

CI/CD with via Firebase using Fastlane

1. Prepare environment(skip to 2 if done)

1. Run \$ **xcode-select --install**
2. Check ruby version using **'ruby -v'**
3. Upgrade to new if needed (if < 2.6.5)
 - use one of package managers (brew, rvm, rbenv, etc...)
 - upgrade \$ **rvm get stable** or \$ **brew upgrade rbenv ruby-build** or \$ **brew upgrade ruby**
 - check version \$ **'ruby -v'**
 - if version was not upgraded properly, specify path to newest version in /Users/{your_user_name}/.bash_profile , copy and paste two lines below

It depends of package manager you use(recommend to use rvm or rbenv):

rbenv(copy to bash_profile):

```
'export PATH="$HOME/.rbenv/bin:$PATH"  
eval "$(rbenv init -)"
```

rvm(command line tool):

```
$ echo '[[ -s "$HOME/.rvm/scripts/rvm" ]] && .  
"$HOME/.rvm/scripts/rvm" # Load RVM function' >>  
~/.bash_profile
```

brew(copy to bash_profile):

```
'export PATH=/usr/local/Cellar/ruby/{ruby_version}/bin:$PATH'
```

4. Install or update RubyGems
 - \$ **which gem**
 - \$ **gem update --system**
5. Install Bundler Gem
 - \$ **gem install bundler**
6. Install or update fastlane.
 - \$ **sudo gem install fastlane -n /usr/local/bin.**
 - or
 - \$ **sudo gem update fastlane.**

P.S You can use brew for fastlane installation. Make Sure you use latest fastlane version 2.134.0 (18.10.19) \$ **'fastlane -version'**

2. Configure Fastlane

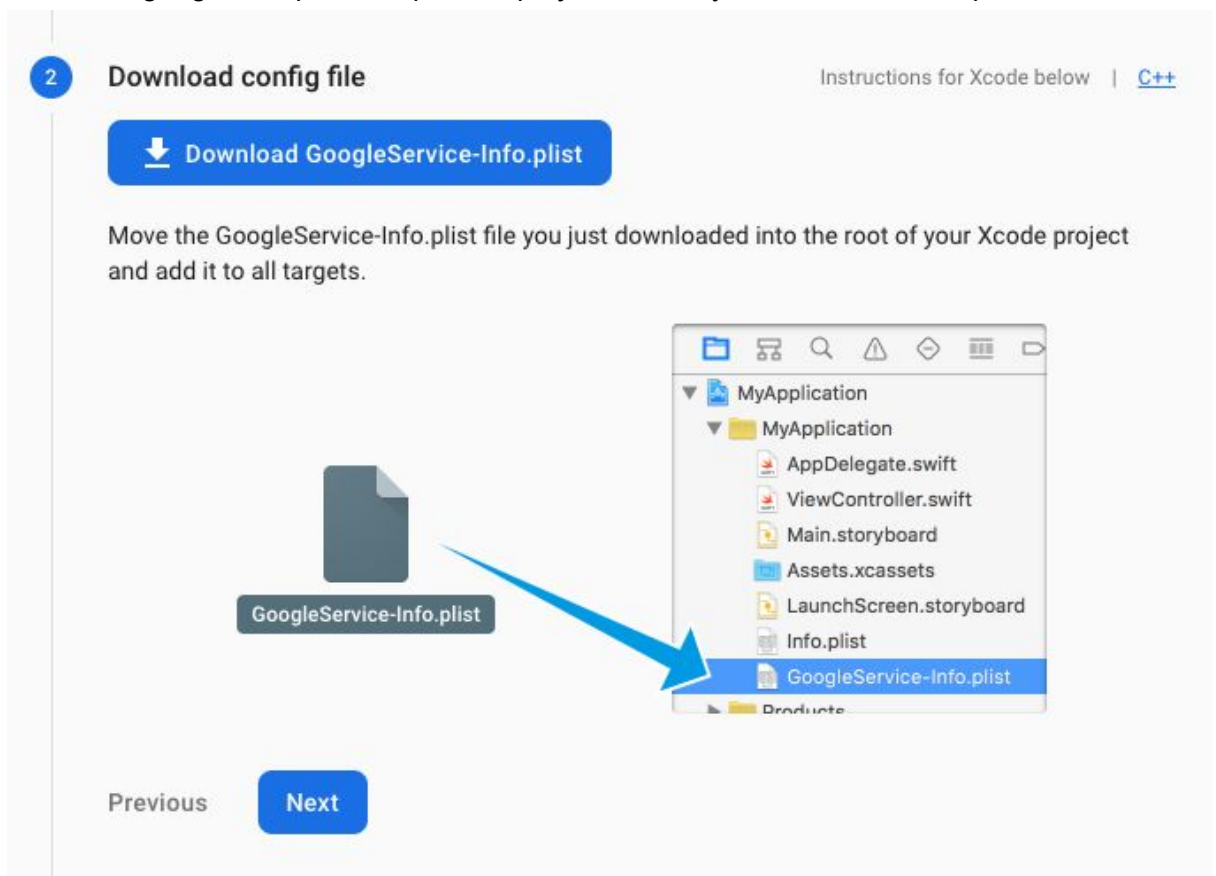
1. Add Firebase plugin
- \$ **fastlane add_plugin firebase_app_distribution**

3. Credentials Apple

1. Drag into credentials folder(./fastlane/credentials/) :
 - DevProvisionProfile
 - DistributionProvisionProfile
 - Certificate.p12(with a private key) (Distribution cert)

3. Configure FireBase

1. Create your project
2. Create application, configure and register it.
3. Download google-info.plist and put into project directory with standard info.plist



4. You can migrate project from Fabric.
 - FirebaseConsole->YourProject->QualityTab(LeftSideMenu)->Crashlytics



The most powerful, yet lightest weight crash reporting solution

[Learn more](#)

1 Are you a Fabric user migrating a Crashlytics app? ?

☐ No, set up a new Firebase app

Choose this if you're setting up Crashlytics for a new app

☐ Yes, migrate my Fabric app to Firebase

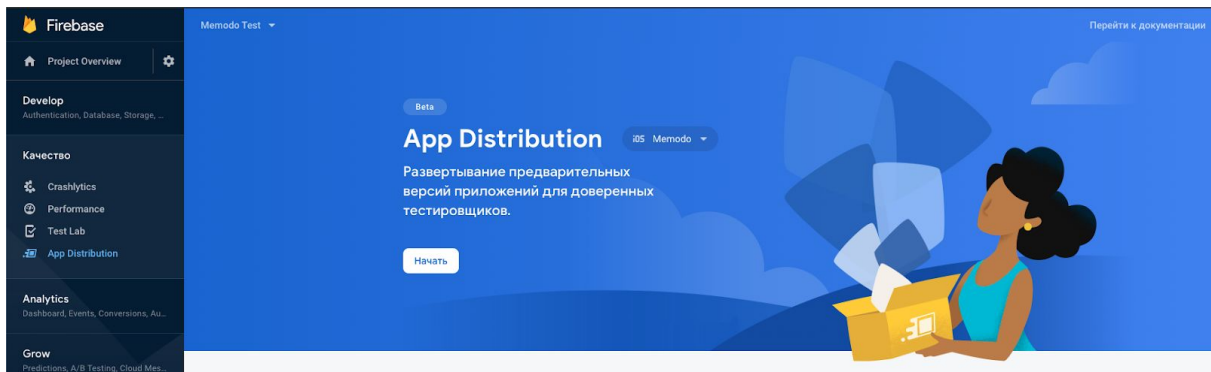
Choose this if you are an existing Fabric user and want to migrate a Fabric Crashlytics app. Crash data will appear in both Fabric and Firebase dashboards.

Next

2 Install the SDK

3 Build and run your app

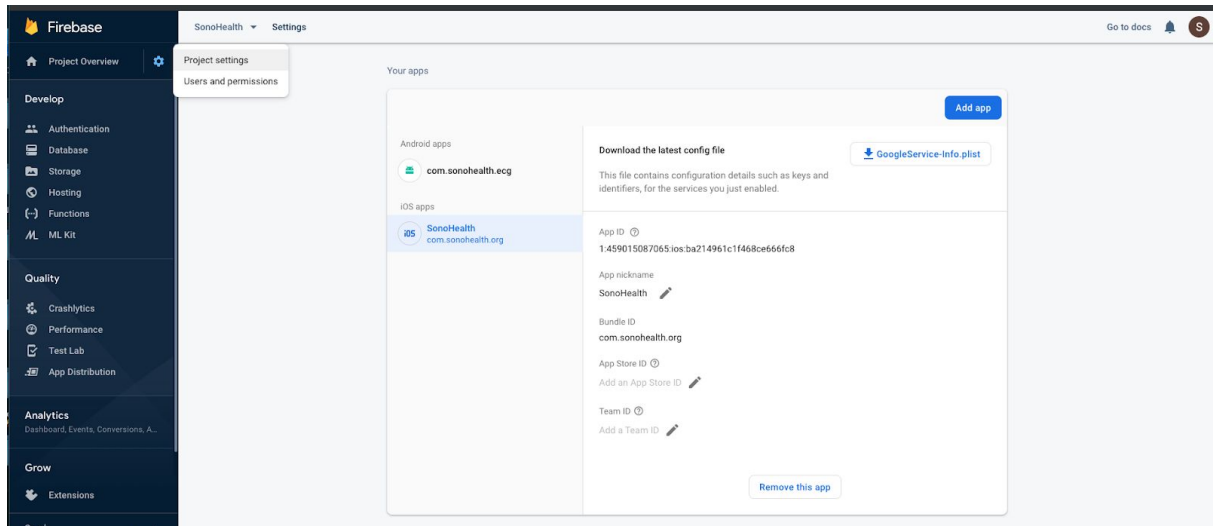
5. FirebaseConsole->YourProject->QualityTab(LeftSideMenu)->AppDistribution
Press "Get Started button"



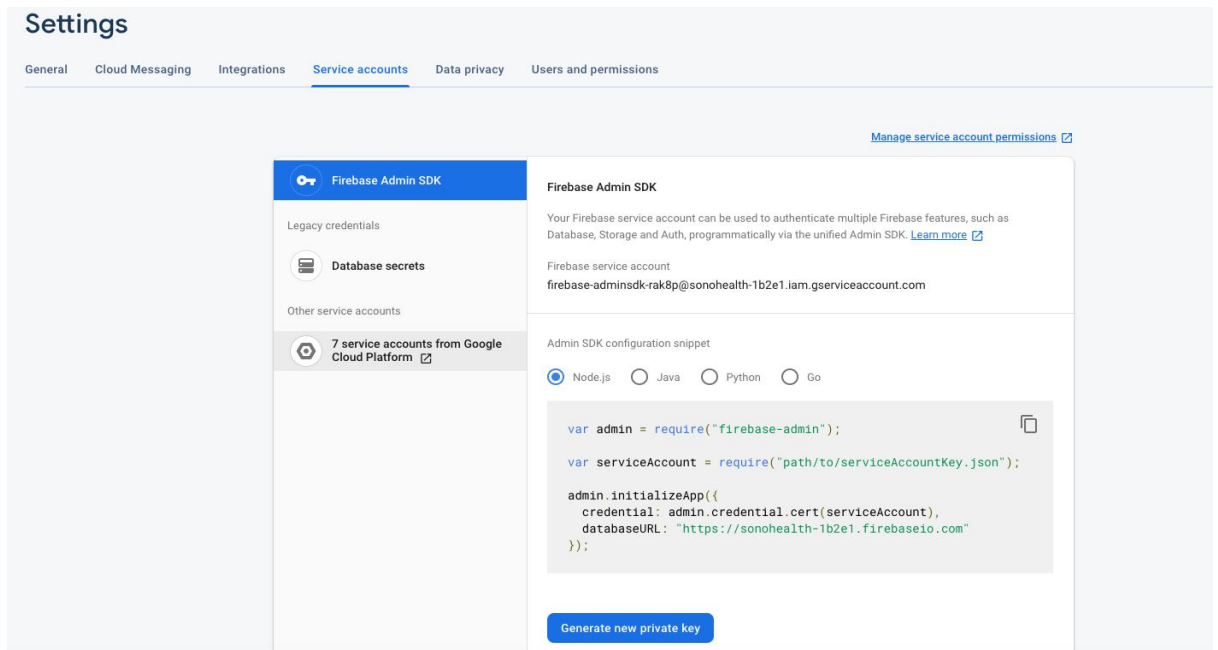
P.S Before migration from Fabric make sure you have admins permissions on fabric.

4. Credentials For Firebase

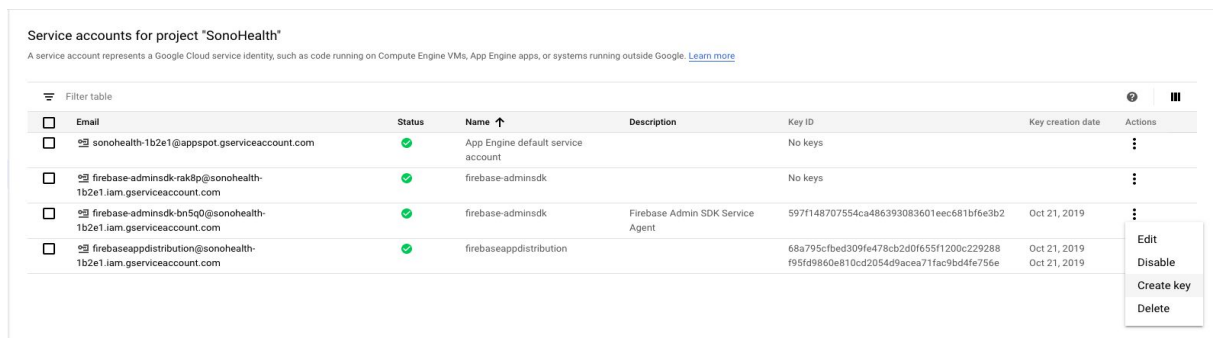
1. Get App ID (copy to clipboard)



2. Create firebase Service key



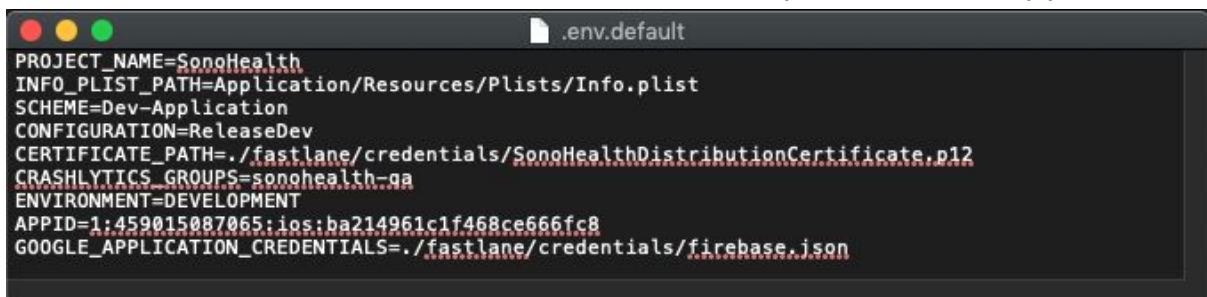
2. Download key and rename to firebase.json



3. Put into ./fastlane/credentials/firebase.json
4. Provide path to key in env.dev file (look up at the bottom)


5. Configure Fastfile

1. Setup file (recommend to copy from another project)
 - **\$ fastlane init**
 - look up example at the bottom
2. Specify **default.env**
 - PROJECT_NAME= { Your project name }
 - INFO_PLIST_PATH= { Path to plist file }
 - SCHEME={ xCode scheme } (dev || stage || prod)
 - CONFIGURATION= { xCode configuration } (debug || release)
 - CERTIFICATE_PATH= { Path to Distribution certificate }
 - CRASHLYTICS_GROUPS= { Firebase groups name } //Comma sep. String
 - ENVIRONMENT= { Environment } (dev || stage || prod)
 - APPID= { Firebase app id }
 - GOOGLE_APPLICATION_CREDENTIALS= { Path to private key }



```
.env.default
PROJECT_NAME=SonoHealth
INFO_PLIST_PATH=Application/Resources/Plists/Info.plist
SCHEME=Dev-Application
CONFIGURATION=ReleaseDev
CERTIFICATE_PATH=./fastlane/credentials/SonoHealthDistributionCertificate.p12
CRASHLYTICS_GROUPS=sonohealth-ga
ENVIRONMENT=DEVELOPMENT
APPID=1:459015087065:ios:ba214961c1f468ce666fc8
GOOGLE_APPLICATION_CREDENTIALS=./fastlane/credentials/firebase.json
```

3. Specify **stage.env**, **prod.env**



```
.env.stage
SCHEME=Stage-Application
CONFIGURATION=ReleaseStage
ENVIRONMENT=STAGE

.env.prod
SCHEME=Prod-Application
CONFIGURATION=ReleaseProd
ENVIRONMENT=PRODUCTION
```

4. Use fastlane to Deploy project.
 - open directory with project **\$ cd {path_to_project}**
 - **\$ fastlane deploy --env {dev or stage or prod} build:{build_number}**

! P.S Make sure you have latest version of RubyGems, Fastlane, Ruby and fastlane-plugin for firebase_app_distributon configured and updated properly.

6. Fastfile example

```
# Uncomment the line if you want fastlane to automatically update itself
# update_fastlane

default_platform(:ios)

platform :ios do
  before_all do |lane, options|
    begin
      delete_temp_keychain
    rescue
    end
    create_temp_keychain
    import_app_certificates
    import_app_provision_profiles
  end

  after_all do |lane, options|
    delete_temp_keychain
  end

  error do |lane, exception, options|
    delete_temp_keychain
  end

  desc "Creates keychain"
  private_lane :create_temp_keychain do
    create_keychain(
      name: keychain_name(),
      default_keychain: false,
      unlock: true,
      timeout: 3600,
      lock_when_sleeps: true,
      password: ""
    )
  end

  desc "Imports certificates to created keychain"
  private_lane :import_app_certificates do
    import_certificate(
      certificate_path: ENV["CERTIFICATE_PATH"],
      keychain_name: keychain_name(),
      keychain_password: ""
    )
  end

  desc "Imports provisioning profiles"
  private_lane :import_app_provision_profiles do
    sh "cp ./credentials/*.mobileprovision ~/Library/MobileDevice/Provisioning Profiles/"
  end

  desc "Delete created keychain"
  private_lane :delete_temp_keychain do
    puts keychain_name()
    delete_keychain(name: keychain_name())
  end
end
```

```

desc "Upload build to firebase"
private_lane :upload_to_firebase do
  firebase_app_distribution(
    app: ENV["APPID"],
    release_notes: ENV["ENVIRONMENT"],
    groups: ENV["CRASHLYTICS_GROUPS"]
  )
  upload_symbols_to_crashlytics
end

lane :deploy do |options|
  set_info_plist_value(path: ENV["INFO_PLIST_PATH"], key: "CFBundleVersion", value: next_version(options).to_s)
  carthage(platform: "iOS", configuration: ENV["CONFIGURATION"], cache_builds: true) if File.file?("../Cartfile")
  gym(
    scheme: ENV["SCHEME"],
    configuration: ENV["CONFIGURATION"],
    export_method: "ad-hoc",
    export_options: {
      compileBitcode: false,
    }
  )
  upload_to_firebase
end

def keychain_name()
  "Keychain_#{ENV["PROJECT_NAME"]}"
end

def next_version(options)
  def to_i?(str)
    Integer(str)
  rescue
    false
  end
  # Version from file
  def version?(filename)
    if File.exists?(filename)
      to_i?(File.open(filename, 'r') { |file| file.read })
    else
      false
    end
  end
  filename = "../build-version"
  new_version = to_i?(options[:build]) || (version?(filename) || 0).next
  # Write
  File.open(filename, 'w') { |file| file.write(new_version) }
  return new_version
end
end

```