AndroidのUIテストが実行できるDockerイメージを作る

前後しますが、この記事 の前提になっている Dockerfile の説明をしたいと思います。

※2017年2月に作ったので、今は、もっとよい方法があるかもしれません

これを読んで得られるもの

- Docker内で完結するAndroidの画面テストの環境の作り方
  - 若干不安定&制約がある、ので実用に足るかは自己判断で

経緯

以前から「画面テストのヘッドレス(オンメモリ)環境」というのを考えていまして…

- ポータビリティがあって
- 「使い捨て」できて
- 実際にUIテストが動いて
- 画面キャプチャが取れて
- 画面動画が取れて
- HostOSを汚さない
  - Host側にデスクトップ(ウィンドウマネージャ)を入れなくてよい

- 安価である
  - 大仰な投資・設備が無くとも出来る
  - 近所に落ちてるPC程度

な仮想環境が出来ないかなぁ、と模索しています。

最近は「Dockerで作れば自然にそうなるよな」と思い、過去 Webブラウザのテスト環境 などを作ったりしました。

※今は公式にイメージがあります

で、Androidの「UIテスト」は？

まー「Firebase Test Labとか金詰めばなんの苦労もなくなるよ!」…なんでしょうけど、金要るヤツは手軽に試せない、のでDockerでやりたい。

でもググると 出来なかった、というエントリが上に来たり するので、なんか「避けるべきもの」として認識してるのかな？という感じも？

でも、Docker Hub を漁って、その元となる Dockerfile をgithubから探すと、いくつか例があるようです。

- https://github.com/softsam/docker-android-emulator
- https://github.com/ksoichiro/dockerfiles/tree/master/android-emulator

これを合成しつつ、自分が欲しい感じに Dockerfile を作っていきます。

やったこと

Dockerfile作成

こちらにあります

Dockerfile詳細

主要なところだけ、解説していきます。

```
# Install all dependencies
RUN apt-get update && \
    apt-get install -y wget openjdk-8-jdk-headless libc6-i386 lib32stdc++6 l
    apt-get clean && \
    apt-get autoclean && \
    rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*
```

Ubuntuをベースイメージとしているので、AndroidEmulatorを動かすのに必要なライブラリを apt-get にてパッケージ取得/インストールし、キャッシュを削除しています。

OpenJDK8 を指定していますが、それ以降のものでも大丈夫だと思います。

```
# Install android tools + sdk
ENV ANDROID_HOME /opt/android-sdk-linux
ENV PATH $PATH:${ANDROID_HOME}/tools:$ANDROID_HOME/platform-tools
```

コンテナ内での「SDKインストールディレクトリ」を /opt/android-sdk-linux と定義し作成、ANDROID_HOME , PATH 環境変数を定義してます。

```
# Set up insecure default key
RUN mkdir -m 0750 /.android
ADD resources/insecure_shared_adbkey /.android/adbkey
ADD resources/insecure_shared_adbkey.pub /.android/adbkey.pub
```

コンテナ起動時には、ユーザが root 、ホームディレクトリが / になる(なんで /root じゃないんだw)ので、/ 直下に .android/ ディレクトリを作成し、そこに「ADB起動時に必要な鍵ファイル」を配置しています。

```
RUN wget -qO- "http://dl.google.com/android/android-sdk_r24.4.1-linux.tgz" |
    echo y | android update sdk --no-ui --all --filter platform-tools --forc
```

AndroidSDK をダウンロード、/opt に展開、 Android SDK Tools の更新、をしています。

android-sdk_r24.4.1-linux.tgz は「更新されていくところ」なので、定期的に編集が必要かもです。

途中 echo y を噛ましているのは「対話型で"y"入力が必要」なところの回避のためです。

```
RUN mkdir -m 0750 "$ANDROID_HOME/licenses/"
RUN echo -e "\n8933(略)" > "$ANDROID_HOME/licenses/android-sdk-license"
RUN echo -e "\n8483(略)" > "$ANDROID_HOME/licenses/android-sdk-preview-licen
```

この記事で解説してます「SDK実行時に必要なライセンスファイル」を仕込んでます。

```
# Install dependencies for emulator
RUN echo y | android update sdk --no-ui --all -t `android list sdk --all|gre
    echo y | android update sdk --no-ui --all -t `android list sdk --all|gre
```

エミュレータの「種類の取得」と「選択してのインストール」を行っています。

android update sdk で「必要なプラットフォーム(CPU種とAndroidAPIバージョン)」を指定するわけですが…何故か「決められた番号」を入力/指定するようになってます。

そこで andorid list sdk コマンドで取得した「一覧のテキスト」から、grep/awk で目的なものをフィルタし、引数に送ってます。

ここは、実際にテストを流す「AndroidStudioで作られたプロジェクトのビルドファイル」(app/build.gradle 内の android 節の設定)に対応したものを指定します。

※今回、対象のプロジェクトの app/build.gradle は以下の感じでした。

```
...
android {
  compileSdkVersion 23
  buildToolsVersion "21.1.2"
  ...
}
```

```
RUN echo n | android create avd --force -n "arm" -t android-21 --abi default
```

Androidエミュレータを作成しています。

末尾は「動かす際の画面サイズ」なのですが、文字列で指定する必要があり ここらへん から拾いました。

```
# Avoid emulator assumes HOME as '/'.
ENV HOME /root
ADD scripts/wait-for-emulator /usr/local/bin/
ADD scripts/start-emulator /usr/local/bin/
ADD scripts/init-emulator /usr/local/bin/
```

エミュレータを使ってのテストコマンドを実行するための補助スクリプトをイメージ内に仕込んでいます。

- start-emulator
  - テストコマンドのランチャー。引数に「実際に起動したいテストのコマンド」を指定する。内部では init-emulator を呼ぶ。

- init-emulator
  - emulatorコマンドを実際に呼び、反応可能になるまで待つ。内部では wait-for-emulator を呼ぶ。

- wait-for-emulator
  - emulatorが反応するまで何度かリトライ&スリープして待つ。

```
# 暖機運転(anrdoidのJavaProjectを作り、ビルド、gradle実行可能かテスト)
# RUN mkdir -p /opt/tmp && android create project -g -v 0.9.+ -a MainActivit
# 上記をやりたかったが、android-sdkのバグでコケるため「完成品のgradleプロジェクトサンプ
RUN mkdir /opt/tmp/
ADD resources/sample-proj.tgz /opt/tmp/
# ADDの効力で、tgzは展開されているはず
RUN cd /opt/tmp/sample-proj && ./gradlew task clean test
RUN rm -rf /opt/tmp
```

一度「AndoridStudioで作られたプロジェクト」を送り込んで、ビルドしています。

AndroidSDKはインストール(というより配置かな)したとしても、実際にビルド(gradleコマンドでコンパイル)する時に利用するファイルの大半をダウンロードするので、その時でないと「仕込みが失敗しているか」が確認できません。

そこで一度「動作確認のためのビルド」を行っています。

(コメントにもあるように「その場でプロジェクトをコマンドで創りだして、それをビルド」したかったのですが、出来なかったのでサンプルプロジェクトを送り込んで実行しています)

ビルド＆テスト実行方法

ビルドは、Dockerfile のあるディレクトリでコンソールから

```
docker build . -t android-emulator-image
```

と実行します。(android-emulator-image はイメージ名なので、変更頂いて構いません)

エミュレータを使わないテスト(GUIを起動しないUnitテスト)は、Androidのプロジェクト(build.gradleが在るところ)へ移動、コンソールから

```
docker run -t -i -v `pwd`:/workspace android-emulator-image ./gradlew clean
```

エミュレータを使うテストは、コマンドを一つかまして、

```
docker run -t -i -v `pwd`:/workspace android-emulator-image start-emulator "
```

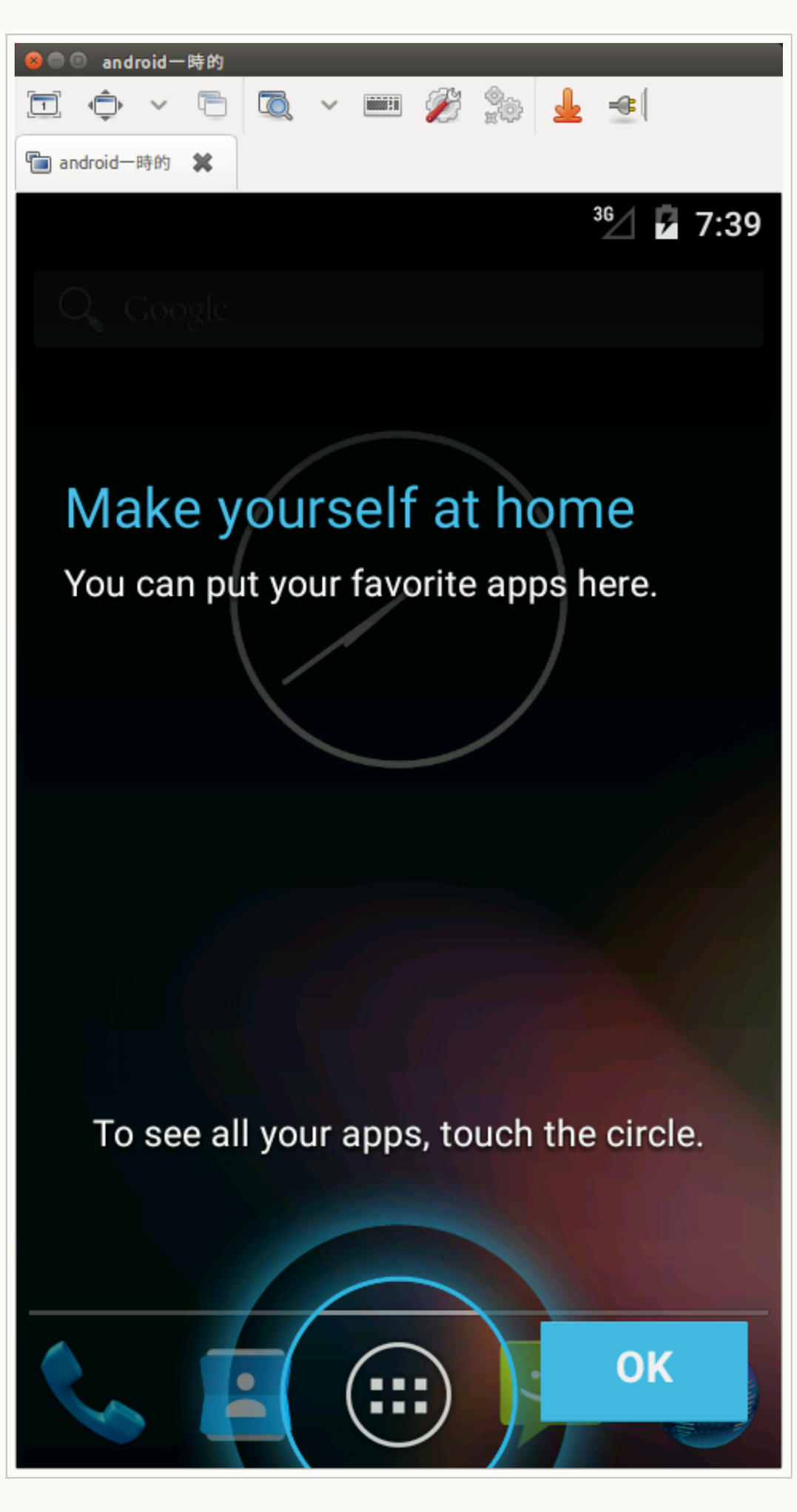
という風に実行します。

実行中のエミュレータ状態の確認

Android Emulator には「VNCで画面が観られる機能(オプション)」があり、今回作ったDockerイメージにも「ポートを外だしの考慮」があるため、

- docker run 時に -p オプションで 5900 ポートを開放し localhost:5900 でVNCViewer接続
- docker inspect でコンテナのIP(172.17.0.x)を調べ、 (コンテナのIP):5900 でVNCViewer接続

によって、 docker run 実行中の様子をVNCViewerで確認することができます。



※画像はロック画面ですが、画面下の○をフリックするとテストの動きが観れます

画面は内部的に存在するので、「テスト時にspoon等キャプチャライブラリで画面ハードコピーを撮る」はもちろん可能です。

実証していませんが、経験から「VNC上にデスクトップがある」なら、RecordMyDesktop コマンド等キャプチャツールにより"内側から"動画を撮ることも可能…なはずです。

問題・課題

運用していく上で、いろいろありました。

動くが問題あるもの

- 安定しない
  - テストによっては「謎の突然死」がある

- ある程度のイメージサイズとなる(4GBくらい)
- 遅い
  - Dockerイメージのビルドが遅い
  - 実際のテストが遅い
  - emulatorを使ったテストは更に遅い

ロケール/タイムゾーンが英語圏

エミュレータを設定し「ロケール/タイムゾーンを日本にする」ことが出来るのですが、「テスト中に謎の突然死を遂げる」ので、デフォルトにしています。

一応「仕込むところと仕込み方」は用意してあるのですが…。

動かすエミュレータの「プラットフォームとAndroidAPIバージョン」について

今回、動かすためにつかった「エミュレータのプラットフォーム」は

- SDK Platform Android 5.0.1, API 21
- ARM EABI v7a System Image, Android API 21, revision 4

ですが、これを選んでいる理由は…

- x86系は動かない : kvmが必要だ、と言われるがその仕込み方が解らない
- APIが新しすぎるものは極端に遅くなる : 19が動くもので最速だったので21でも遅い

など、問題と試行錯誤があったからです。(すべての組み合わせを試したわけではありません)

恐らくですが、ARMは「ソフトウェア上でエミュレーションしている(仮想OSソフトを必要としない)」分、「挙動遅いが依存が少ない」となり、Dockerで動かすには有利に働くのかな、と。

この状況のせいで「Dockerでのテスト自体の性質/用途」が

- 色んな環境での実行を試す

という使い方は出来ず、

- 一般的な環境での挙動に問題ないかを試す

程度に、制限されてしまいます。(すごく残念…なのでなんとかしたい)

所感

Googleかつ普及したプラットフォームであるので「定番なやり方が在るだろう」と調べ始めたのですが…案外無かったですね。

もしかしたら「最適な定石があって、それを見つけられてない」のかもしれない…ので！ツッコミをお待ちしております。

※記事通りやって動かない場合は github側は動くように維持してます ので、そちらをご確認ください。

追記

Android SDK の利用について、指摘を頂きました。

Koji Hasegawa
@nowsprinting

返信先: @kazuhito_mさん

Android SDKの再配布にあたるのでライセンス云々という話は大丈夫になったのでしたっけ？

午前7:51 · 2017年10月17日

1 ツイートへのリンクをコピー

もうひとつ解ってなかったのですが…「DockerHubにSDK入りのイメージ上げるのはダメ」なのですね。

(この界隈がもうひとつ盛り上がってない(風に見える)のは、こういう理由もあるのかも)

幸いこの Dockerfile は、「ローカルでビルドして使う」前提でしたし、DockerHubにビルド済みイメージはあげてないので問題は無いと思われます。

利用の際は「ローカル環境でのbuild & run」でいきましょう。

参考

以下を参考にさせていただきました。感謝。

- Andoridをヘッドレスで動かす代表的なスライド
  - https://speakerdeck.com/muran001/android-app-ui-testing

- cliのヘルプ
  - http://blog.tyye.net/2015/05/cui-android-sdk.html

- dockerでandroid-emulatorを動かしてるヒトのリポジトリ
  - https://github.com/ksoichiro/android-tests/tree/master/docker-emulator

- dockerで(ry)のyoutube(なにやってるかよくわからない)
  - http://www.2daygeek.com/install-sdk-android-emulator-on-ubuntu-centos-debian-fedora-rhel-opensuse-arch-linux-mint/

- KVMをDockerで動かす方法
  - http://qiita.com/flny/items/ba1ac612d6bea11387ad

- Android SDK CUI インストール 手順
  - http://qiita.com/granoeste/items/1c1f72e93847446e0157

- 失敗報告
  - http://qiita.com/butada/items/aaaa99ffc1715f325b91

- Androidサイズ早見
  - http://genmaicha460.blog27.fc2.com/blog-entry-70.html

吉田裕範 · 3年前

docker for windowsで、イメージをbuildし、
VNC接続を試みているところです。

docker run 時に -p オプションで 5900 ポートを開放し localhost:5900 でVNCViewer接続を、

Realvncの VNC viewerを使用して接続したのですが、
「The connection closed unexpectedly」となり接続できませんでした。

netstat コマンドで、ポートの状況を見ても、開放できているように見えますが、
TCP [::1]:5900 [::]:0 LISTENING

何か、他に気にすべきところがあれば、
教えていただけないでしょうか？