

研究発表

Dockerfileのビルドエラーの自動修正 に向けたエラーの原因の調査

辻 裕真

背景

Dockerなどのコンテナ技術はソフトウェアのインフラを構築する軽量仮想化技術として近年関心を集めている.

そのコンテナを構成するDockerfileのビルドは成功率が他の言語と比べて低く, 約34%ものDockerfileがビルドに失敗する.[1]

Dockerに関する研究はパフォーマンスに関わるものが多く, ビルドに関する研究はあまり行われていない.

[1]Jürgen Cito, Gerald Schermann, John Erik Wittern, Philipp Leitner, Sali Zumberi and Harald C. Gall.
An Empirical Analysis of the Docker Container Ecosystem on GitHub. Proceedings of the IEEE/ACM
14th International Conference on Mining Software Repositories, pp. 323-333, 2017

背景

- Dockerfileのビルド失敗時に修正に要する時間
 - 124.5分（平均）， 44.2分（中央値）
 - C++(中央値：5分)やJava(中央値：12分)と比べるとかなり長い[2]

[2]Yiwen Wu, Tao Wang, Yang Zhang and Huaimin Wang.

An Empirical Study of Build Failures in the Docker Context. Proceedings of the 17th International Conference on Mining Software Repositories, pp. 76–80, 2020

目的とアプローチ

- ・ 目的： Dockerfileのビルドエラーの原因を明らかにすること
- ・ アプローチ
 - ・ ビルドログとDockerfileからビルドエラーの原因を調査する
 - ・ エラーが起こるDockerfileの特徴
 - ・ エラーが起きやすい命令の種類

その後の研究では、ビルドエラーが起こったDockerfileを自動で修正するシステムを開発し、Dockerfileの修正に対する負担を減らしたい。

コンテナとは

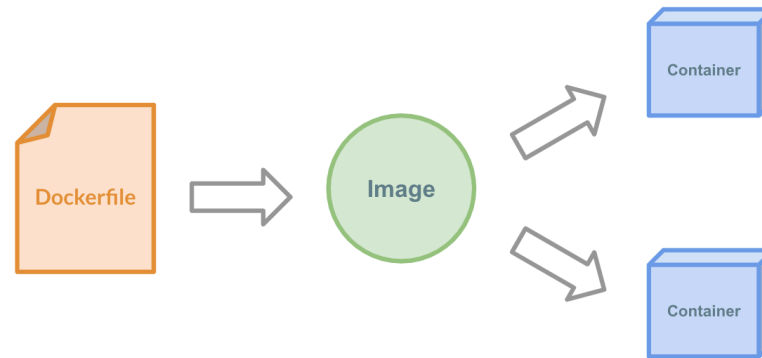


- 高速に実行できるアプリケーション環境
- 仮想マシンは別のホストOSをエミュレートしているが、コンテナは基盤となるカーネルをホストと共有するため、オーバーヘッドが大幅に削減される。

Dockerとは

- Dockerはコンテナを作成, 実行するためのプラットフォームの1つである.
- 広く利用されており, Docker Hubなどのリポジトリによって Dockerイメージの保管や共有が行える.

Dockerfile



- DockerfileをビルドすることでDockerイメージが作成される.
- Dockerイメージからコンテナを起動できる.
- コンテナのベースとなるイメージを指定したり, コマンドの実行, ファイルの追加などの様々な命令を記述している.

Dockerfileとエラーの例

Dockerfileの例

```
FROM centos:7  #centos:7をベースイメージに指定

RUN yum install -y java  #javaのインストール

ADD files/apache-tomcat-9.0.6.tar.gz /opt/  #指定したファイルをコンテナ内にコピー

CMD [ "/opt/apache-tomcat-9.0.6/bin/catalina.sh", "run" ]  #指定したコマンドを実行
```

1行目のFROM命令で存在しないタグを指定した場合

```
Step 1/4 : FROM centos:hoge
manifest for centos:hoge not found: manifest unknown: manifest unknown
```

上記のエラーメッセージが出力されビルドに失敗する.

先行研究との違い

- Yiwen WuらはDocker Hubに公開されているビルドデータからDockerfileがビルドに失敗する頻度, 修正までにかかる時間, ビルド失敗の頻度と修正時間の時間経過による変化の調査を行っているが, ビルド失敗の原因には触れられていない.[2]
- Jürgen Citoらはランダムにプロジェクトを選出しDockerfileのビルドを行い失敗した割合, ビルドにかかった時間を調査していたがこちらもビルド失敗の原因には触れられていない.[1]
- 本研究では実際にプロジェクトのDockerfileをビルドし, ビルド失敗の原因を調査することを目的としている.

Research Questions

RQ1：ビルドに失敗するDockerfileの特徴は何か.

ビルドに失敗したDockerfileのサイズ, コミット日時, Linterの結果などからビルドに失敗しやすいDockerfileの特徴を調査する.

RQ2：Dockerfileのビルドエラーの原因は何か.

ビルドに失敗した際, エラーログからDockerfileのどの命令でなぜエラーが起こったのかを調査する.

調査方法

MSRのDockerのデータセット[3](GitHub上のDockerfileが含まれているプロジェクトのセット, 16959のプロジェクトが含まれている)でプロジェクトをフォーク数, スター数がともに50以上のものに絞る.

プロジェクト内のDockerfileが変更されたコミットを特定し, 各リビジョンでビルド(docker build, Linterなど)を行い, ビルドが失敗した場合は原因の特定を行う.

コミットを遡りビルドを行う理由はビルドに失敗したDockerfileがどう修正されたかを今後確認するためである.

[3]<https://github.com/sealuzh/msr18-docker-dataset>

データセット

- ・ 対象プロジェクト：77プロジェクト
- ・ プロジェクトに含まれるDockerfileの総数：491
- ・ 全てのDockerfileの総リビジョン数(総ビルド数)：2596

調査結果

データの基本情報

- ・ 各プロジェクトに含まれるDockerfileの数

平均：6.4 中央値：3 範囲：1～49

- ・ 各Dockerfileのリビジョン数

平均：5.3 中央値：2 範囲：1～81

- ・ ビルド結果

成功：879(34.7%) 失敗：1653(65.3%)

調査結果

エラーが起きた命令

RUN	782(47.3%)
FROM	312(18.9%)
ADD	312(18.9%)
COPY	195(11.8%)
ENV	5(0.3%)

ビルド以前のエラー：47(2.8%)

調査結果

RUN命令のエラー(782件)

- apt-get installによるエラー：236
- wgetによるエラー：71
- nom installによるエラー：31

様々なエラーでビルドが失敗している.

RUN命令では複数のコマンドが実行されることが多いので, どのコマンドでエラーが発生したかを判断するのが難しく, エラーの原因が特定できていないものが多い.

調査結果

ADD, COPY命令のエラー(507件)

- ・ ファイル, ディレクトリが存在しない : 429(84.6%)
- ・ ソースファイルが指定されていない : 74(14.6%)
- ・ 指定されたファイルにアクセスできない : 1(0.2%)
- ・ 原因不明 : 3(0.6%)

調査結果

FROM命令のエラー(312件)

- ・ ベースイメージが存在しない：168(53.8%)
- ・ ベースイメージの指定方法が間違っている：41(13.1%)
- ・ ベースイメージに指定したタグが存在しない：36(11.5%)
- ・ ベースイメージが空白である(指定されていない)：14(4.5%)
- ・ 原因不明：53(17.0%)

調査結果

ENV命令のエラー(5件)

- ・ 必要なシングルクォートが存在しない：5(100%)

調査結果

ビルド以前のエラー(47件)

- ・ 存在しない命令が使われている : 38(80.9%)
- ・ DockerfileがFROM命令から始まっていない : 8(17.0%)
- ・ ENV命令に引数が1つしかない : 1(2.1%)

現在行っていること

- ・ ビルドが失敗するDockerfileに見られる特徴の調査
- ・ 論文の執筆

今後の作業

- ・ 調査結果をまとめて論文を書く.