drivemode

Keishin Yokomaku / shibuya.apk #13

Automation with Wercker and Container Builder

About Me

- Keishin Yokomaku
  - Drivemode, Inc. / Principal Engineer
  - 
    KeithYokoma: GitHub / Twitter / Qiita / Tumblr / Stack Overflow
    - Books: Mobile App Dev Guide / Android Academia / Grimoire of Android
    - Fun: Gymnastics / Cycling / Photography / Motorsport
    - Today's Quote: "Power is everything."

Automation with Wercker and Container Builder

“Werckerはいいぞ”

Wercker?

- Docker-native CI/CD Automation Platform
  - 自動で Docker コンテナをテスト、デプロイできるサービス
    - Docker イメージを pull してきてコンテナを立ち上げる
    - コンテナ内でコマンドを実行する
  - パイプラインでテストとデプロイを分離できる
    - テストが正常終了したときのみデプロイが実行される

Wercker for Android CI

- Docker イメージさえあればビルドができる
- Android アプリをビルドするために必要なものがあれば...🤔
  - Android SDK & NDK
  - Java
  - etc...

Wercker for Android CD

- Android アプリのデプロイ => apk の配信
- apk を配信してくれるサービスにデプロイすれば...🤔
  - DeployGate
  - fastlane
  - etc...

 “完全に理解した”

Steps

1. Dockerfile から Docker イメージをビルド
2. ビルドしたイメージを dockerhub などのレジストリに登録
3. ビルドしたいリポジトリにwercker.yml を配置
4. Wercker の設定

Building a Docker image

- Dockerfile をつくろう(例)
  - Ubuntu をベースに
  - Android SDK と Java をインストールして
  - 必要な環境変数を整えたイメージ
- e.g. http://bit.ly/2ngAz0S

Pushing the image to a registry

- Wercker が扱えるレジストリ
  - Docker Hub
  - Google Container Registry
  - Amazon ECR
  - Private Registry(quay.io)
- プライベートリポジトリにしましょう (http://bit.ly/2nuwkz9)

Build an image using ‘docker-machine'

```
(mac)                $ docker-machine start image-builder
(mac)                $ docker-machine ssh image-builder
(docker-machine) $ ls
Dockerfile
(docker-machine) $ docker build -t account/repository:tag .
(docker-machine) $ docker login
(docker-machine) $ docker push account/repository:tag
(docker-machine) $ exit
(mac)                $ docker-machine stop image-builder
```

Configure wercker.yml

- Android アプリのリポジトリに YAML ファイルを置く
  - 名前は必ず wercker.yml
- ビルド、デプロイそれぞれのハッシュ内にビルドに必要な手順を書く
  - ビルドとデプロイで異なる Docker イメージを使用可能

Configure wercker.yml

```
build:
  box:
    id: account/repository
    username: $USERNAME
    password: $PASSWORD
    tag: tag
  steps:
    - script:
      name: assemble
      code: |
        ./gradlew —stacktrace —project-cache-dir=$WERCKER_CACHE_DIR assemble
    - script:
      name: test
      code: |
        ./gradlew —stacktrace —project-cache-dir=$WERCKER_CACHE_DIR test lint
```

Configure wercker.yml

```
build:
  box:
    id: account/repository
    username: $USERNAME
    password: $PASSWORD
    tag: tag
  steps:
    - script:
      name: assemble
      code: |
        ./gradlew –stacktrace –project-cache-dir=$WERCKER_CACHE_DIR assemble
    - script:
      name: test
      code: |
        ./gradlew –stacktrace –project-cache-dir=$WERCKER_CACHE_DIR test lint
```

Configure wercker.yml

```
build:
  box:
    id: account/repository
    username: $USERNAME
    password: $PASSWORD
    tag: tag
  steps:
    - script:
      name: assemble
      code: |
        ./gradlew —stacktrace —project-cache-dir=$WERCKER_CACHE_DIR assemble
    - script:
      name: test
      code: |
        ./gradlew —stacktrace —project-cache-dir=$WERCKER_CACHE_DIR test lint
```

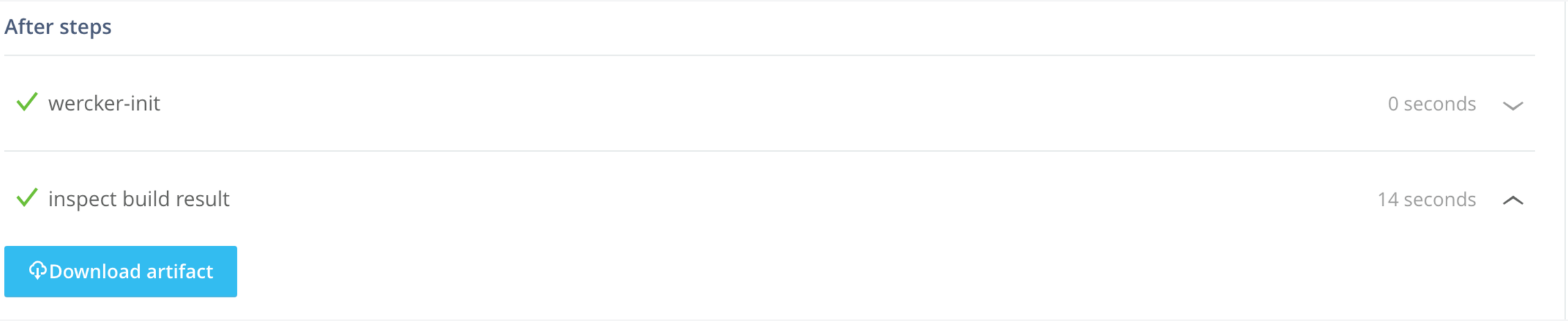
Tips

- 1つのステップは10分でタイムアウト
  - コマンドの実行が長くなるときは分ける
- Wercker の Web インタフェースでの出力がかなりシンプル
  - 困ったときに使いたい情報は echo で出力するステップを作る
  - ログ出力が長いと Web インタフェースが激重なので適宜調整

Saving artifacts

- after-steps に手順を書く
  - 環境変数に保存場所が定義してある
    - $WERCKER_REPORT_ARTIFACTS_DIR
  - 入れたいものだけ cp する
- 保存した成果物は Web からダウンロードできる
  - 場所が分かりづらいので注意

Saving artifacts



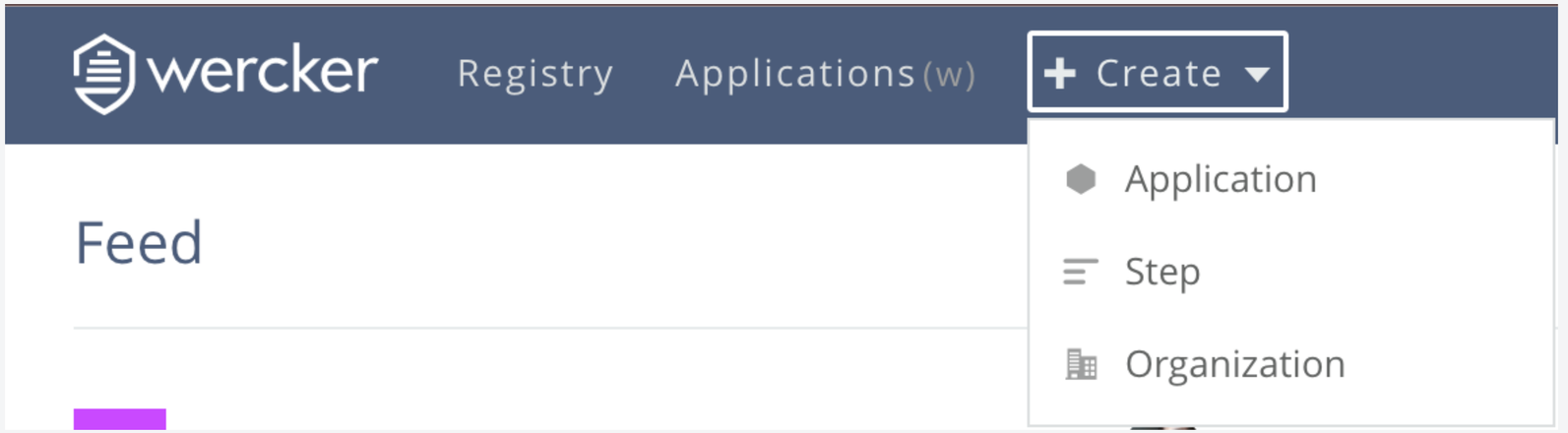
Saving artifacts

```
after-steps:
  - script:
    name: save artifacts
    code: |
      ls -la ./app/build/outputs/
      cp -r ./app/build/outputs/* ${WERCKER_REPORT_ARTIFACTS_DIR}
      cp -r ./app/build/reports/* ${WERCKER_REPORT_ARTIFACTS_DIR}
```

Saving artifacts

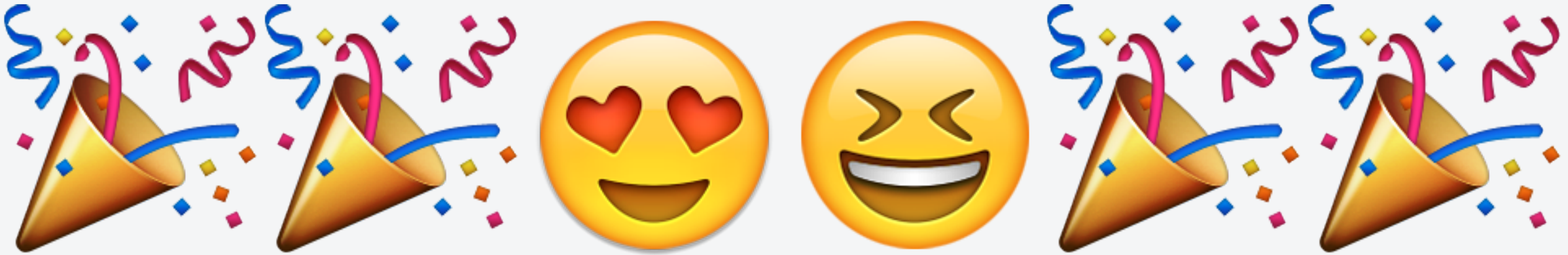
```
after-steps:
  - script:
    name: save artifacts
    code: |
      ls -la ./app/build/outputs/
      cp -r ./app/build/outputs/* ${WERCKER_REPORT_ARTIFACTS_DIR}
      cp -r ./app/build/reports/* ${WERCKER_REPORT_ARTIFACTS_DIR}
```

Connect Wercker with GitHub



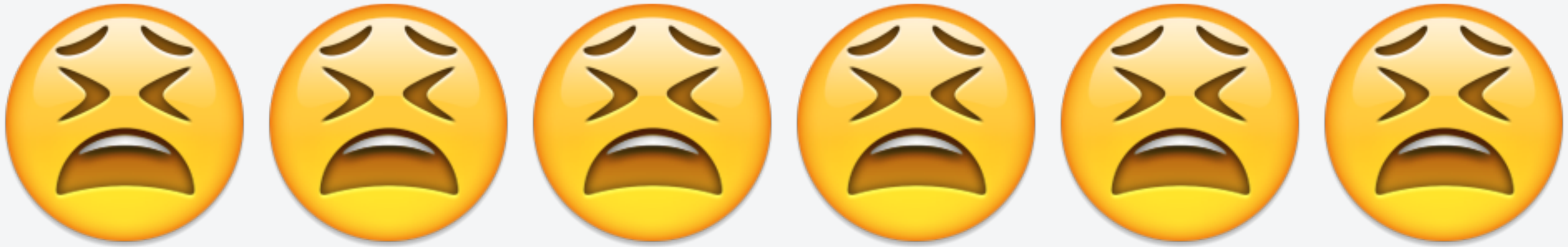
Connect Wercker with GitHub

- リポジトリを選んで...
- オーナーを選んで...
- デプロイキーを追加して...
  - public なプロジェクトの場合は https 経由でアクセスする設定でも OK
- 完了！



運用すると見えてくる Wercker の手間

- SDK のアップデートやサポートリポジトリのアップデート
  - Docker イメージを作り直すことになるが...？🤔
  - 毎回 docker-machine でビルド...？🤔
  - docker-machine と Android Emulator は同時に動かせないぞ...？🤔
  - イメージ作るだけで 10 分以上かかるんだけど...？🤔
  - タグの管理を自分で考えるの面倒くさいんだけど...？🤔



“そこでContainer Builderですよ”

Container Builder?

- これは何？
  - Google のサービス
  - Dockerfile をわたすと自動で Docker イメージを作ってレジストリに登録してくれるすごいやつ

Container Builder?

- これは何？
  - Google のサービス
  - Dockerfile をわたすと**自動で Docker イメージを作ってレジストリに登録**してくれるすごいやつ

Container Builder

- イメージの TAG の管理方法を設定できる
  - 固定で latest のまま
  - git のコミット ID をタグにする
  - etc...
- GitHub と連携できる

Workflow of updating Docker image

- Dockerfile を編集
- コミットを push
- Container Builder がビルドを始める
- 完了したら新しいタグを wercker.yml に設定する

 “完全に理解した”

Configure Container Builder

- Google Cloud Console から Container Registry を開く
- Build triggers から GitHub のリポジトリを選択してトリガーを設定
  - どのブランチの push でビルドを始めるか
  - Dockerfile の場所
  - イメージ名

Configure Container Builder

- Dockerfile のビルド
  - 10分以上かかるとタイムアウトする
  - cloudbuild.yaml をつかって調整するよう変更

cloudbuild.yaml

```
timeout: 40m
steps:
- name: gcr.io/cloud-builders/docker
  args: ['build', '--tag=gcr.io/$PROJECT_ID/$REPO_NAME:$COMMIT_SHA', '.']
images: ['gcr.io/$PROJECT_ID/$REPO_NAME:$COMMIT_SHA']
```

cloudbuild.yaml

```
timeout: 40m
steps:
- name: gcr.io/cloud-builders/docker
  args: ['build', '--tag=gcr.io/$PROJECT_ID/$REPO_NAME:$COMMIT_SHA', '.']
images: ['gcr.io/$PROJECT_ID/$REPO_NAME:$COMMIT_SHA']
```

cloudbuild.yaml

```
timeout: 40m
steps:
- name: gcr.io/cloud-builders/docker
  args: ['build', '--tag=gcr.io/$PROJECT_ID/$REPO_NAME:$COMMIT_SHA', '.']
images: ['gcr.io/$PROJECT_ID/$REPO_NAME:$COMMIT_SHA']
```

cloudbuild.yaml

```
timeout: 40m
steps:
- name: gcr.io/cloud-builders/docker
  args: ['build', '--tag=gcr.io/$PROJECT_ID/$REPO_NAME:$COMMIT_SHA', '.']
images: ['gcr.io/$PROJECT_ID/$REPO_NAME:$COMMIT_SHA']
```

cloudbuild.yaml

```
timeout: 40m
steps:
- name: gcr.io/cloud-builders/docker
  args: ['build', '--tag=gcr.io/$PROJECT_ID/$REPO_NAME:$COMMIT_SHA', '.']
images: ['gcr.io/$PROJECT_ID/$REPO_NAME:$COMMIT_SHA']
```

Change wercker.yml

- Google Container Registry から引っ張ってくるようにする
  - JSON で認証する方式になる
  - username は必ず _json_key
  - 認証情報の入った JSON は環境変数に入れておく
  - gcr.io をレジストリに指定する

Change wercker.yml

```
build:
  box:
    id: account/repository
    username: $USERNAME
    password: $PASSWORD
    tag: tag

box:
  id: gcr.io/<PROJECT_ID>/<IMAGE_NAME>
    username: _json_key
    passwrod: $GCR_JSON_KEY_FILE
    registry: https://gcr.io
    tag: <TAG>
```

$GCR_JSON_KEY_FILE as Environment variable in Wercker CI

```
{
  “type”: “service_account”,
  “project_id”: “PROJECT_ID”,
  “private_key_id”: “hogefugapiyofoobarbaz”,
  “private_key”: “——-BEGIN PRIVATE KEY——-\nhogehoge\n——-END PRIVATE KEY——-”
  “client_email”: “foobar@hogehoge.iam.gserviceaccount.com”
  ……
}
```

Wercker + Container Builder

- ビルド環境を Docker イメージに詰め込める
  - ビルド・デプロイ環境に何が必要かコードに落とせる
  - コードに落とした Dockerfile をリビジョン管理できる
  - イメージを使って CI/CD ができる
  - イメージをホストしてくれる場所から pull するだけで使える

Wercker + Container Builder

- CI 環境を CI/CD する
  - 手元の PC のリソースを使わずにイメージをビルドできる
  - Docker イメージのリビジョン管理を自動化できる
  - 環境を変える時は TAG を変えるだけ

drivemode

Keishin Yokomaku / shibuya.apk #13

Automation with Wercker and Container Builder