details from your key.properties file
- Configure release buildType

To Be Continued...

```
storePassword=<password from previous step>
keyPassword=<password from previous step>
keyAlias=key
storeFile=<location of the key store file, such as /Users/<user name>/key.jks>
```

```
def keystoreProperties = new Properties()
def keystorePropertiesFile = rootProject.file('key.properties')
if (keystorePropertiesFile.exists()) {
    keystoreProperties.load(new FileInputStream(keystorePropertiesFile))
}
```

```
signingConfigs {
    release {
        keyAlias keystoreProperties['keyAlias']
        keyPassword keystoreProperties['keyPassword']
        storeFile file(keystoreProperties['storeFile'])
        storePassword keystoreProperties['storePassword']
    }
}
buildTypes {
    release {
        signingConfig signingConfigs.release
    }
}
```

Code Signing - Setup Android (2)

Easy:

- upload your keystore
- Enter keystore password
- Enter your alias
- Give alias password

Set up Android code signing to enable installing your app on real devices and publishing it to Google Play.

Keystore

my_keystore



Keystore password

[Delete/Modify password](#)

Key alias

mykey

Key password

[Delete/Modify password](#)

Save

Cancel

Code Signing - Setup iOS (1)

Easy way:

- Use CodeMagic automatic signing
- **Success rate:** 0 for me

My Way

- Upload distribution key in p12 format
- Upload provisioning profile

To Be Continued...

Code Signing - Setup iOS (2)

Easy - but tricky:

- Choose the correct distribution key and provisioning profile
- Generate P12 key for your distribution key
- Give your password

Select code signing method

☐ Automatic ☒ Manual

Manually upload your signing certificate and provisioning profile for code signing.

Code signing certificate

ThuyTranDistribution23May.p12 ×


Certificate password

[Delete/Modify password](#)

Provisioning profiles

If your app contains app extensions, you will need to upload an additional provisioning profile for each extension.

MyWalletDistribution.mobileprovision ×

 Choose a file or drag it here.

Save

Cancel

Publish

- Android (1)
- Setup Service Account in Google Play to allow third party upload
- After all steps are done, download the json file and keep it to use later.

Create service account

Service account name [?]
Nevercode

Service account ID
nevercode @api-7616106293651970307-207344.ia

☒ **Furnish a new private key**
Downloads a file that contains the private key. Store the file securely; it can't be recovered if lost.

Key type
☒ **JSON** (Recommended)
☐ **P12** (For backward compatibility with code using the P12 format)

☐ **Enable G Suite Domain-wide Delegation**
Allows this service account to be authorized to access all domain resources without manual authorization on their part. [Learn more](#)

Role [?]
Select a role

Selected

- Datastore
- Error Reporting
- IAM
- Logging
- Monitoring
- Organization Policy
- Reserve Partner
- Resource Manager
- Roles
- Service Accounts**
- Service Management
- Service Usage
- Stackdriver Debugger
- Storage

Service Account User

Service Account Admin
Service Account Key Admin
Service Account Token Creator

7. Find the new service account and click **Grant Access**.

8. Select **Release manager** from the Role dropdown and click **Add user**.

Publish

- Android (2)
- Upload Service Account json file
- Select channel to publish to

Google Play

Set up publishing to one of the Google Play tracks.

Credentials (.json)

api-8319088387818250133-607651-d4e0163a063f.json ✕

Track

☐ Internal ☐ Alpha ☒ Beta ☐ Production

☐ Publish even if tests fail

Save Cancel

Publish - iOS

Easy - but tricky:

- Enter your AppleID and password
- App-specific password is tricky part

App Store Connect

Set up publishing to App Store Connect.

Apple ID

Tranhathuy@gmail.com

App-specific password

[Delete/Modify password](#)

App ID

A unique ID assigned to the application in App Store Connect.

com.nartus.wallet

☐ Publish even if tests fail

<https://appleid.apple.com/account/manage>

Good

~~But not great~~



Q&A

Any Questions?

Suesi Tran

An Fluttering Ando-chan

