

Dockerfileの自動修正に向けた、 依存関係とビルドエラーの 関係分析

発表者：坂本

背景

▶ 近年のソフトウェア開発は迅速な開発が主流

- 迅速な開発の障壁として. .

開発環境, テスト環境, 本番環境の違い

→ 必要なパッケージがない, パスが通っていない, など

組織・企業のメンバーそれぞれの開発環境の違い

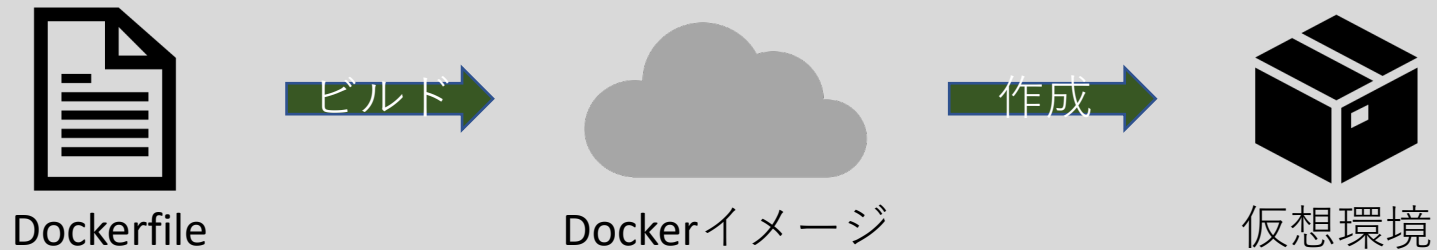
→ メンバーでOS, アプリケーションの種類やバージョンが違う



Dockerを利用

Docker

- ▶ 仮想環境を作成，配布，実行するためのプラットフォーム
 - 開発のためのOSやアプリケーションがパッケージ化される



メリット

- 複数のコンテナが軽量で高速に動作
- 環境の配布が容易
- 環境構築が容易

Dockerfile

- ▶ イメージを作成するための命令を記述したテキストファイル
 - 基本的に1行につき1命令が記述され、1レイヤを構成
 - Dockerfileからイメージがビルドされる

```
FROM ubuntu:20.04 ← ベースイメージ
COPY . /app
RUN apt-get update && ¥
    apt-get install -y python2.7 && ¥
    apt-get install -y pip=20.3.4
RUN pip install numpy==1.18.1
```

FROM：新しいビルドステージを構築

ベースイメージを指定 or 最低限のコンテナを指定

COPY：ファイルやディレクトリをイメージ側にコピー

RUN：ビルド時に実行するシェルコマンドを指定

Dockerfileの依存関係

1. Dockerfile内の依存関係

```
FROM ubuntu:16.04
COPY . /app
RUN apt-get update && ¥
    apt-get install -y python2.7 && ¥
    apt-get install -y pip=20.3.4
RUN pip install numpy==1.18.1
```

pipのバージョンとnumpyのバージョンに互換性がない

2. Dockerfile間の依存関係（継承）

```
FROM ubuntu:20.04
COPY . /app
RUN apt-get update && ¥
    apt-get install -y python2.7 && ¥
    apt-get install -y pip=20.3.4
RUN pip install numpy==1.18.1
```

Ubuntu20.04ではpip20.3.4がサポートされていない

継承

▶ 継承

- ベースイメージに含まれるバージョン情報も引き継ぐ

A:latest

pip 20.3.4
Python2.7
Ubuntu:16.04



Dockerfile

```
FROM A:latest  
RUN pip install numpy==1.18.1
```

イメージ(A)で使用しているPython2.7とnumpyのバージョンの整合性がない



ビルドエラー

先行研究

▶ Dockerfileのビルドエラー自動修正の必要性

- Dockerfileのビルドエラーは**34%**以上失敗している[1]
- ビルドエラーの修正は他の言語の修正よりも時間がかかる[2]

▶ Shipwright

- ルールベースでエラーを分類
- Dockerfileを自動修正するためのツール
- Dockerfile内の命令間の依存関係のみを考慮して修正
- ビルドエラーが発生するDockerfileのうち約**19%**のみを修正

Shipwrightの問題点

Dockerfile内の依存関係のみを考慮して自動修正をおこなっており、
親イメージとの継承関係が考慮されていない

[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

[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

本研究の着眼点

▶ 着眼点

- ビルドに時間がかかるのは、**Dockerfile**が有する 2 つの依存関係が複雑だからではないか

▶ 動機

- イメージ間の依存関係を明らかにすることで、**Dockerfile**の迅速な修正に役立つ

目的とRQ

▶ 目的

- 依存関係（イメージ間，命令間の依存関係）とビルドエラーの関係を明らかにする

▶ RQ

- RQ1：Dockerfile内の依存関係はどの程度存在するか
- RQ2：Dockerfile間の依存関係は存在するか
- RQ3：継承の深さはビルドエラーに影響があるか

データセット

- ▶ Shipwrightで使されたデータセットを使用
- ▶ 変数でベースイメージをビルド時に指定するDockerfileなど
 - 3,625/20,480
→これは除外
- ▶ オフィシャルイメージを使用しているDockerfile
 - 12,269/20,480
- ▶ ユーザイメージを継承しているDockerfile
 - 4,369/20,480
- ▶ 継承を用いていないDockerfile
 - 470/20,480

RQ1

RQ1:Dockerfile内の依存関係はどの程度存在するか

- ▶ 先行研究のエラー分類を基に，エラーが発生している Dockerfile のビルドログを目視
- ▶ 19個のルールを作成
 - 14.1% (764件)がDockerfile内の依存関係が原因でビルドに失敗
- ▶ エラーの原因
 - ほとんどがバージョンに互換性がない

RQ2

RQ2:Dockerfile間の依存関係は存在するか

- ▶ 継承を使用していないDockerfileのビルド失敗率
 - 11.0% (52/470)
- ▶ 継承を使用しているDockerfileのビルド失敗率
 - 26.9% (4,484/16,638)
 - オフィシャルイメージを継承しているDockerfile
25.8% (3,166/12,269)
 - ユーザーイメージを継承しているDockerfile
30.1% (1,318/4,369)

RQ3

RQ3:継承の深さはビルド結果に影響するか

▶ オフィシャルイメージの継承の深さ

- 各継承の深さのオフィシャルイメージを継承しているDockerfile

深さ	1	2	3	4	5	6
失敗件数/件数	1178/3512	146/911	13/58	7/14	65/327	1/6
失敗率	33.5%	16.0%	22.4%	50%	19.8%	16.6%

▶ 継承を使用していないイメージを使用しているDockerfileのビルド失敗率が高い

RQ3

RQ3:Dockerfile間の依存関係は存在するか

▶ ユーザイメージの継承の深さ

- 各継承の深さのユーザイメージを継承しているDockerfile

深さ	0	1	2	3	4	5
件数件数/件数	223/730	44/131	19/68	8/21	1/4	1/1
失敗率	30.5%	33.5%	27.4%	38.0%	25%	100%

▶ 深さが深くなるごとにビルド失敗しやすくなる傾向

今後の予定

- ▶ 継承を使用している Dockerfile から Dockerfile 内の依存関係が原因でエラー発生している Docekrfile を除く