

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

発表者：坂本

ソフトウェア開発の障壁

| 近年は迅速なソフトウェア開発が求められる

- プロダクト納期の短期化
- 企業間の競争力激化

| 迅速な開発の障壁

- チーム開発におけるメンバー間の開発環境の違い
- 開発, テスト, 本番環境の違い

→環境の違いによるエラーか, ソースコードが引き起こすエラーなのか
特定に時間がかかる

障壁を乗り越えるために, Dockerの普及が加速

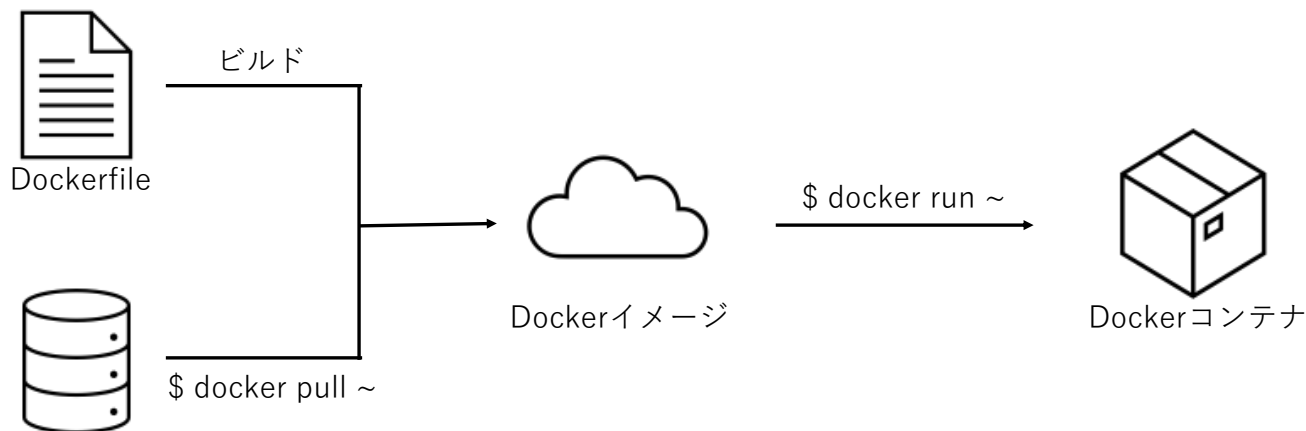
Docker

| コンテナ型仮想化技術

- アプリケーション環境を隔離空間に構築する技術
- 開発のためのOSやアプリケーションをパッケージ化

| メリット

- 環境の配布，再現が容易
- 環境構築が容易



Dockerイメージ

| 特定の環境のスナップショット

- DockerHub* で多くのDockerイメージ (以下イメージ)が共有
 - 500万件以上のイメージ
- あるイメージからは同様の環境が作成される
- 複数のレイヤの集まりとして成り立つ

Dockerfile

| アプリケーション環境の作成手順を記したテキストファイル

- 1行に 1命令を記述
- 1つの命令が 1つのレイヤに対応

```
1 FROM ubuntu:20.04
2 RUN apt-get update && apt-get install python3
3 COPY . /app
4 WORKDIR /app
```

| ベースイメージ

- FROM命令でベースイメージを指定
- 既存のイメージをベースイメージに使用 (継承)
 - ベースイメージに新たな機能を追加
- フルスクラッチでの記述も可能

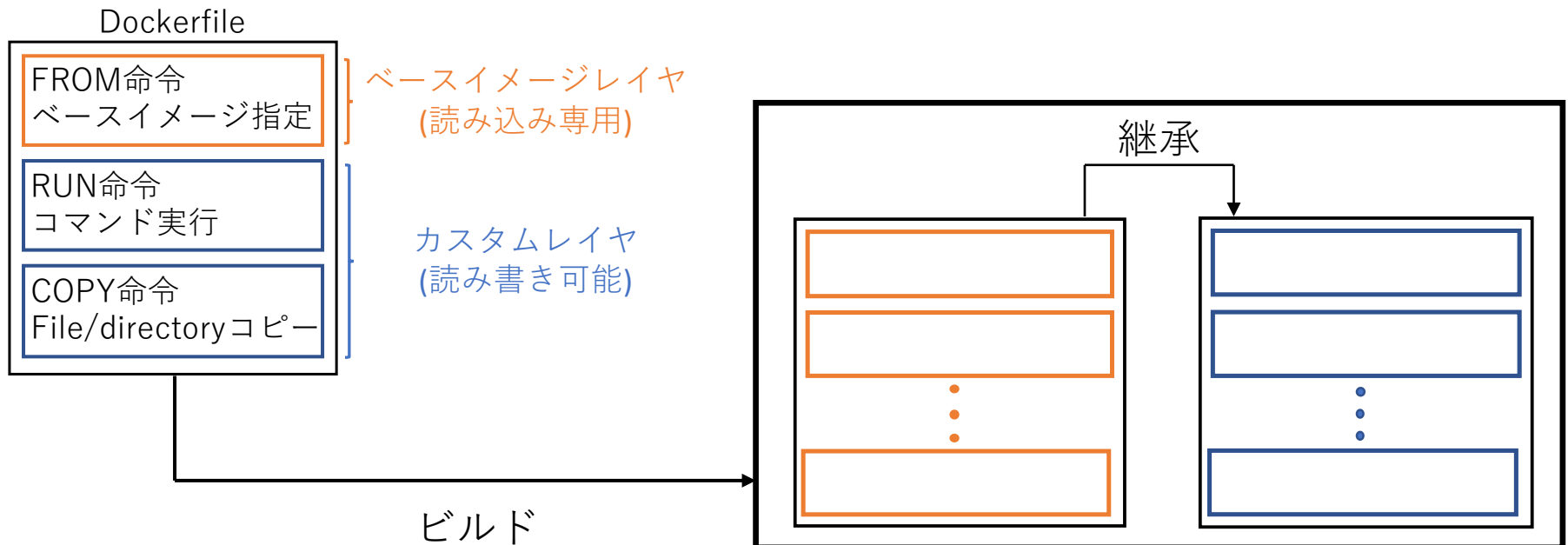
イメージのレイヤ構造

| ベースイメージレイヤ

- ベースイメージが構成するレイヤ

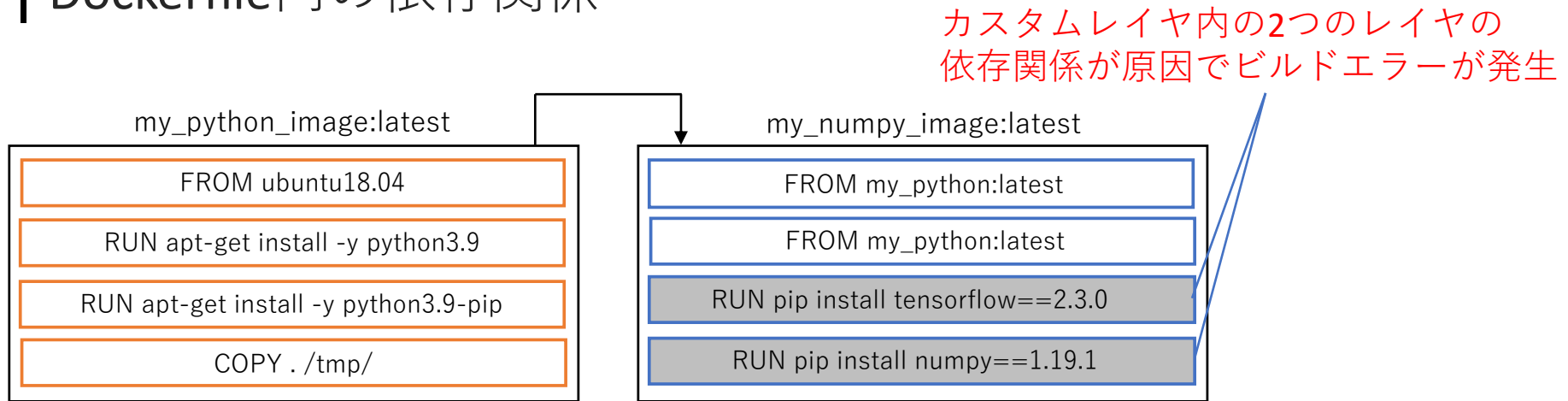
| カスタムレイヤ

- 開発者が新たに記述した命令が生成するレイヤ

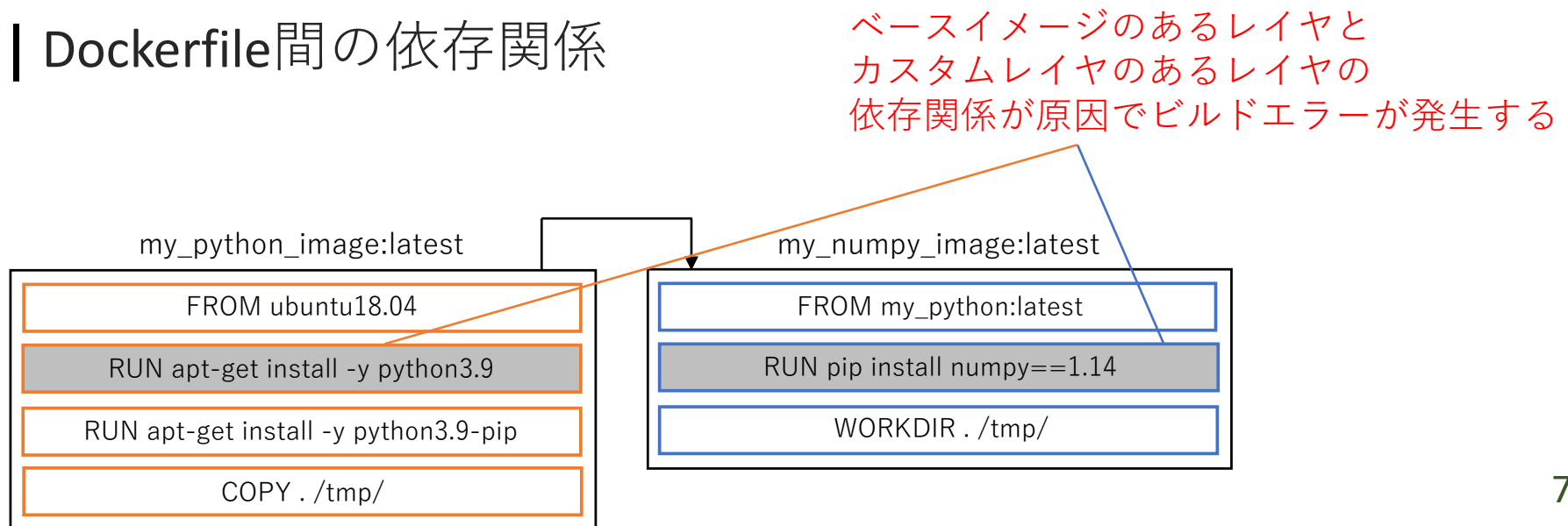


Dockerfileの依存関係

| Dockerfile内の依存関係



| Dockerfile間の依存関係



ビルドエラー自動修正の必要性

| ビルドエラーに関する既存研究

- Dockerfileのビルドエラーは34%以上失敗^[1]
- ビルドエラーの修正には多くの時間がかかる^[2]

| Shipwright^[3]

- Docekrfileを自動修正するためのツール
- ルールベースでエラーを分類
- ビルドエラーが発生するDockefileのうち約19%のみを修正

Shipwrightの問題点

2つの依存関係のうち、Dockerfile内の依存関係のみを考慮
ベースイメージとの継承関係が考慮されていない

[1] Cito, J., Schermann, G., Wittern, J. E., Leitner, P., Zumberi, S. and Gall, H. C.: An Empirical Analysis of the Docker Container Ecosystem on GitHub, *Proceedings of MSR '17*, pp.323–333 (2017).

[2] Wu, Y., Zhang, Y., Wang, T. and Wang, H.: An Empirical Study of Build Failures in the Docker Context, *Proceedings of MSR '20*, pp. 76–80 (2020).

[3] Henkel, J., Silva, D., Teixeira, L., d'Amorim, M. and Reps, T.: Shipwright: A HumanHuman-in-the-Loop System for Dockerfile Repair, *Proceedings of ICSE '21*, pp. 1148–1160 (2021).

着眼点

| 着眼点

- ビルドエラーの修正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

- Dockefileの依存関係を明らかにすることで、 Dokcerfileの迅速な修正に役立つ

目的 & RQ

| 目的

- Dockerfileの依存関係とビルドエラーの関係を明らかにする

| RQ

- RQ 1 : Dockerfile内の依存関係によるビルドエラーはどの程度存在するか
- RQ2 : Dockerfile間の依存関係によるビルドエラーはどの程度存在するか
- RQ3 : 継承の深さはビルドエラーに影響があるか
- RQ4 : 2つの依存関係によるビルドエラー修正時間に差はあるか

データセット

| Shipwrightで使用されたデータセットを使用

- スター数10以上で単一のDockerfileを持つ20,526件のリポジトリ

| フィルタリング

- マルチステージビルドを使用しているDockerfile

対象リポジトリ : 17,325件
ビルドエラーが発生しているDockerfile : 4,698件

RQ1 (Dockerfile内の依存関係が原因のビルドエラーはどの程度存在するか)

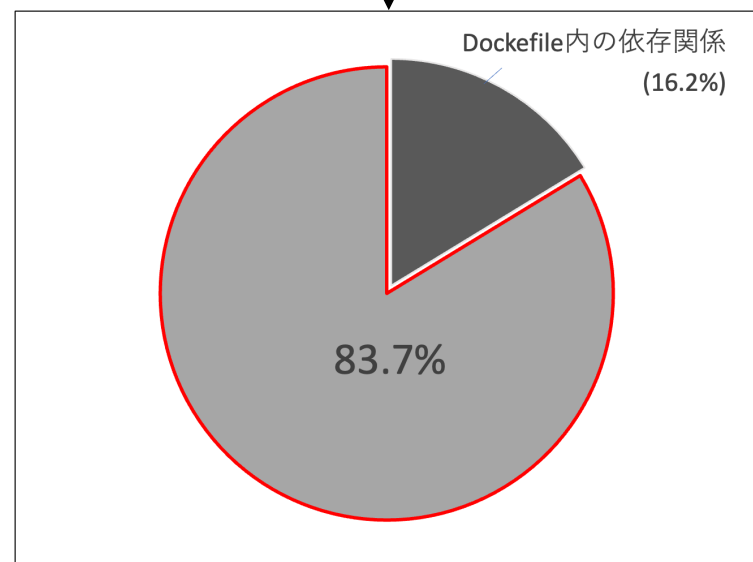
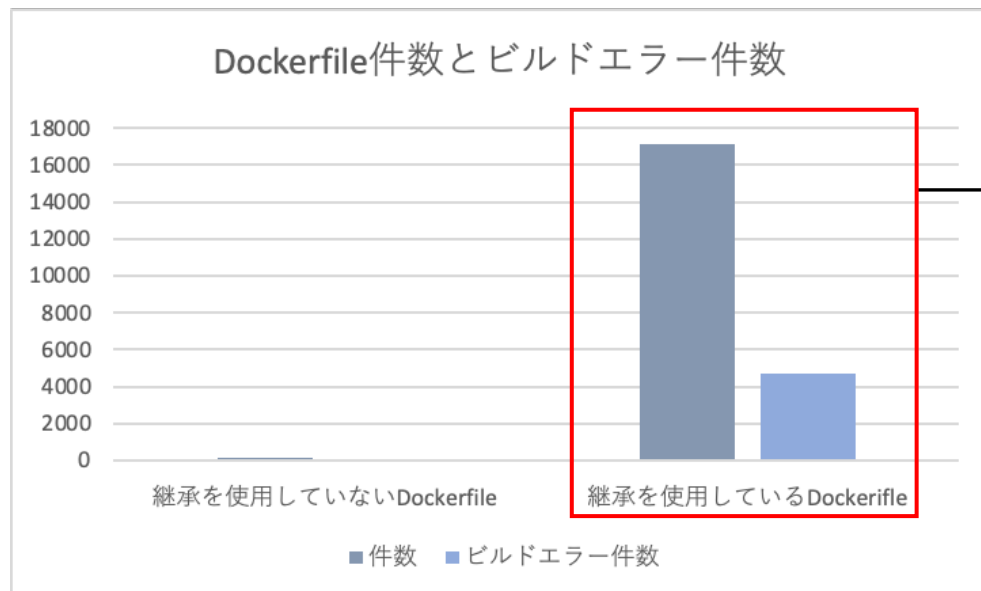
| エラーが発生しているDockerfileのビルドログを目視

- Shipwrightのエラー分類に基づいて行う
- Dockerfile内の命令とビルドログの組み合わせを考慮

| 19件のビルドエラーパターンを特定

- 16.2% (764件) がDockerfile内の依存関係が原因でビルドに失敗

RQ2 (Dockerfile間の依存関係が原因のビルドエラーはどの程度発生するか)



RQ3 (継承の深さはビルドエラーに影響があるか)

| アプローチ

- Dockerオフィシャルイメージ, ユーザイメージを継承している Dockerfileに分類

| 結果

- オフィシャルイメージのDockerfileの深さとビルド失敗率

深さ	0	1	2	3	4	5	合計
件数 (件)	1,178	146	13	7	65	1	1,410
割合 (%)	83.5	10.3	0.9	0.4	4.6	0	100

- ユーザイメージのDockerfileの深さとビルド失敗率

深さ	0	1	2	3	4	5	合計
件数 (件)	223	44	19	8	1	1	286
割合 (%)	77.9	15.3	6.6	2.7	0.3	0.3	100

今後の予定

- | RQ4について分析を行う

- | 最終的なデータセット確定させたので反映する