ブログ開設 (無料) ログイン ヘルプ▼ 茶 Hatena えんじにあ雑記! Unity C# 技術書 UniRxで Screen-Shooterで C#でListに対する操作を簡潔に書く! 「ラムダ式はデリゲート型ではないので デザインパターンの解説と実装例 「バッチ処理でドローコールを減らす」の巻! IObserver<T>型に変換できません」 ~便利なメソッドと使い方集~ 華麗にリリース申請を決める! 🎉 と言われて困った話と解決方法 iOS/Android申請用のスクショをさ デザインパターンの解説と実装例! C#のListに対する操作を簡潔に! uGUIのバッチ処理について UniRx TReactive Property O Subsc くっと作る ribeにラムダ式が使えない **□ CI/CD Flutter ©** 2020-09-13 プロフィール Github ActionsでiOSアプリのCI/CDをやってみた **f** Facebook Twitter **∨** Pocket **B!** Hatebu マミーめも~ (id:flat-M_M) PRO **★** 読者になる 3 検索 FlutterでのiOSアプリ開発 Q 記事を検索 GithubActions x Deploygateで 最新記事 CI/CD環境を整える! ContentSizeFilterを実装してみる 改めてuGUIのリファレンスを読んでみたまとめ 2020年の振り返りと100歳までの抱負 自作Packageを作成、UPMで管理してみた GoogleSpreadSheetを使ってマスタデータを管理す > Flutterで<u>iOS/Android</u>アプリを開発していた際に、GithubActionsを使って<u>iOS</u>のCI/CD環境を構築した際のまと めです。 INDEX 月別アーカイブ この記事 is 何? •完成品 **2021 (4)** 使用したもの **2020 (29)** •yamlファイルの説明 2020 / 12 (1) ビルドが走るタイミングの設定 2020 / 9 (3) XCodeのバージョン設定 2020 / 8 (2) ■Flutterのインストールをキャッシュしておく 2020 / 7 (3) 2020 / 6 (1) ■Flutterのインストール 2020 / 5 (3) ■Flutterのライブラリ関連のインストール 2020 / 4 (7) ■ Provisioning Profileのデコード 2020 / 3 (3) ・Certificateの設定 2020 / 2 (4) ■ビルドとアーカイブ 2020 / 1 (2) ・使用しているSecretsの詳細 詰まったこと ■base64のデコード カテゴリー C# (11) Unity (18) この記事 is 何? CI/CD (5) Flutter (2) Yahoo!さん主催の<u>ハッカソン</u>「HackU」に向けてチーム開発している際に、デザイナーのメンバーにもアプリを 手元の実機で確認してもらえる環境が作りたい! イベント (1) できればGithubActionsとか使ってmasterにプッシュされたら自動でビルドとデプロイを行うみたいなカッコよ Golang (1) さげなことしてみたい! サーバサイド (1) と思ってCI/CD環境を整えた際に得た知見をまとめた記事です。 論文 (1) Android用のCI/CD環境を整えたお話は<u>こちら</u>にまとめました。 技術本の読後まとめ (4) 兎にも角にも完成品の<u>yaml</u>ファイルをひとまず次に載せておきます。 完成品 name: iOS CI/CD on: push: branches: master pull_request: branches: [master jobs: build: runs-on: macos-latest steps: - uses: actions/checkout@v2 - name: Select Xcode version 11.7 run: sudo xcode-select -s '/Applications/Xcode_11.7.app/Contents/Developer - name: Show Xcode version run: xcodebuild -version - name: setup cache uses: actions/cache@v1 with: path: /Users/runner/hostedtoolcache/flutter key: \${{runner.0S}}-flutter-install-cache - name: install flutter uses: subosito/flutter-action@v1 with: flutter-version: '1.20.2' channel: 'stable' - name: flutter dependencies install run: flutter pub get - name: Import Provisioning Profile run: mkdir -p ~/Library/MobileDevice/Provisioning\ Profiles touch ~/Library/MobileDevice/Provisioning\ Profiles/decoded.mobileprovisi echo -n '\${{ secrets.PROVISIONING_PROFILE }}' | base64 -d -o ~/Library/Mo - name: Import Code-Signing Certificates uses: Apple-Actions/import-codesign-certs@v1 with: p12-file-base64: \${{ secrets.CERTIFICATES_P12 }} p12-password: \${{ secrets.CERTIFICATE_PASSWORD }} - run: flutter build ios - name: XCode Build Archive uses: yukiarrr/ios-build-action@v1.1.1 with: project-path: ios/Runner.xcodeproj p12-base64: \${{ secrets.CERTIFICATES_P12 }} certificate-password: \${{ secrets.CERTIFICATE_PASSWORD }} mobileprovision-base64: \${{ secrets.PROVISIONING_PROFILE }} code-signing-identity: \${{ secrets.CODE_SIGNING_IDENTITY }} team-id: \${{ secrets.TEAM_ID }} workspace-path: ios/Runner.xcworkspace output-path: app-release.ipa export-method: ad-hoc - name: Distribute iOS app run: | curl \ -H "Authorization: token \${{secrets.DEPLOYGATE_API_KEY}}" \ -F "file=@/Users/runner/work/tapiten_app/tapiten_app/app-release.ipa" -F "message=git:\$GIT_HASH" \ -F "distribution_name=\$GIT_BRANCH" \ -F "release_note=new ios build" \ -F "distribution_key=\${{secrets.IOS_DISTRIBUTION_HASH}}" \ "https://deploygate.com/api/users/\${{secrets.DEPLOYGATE_USER}}/apps" 使用したもの • publicなリポジトリ deploygate publicな<u>リポジトリ</u>にした理由は特に隠さなければいけない情報が<u>リポジトリ</u>に含まれていないことと、 GithubActionsの使用制限がpublic<u>リポジトリ</u>の場合は無いらしいから、という理由です。 yamlファイルの説明 ビルドが走るタイミングの設定 on: push: branches: master pull_request: branches: master この部分で、masterブランチにpushされた時と、masterブランチに向けたpull_requestが作 **瞬**にワークフローが実行されるように設定しています。 XCodeのバージョン設定 name: Select Xcode version 11.7 run: sudo xcode-select -s '/Applications/Xcode_11.7.app/Contents/Developer' <u>iOS</u>ビルド時にはたまに<u>XCode</u>のバージョンの差異によってビルドが失敗したりすることもあるので、バージョン を指定しておきます。 **Flutter**のインストールをキャッシュしておく name: setup cache uses: actions/cache@v1 with: path: /Users/runner/hostedtoolcache/flutter key: \${{runner.0S}}-flutter-install-cache この部分でFlutterのインストール先のファイルをキャッシュ対象にしています。 毎回インストールしていると遅くなってしまうので、 同じバージョンでビルドする際にはキャッシュを使った方 早いです。 Flutterのインストール name: install flutter uses: subosito/flutter-action@v1 with: flutter-version: '1.20.2' channel: 'stable' Flutterもローカル環境と同じバージョンをインストールします。 Flutterのライブラリ関連のインストール name: flutter dependencies install run: flutter pub get Flutterのライブラリ関連のインストールを行います。 Provisioning Profileのデコード name: Import Provisioning Profile run: l mkdir -p ~/Library/MobileDevice/Provisioning\ Profiles touch ~/Library/MobileDevice/Provisioning\ Profiles/decoded.mobileprovision echo -n '\${{ secrets.PROVISIONING_PROFILE }}' | base64 -d -o ~/Library/MobileDe <u>Github</u>のsecretsに設定している<u>base64</u>で<u>エンコード</u>したprovisioning_profileを取り出して、デコードします。 攻行コードが含まれないように注意してください。 最後にも記載していますが、改行コードが含まれているだけでデコード後の値が変わってしまいエラーになってし まいます。 Certificateの設定 name: Import Code-Signing Certificates uses: Apple-Actions/import-codesign-certs@v1 with: p12-file-base64: \${{ secrets.CERTIFICATES_P12 }} p12-password: \${{ secrets.CERTIFICATE_PASSWORD }} <u>base64</u>で<u>エンコード</u>したp12ファイルの中身と設定したパスワードからcodesignをします。 run: flutter build ios name: XCode Build Archive uses: yukiarrr/ios-build-action@v1.1.1 with: project-path: ios/Runner.xcodeproj p12-base64: \${{ secrets.CERTIFICATES_P12 }} certificate-password: \${{ secrets.CERTIFICATE_PASSWORD }} mobileprovision-base64: \${{ secrets.PROVISIONING_PROFILE }} code-signing-identity: \${{ secrets.CODE_SIGNING_IDENTITY }} team-id: \${{ secrets.TEAM_ID }} workspace-path: ios/Runner.xcworkspace output-path: app-release.ipa export-method: ad-hoc ipa出力のための<u>アーカイブ</u>にはi<u>OS build actio</u>nを利用させて頂きました。 使用しているSecretsの詳細 名前 内容 mobileprovisionをbase64でエンコードしたもの PROVISIONING_PROFILE p12ファイルをbase64でエンコードしたもの CERTIFICATES_P12 p12ファイルに設定したパスワード CERTIFICATE_PASSWORD CODE_SIGNING_IDENTITY code sign identity Deploygateのアカウント設定>APIKeyに記載されている値です DEPLOYGATE_API_KEY Deploygateのアプリ配布ページURLに記載されているハッシュ値です DEPLOYGATE_DIST_PAGE_HASH Deploygateのアカウントに登録されたユーザ名です DEPLOYGATE_USER 詰まったこと base64のデコード <u>base64にエンコード</u>したprovisioning profileのデコードがどうやらうまく行っていない... というところで数時間を費やしたのですが、原因は最後の改行文字を削除し忘れていた。 base64 sample.mobileprovision | pbcopy で<u>コピーアンドペースト</u>していたのですが、最後の改行文字ももちろん文字として認識されるのでデコードした際 に違う値となってしまっていたのが原因でした。 言われてみればなんてことない些細なミスなのですが、気づくのに相当時間を費やしてしまったので載せておきま す。 マミーめも~ (id:flat-M_M) 191日前 *+ B!ブックマーク (7) シェア 関連記事 2020-09-13 XCodeのビルドを高速化する並列ビルドの設定方法 Codeのビルドを高速化す CithubActionsを使ったiOSのCI/CD環境構築の際にiOSのビルドが... 並列ビルドの設定方法 Github ActionsでAndroidアプリのCI/CDをやってみた FlutterでiOS/Androidアプリを開発していた際に、GithubActions... ubActions x Deploy CI/CD環境を整える! ₩ コメントを書く XCodeのビルドを高速化する並列ビルドの Github ActionsでAndroidアプリのCI/CD ■ えんじにあ雑記!Powered by Hatena Blog | ブログを報告する はてなブログをはじめよう! flat-M Mさんは、はてなブログを使っています。あなたもはてなブログをはじめてみませんか?