Gitlab CI + Fastlane

Автоматизация сборок в GitLab CI

Определения

Основные термины

- GitLab CI - автоматическая сборка для Gitlab
- Fastlane - управление сборками через терминал
- Раннер - компьютер на MacOS, например macMini
- HomeBrew - менеджер зависимостей для Mac

Gitlab runner

Начальная настройка раннера на macOS

- Установить homebrew
- Для установки выполняем в терминале brew install gitlab-runner
- Регистрируем раннер в терминале gitlab-runner register
- При регистрации будет запрошен:
  - URL gitlab репозитория(в гитлаб: Settings -> Runners)
  - Токен для настройки(в гитлаб: Settings -> Runners)
  - Название для раннера
  - Теги раннера
- После регистрации потребуется ввести среду выполнения команд, в нашем случае shell
- После этого выполняем gitlab-runner install и gitlab-runner start
- Перезагружаем раннер
- Проверяем, что раннер появился в Settings -> Runners и позеленел
- Основная конфигурация раннера хранится на нем в ~/.gitlab-runner/config.toml

Fastlane

Начальная настройка Fastlane на раннере

- Устанавливаем, выполнив в терминале brew install fastlane
- Переходим в папку проекта через терминал cd путь_до_проекта
- Выполняем fastlane init или fastlane init swift, для генерации файлов конфигурации на Swift
- Сгенерируется папка fastlane с файлами конфигурации
- Fastfile состоит из lanes, по сути список команд с названием
- Также есть встроенные команды для упрощения работы со сборками(scan, gym и тд)
- В lanes можно передавать параметры, а также получать конфиденциальные параметры из GitLab CI/CD variables

Пример lane

Также примеры lanes можно найти в официальном репозитории fastlane

```
lane :test do

  ENV["FASTLANE_XCODE_LIST_TIMEOUT"] = "120"
  ENV["FASTLANE_XCODEBUILD_SETTINGS_TIMEOUT"] = "120"

  run_tests(
    project: ENV["XCODE_PROJECT"],
    scheme: ENV["XCODE_SCHEME_TESTS"],
    clean: true,
    suppress_xcode_output: false)

end
```

Пайплайны

.gitlab-ci.yml

- Pipeline состоит из stages
- Stage состоят из Jobs
- Каждая Stage состоит из нескольких Jobs, которые запускаются последовательно или параллельно
- Stage выполняются только последовательно
- allow_failure: true позволяет стартовать следующему Stage, если Job внутри предыдущей Stage зафейлился
- When:on_failure внутри Job выполняется только если пайплайн фейлится
- Есть встроенный редактор CI/CD -> Editor

Пример Stage из пайплайна

Больше примеров на сайте GitLab

```yaml
fastlane-test:
  stage: test
  when: always
  only:
    - master
    - develop
    - merge_request
  script:
    - bundle exec fastlane test
```

Преимущества и недостатки
Gitlab CI и Fastlane

- Преимущества
  - Автоматизация рутинных задач(линтовщик, тесты, статические анализаторы, выкладка в S3, Firebase, Testflight, бамп версии, отправка сообщений в Slack, комментарии к задаче Jira, etc)
  - Можно создавать свои скрипты(например для проверки количества строк в MR)
- Недостатки
  - Сложность начальной настройки
  - Необходимость периодического обновления раннеров
  - Сложность параллелизации