What is FastLane?

Fastlane can be simply described as the easiest way to automate building and release your iOS and Android apps, via a suite of tools that can work either autonomously or in tandem to accomplish tasks . Fastlane is a tool for iOS and Android developers to automate tedious tasks like generating screenshots, dealing with provisioning profiles, and releasing your application.

The benefit of leveraging one or more fastlane actions is in your ability to save hours and even days, saving you the laborious task of having to submit, provision, and take screenshots manually, and instead allowing you to focus on what matters: that is, feature development. This is the mantra of Continuous Deployment and CD—the ability to code and release iteratively and rapidly, with minimal barriers. This is what fastlane is.

1.Setting Up fastlane

The tool fastlane is a collection of Ruby scripts, so you must have the correct version of Ruby installed. Chances are that your OS comes with Ruby 2.0 by default, but you can confirm whether this is the case by opening Terminal and entering:

ruby -v

If installed then good if not then install using Homebrew: /usr/bin/ruby -e \

"$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)
Then install ruby using
brew update && brew install ruby
You'll also need Xcode Command Line Tools (CLT). To ensure that they're installed, enter into Terminal: xcode-select --install

Now after all this setup install fastlane using command :

brew cask install fastlane

Once installation completes, you're ready to set up fastlane in your project. But before you do, here's a high-level look at the fastlane tools.

The fastlane Toolchain

To work its magic, fastlane brings the following set of tools all under one

roof:
● produce creates new iOS apps in both App Store Connect and the Apple Developer Portal.
● cert automatically creates and maintains iOS code-signing certificates.
● sigh creates, renews, downloads and repairs provisioning profiles.
● snapshot automates taking localized screenshots of your iOS app on every device.
● frameit puts your screenshots into the right device frames.
● gym builds and packages your iOS apps.
● deliver uploads screenshots, metadata and your apps to the App Store.
● pem automatically generates and renews your push notification profiles.
● spaceship is a Ruby library able to access the Apple Developer Center and App Store Connect APIs. ● pilot automates TestFlight deployments and manages beta testers.
● boarding invites beta testers.
● match syncs certificates and provisioning profiles across your team, using Git.
● scan runs tests on your apps.

2. Adding fastlane to your project

Open terminal and navigate to your project directory using cd command then enter

---

fastlane init
If you get a “permission denied” error, prefix this command with sudo.

Back in the project folder, you’ll see a few new things: Gemfile, which includes the fastlane gem as a project dependency and a fastlane folder containing:

● Appfile: stores the app identifier, your Apple ID and any other identifying information that fastlane needs to set up your app.
● Fastfile: manages the lanes you’ll create to invoke fastlane actions.

Open Fastfile in a text editor of your choice, disable smart quotes if your text editor supports them, then replace the contents of the file with:

default_platform(:ios)

```
platform :ios do #1

desc "Create app on Apple Developer and App Store Connect sites" #2

lane :create_app do #3

produce end

end
```

What this code does is :

1. Provides a description for the lane. (A lane is a workflow of sequential tasks).
2. Names this lane create_app.
3. Uses produce to add the app to both the Developer Portal and App Store Connect.

Save Fastfile, then open Appfile. Remove the pound (#) sign to uncomment the line starting with apple_id, then replace [[APPLE_ID]] with your actual Apple ID username. By setting this information now, fastlane won't have to repeatedly prompt you for it later on.

3.Creating your App

Open Terminal inside your project folder and enter: fastlane create_app

Enter your App Store Connect password if prompted.Eventually, produce will ask you to enter your bundle ID. Time to create one!
The bundle identifier must be different from every other bundle identifier anyone's ever used in App Store Connect. Try to enter a unique bundle identifier by using the following format:

App name [Insert your email address minus "@" and "."]

If the bundle identifier is already taken, edit it and try again until you've submitted a unique ID.

Then, when you're prompted to submit an app name, it too will have to be unique.

Reopen Appfile, uncomment the line starting with app_identifier, then replace [[APP_IDENTIFIER]] with the bundle ID you just created.

4. Generating the DeliverFile

In terminal enter:

bundle exec fastlane deliver
When fastlane asks; “Do you want to setup deliver?” Enter y in response.

Next fastlane will ask, “Would you like to use Swift instead of Ruby?” Although, as an iOS developer, you’re probably more comfortable working in Swift, as of the writing of this tutorial, fastlane.swift is still in beta. So enter n to use Ruby in your fastlane files.

Once deliver successfully completes, navigate back to mZone/fastlane in your Finder and you’ll see some new things:
● The metadata directory, which will hold the majority of the app’s metadata.
● Deliverfile, which will hold a few remaining pieces of metadata.

● The screenshots directory, which will contain the app screenshots.

In the metadata directory, you’ll notice a bunch of text files named after common App Store items like the description, keywords, categories, etc. fastlane will use these files to submit your app’s metadata information to App Store Connect.

Open en-US/description.txt and add:
Some description about the app like what it does and what are the benefits

Then, in the same folder, use your favorite text/code editor to create a JSON file named app_store_rating_config.json containing:

```
{
"CARTOON_FANTASY_VIOLENCE": 0,
"REALISTIC_VIOLENCE": 0,
"PROLONGED_GRAPHIC_SADISTIC_REALISTIC_VIOLENCE": 0,
"PROFANITY_CRUDE_HUMOR": 0,
"MATURE_SUGGESTIVE": 0,
"HORROR": 0,
"MEDICAL_TREATMENT_INFO": 0, "ALCOHOL_TOBACCO_DRUGS":
0,
"GAMBLING": 2,

"SEXUAL_CONTENT_NUDITY": 0,
"GRAPHIC_SEXUAL_CONTENT_NUDITY": 0,
"UNRESTRICTED_WEB_ACCESS": 0, "GAMBLING_CONTESTS": 0

}
```

This rating configuration indicates that the app has “frequent/intense” simulated gambling (i.e., value = 2) and none of the other listed content. This file gives Apple the information it needs to assign an appropriate age rating.
And, lastly, in the review_information folder, add your email address to email_address.txt, your first name to first_name.txt, your last name to last_name.txt and your phone number to phone_number.txt. Preface the phone number with ‘+’ followed by the country code, for example; +44 844 209 0611.

5. Automating Screenshots

Set up the project for snapshot by entering in Terminal: fastlane snapshot init

A Snapfile file will now appear in your fastlane folder. Open it and replace the contents of the file with: # 1 - A list of devices you want to take the screenshots from
devices([

"iPhone 8 Plus",

"iPhone SE"])

2 - A list of supported languages

languages(['en-US', 'fr-FR'

])
# 3 - The name of the scheme which contains the UI Tests
scheme("mZone Poker UITests")
# 4 - Where should the resulting screenshots be stored?
output_directory "./fastlane/screenshots"
5 - Clears previous screenshots clear_previous_screenshots(true)
Here, you specify:

1. The devices from which you want fastlane to capture your screenshots.
2. The localized languages you want to capture.
3. The name of the Xcode scheme you’ll soon create to run screenshot automation.
4. The screenshot output directory.
5. That fastlane should clear any screenshots in the output directory before capturing new ones. 6. Save the file before closing.

Return to Terminal and note the instructions that appeared after running

fastlane snapshot init:

6.Creating the IPA file

Building and uploading the app can also be a time-consuming process. But guess what — fastlane can do this with its gym tool! Run this command in the terminal to create a gymfile.

fastlane gym init
Open gymfile and replace contents with:

```
#1
scheme("mZone Poker")
#2
output_directory("./fastlane/builds")
#3
include_bitcode(false)
#4
include_symbols(false)
#5 export_xcargs("-allowProvisioningUpdates")
```

This code does the following:

1. Specifies mZone Poker's scheme.
2. Specifies where fastlane should store the .ipa app binary file.
3. Excludes bitcode from the build. Bitcode allows Apple to optimize your app, but exclude it for now to speed up the build.
4. Excludes symbols from the build. Including symbols allows Apple to access the app's debug information, but exclude it for now to speed up the build.
5. Allows Xcode to use automatic provisioning.

Open Fastfile and add the following below the screenshot lane:

```
desc "Create ipa" lane :build do

#1 enable_automatic_code_signing #2
increment_build_number
#3
gym

End
```

This build lane:

1. Enables automatic provisioning in Xcode.

2. Increases the build number by 1 (so each build number is unique per App Store Connect's upload requirement).
3. Creates a signed .ipa file.

Save Fastfile, then run build in Terminal: bundle exec fastlane build

If you're prompted to enter your keychain password in order for fastlane to access your certificates, do so. Select Allow Always if you don't want to repeatedly grant the same permission. Once build successfully completes, the signed .ipa should appear in fastlane/builds.

7. Uploading to App Store Connect

To upload the screenshots, metadata and .ipa file to App Store Connect, you'll use deliver. First, replace Deliverfile's contents with:

```
#1
price_tier(0)
#2 submission_information({

export_compliance_encryption_updated: false,
export_compliance_uses_encryption: false,
content_rights_contains_third_party_content: false,
add_id_info_uses_idfa: false

})
#3 app_rating_config_path("./fastlane/metadata/
app_store_rating_config.json") #4
ipa("./fastlane/builds/mZone Poker.ipa")
#5
submit_for_review(true)
#6
automatic_release(false)
```

Here you:

● Set price tier to 0, indicating it's a free app.
● Answer the questions Apple would present to you upon manually submitting for review. ● Provide the app rating configuration location.

● Provide the .ipa file location.
● Set submit_for_review to true to automatically submit the app for review.
● Set automatic_release to false so you'll have to release the app manually after it's accepted by app review.

Open Fastfile. After the build lane, add:

desc "Upload to App Store" lane :upload do

deliver End

Then, for fastlane to create a preview of the metadata, open Terminal and enter:

bundle exec fastlane upload

If everything looks good, type y into Terminal to approve.

Once the lane completes, log in to App Store Connect. The screenshots, metadata and build should be there, waiting for review.

You now have separate lanes for creating your app, taking screenshots, building and uploading. You could call them one by one and upload your build on App Store!!