

ビルドエラーを引き起こす Dockerfileの依存関係の分析

和歌山大学
オープンソースソフトウェア工学研究室
坂本 廉也

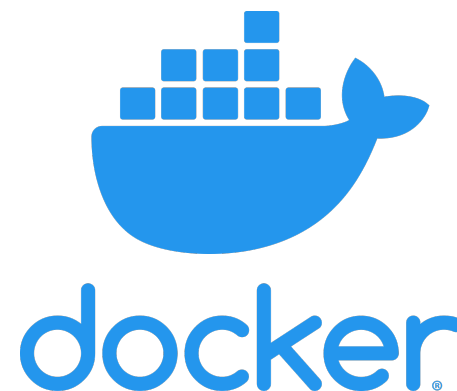
Dockerの普及

| Dockerとは

- コンテナ型仮想環境を提供するためのソフトウェア
 - 独立した仮想アプリケーション環境（ Dockerコンテナ）を動作
- 多くの企業が採用^[1]

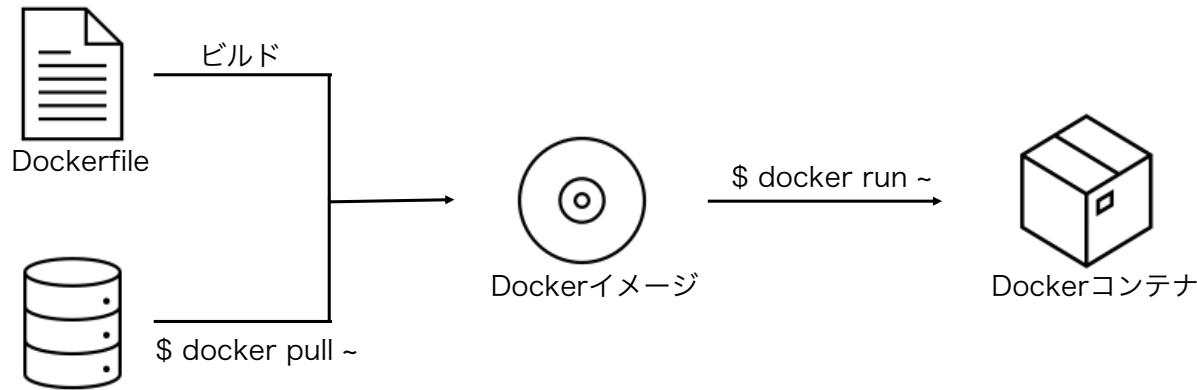
| なぜDocker？

- 環境構築が容易
 - 迅速な開発が可能に
- 環境の再現が容易
 - OSS開発では特に大きなメリット
 - 企業主導のOSS開発も多い



[1] <https://portworx.com/blog/2017-container-adoption-survey/>

Docker



| Dockerイメージ（以下、イメージ）

- コンテナ動作に必要な特定の環境のテンプレート
- Docker Hub^[?]から取得 or Dockerfileをビルド

| Dockerfile

- イメージに含める内容を指示するためのテキストファイル
- 1行に1命令を記述
 - 様々な命令が存在

Dockerfile

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

- FROM : ベースイメージの継承 (既存イメージの機能を使用)

| Dockerfileのビルド

- Dockerfile内の命令を上から順に実行
 - Dockerfile内の1命令が1つのレイヤを作成・実行結果を格納
- Dockerfileのビルドは失敗することがある (ビルドエラー)
 - ビルドエラー発生率は高く [2], その修正には多くの時間が必要 [3]

[2] 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).

[3] 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).

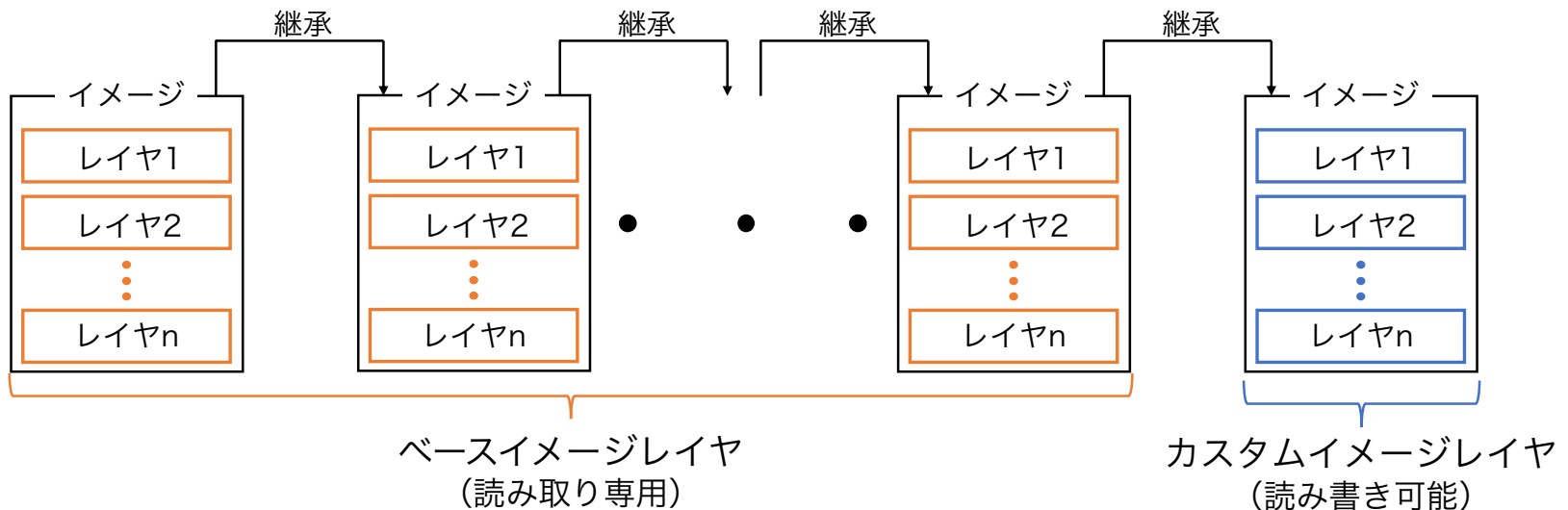
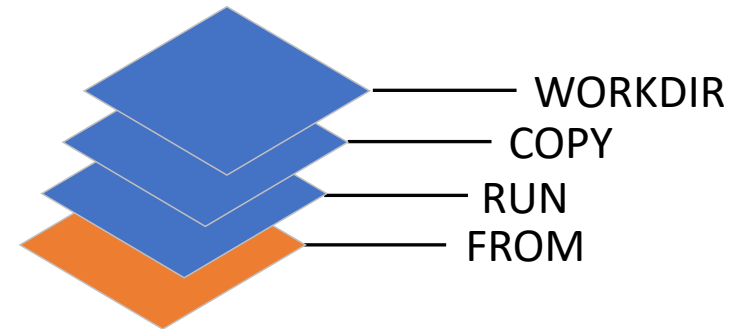
イメージのレイヤ構造

| イメージのレイヤ構造

– レイヤをスタックし，レイヤの集合としてイメージが成立

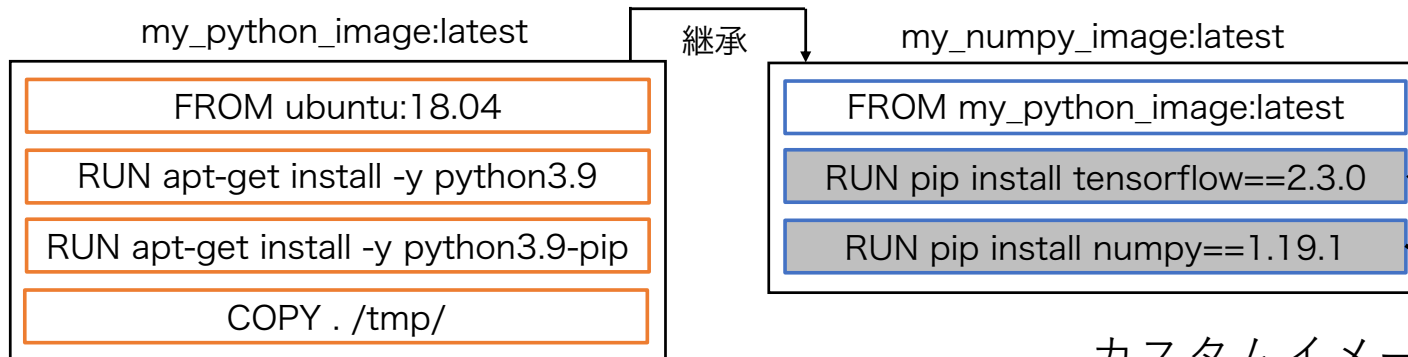
```
1 FROM ubuntu:20.04
2 RUN apt-get update && apt-get install python3
3 COPY . /app
4 WORKDIR /app
```

ビルド



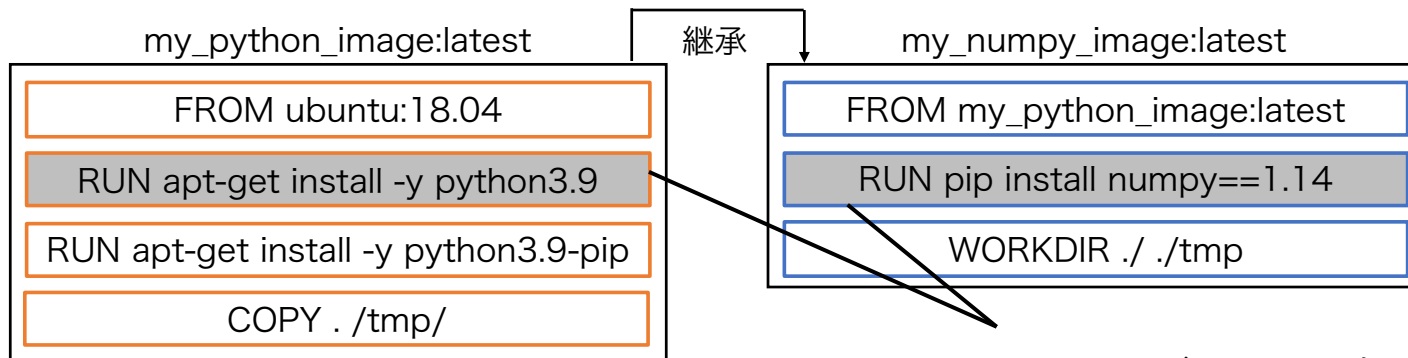
2つの依存関係とビルドエラー

レイヤレベルの依存関係が原因のビルドエラー



カスタムイメージレイヤ内の2つの命令の依存関係の互換性がない

イメージレベルの依存関係が原因のビルドエラー



ベースイメージレイヤ内のある命令と
カスタムイメージレイヤ内のある命令の互換性がない

ビルドエラーの自動修正

| Shipwright^[4]

- ビルドエラーを自動修正するためのツール（修正率：18.9%）
- エラーメッセージを分類し，分類結果を基に修正パッチを作成

Shipwrightの問題点

ビルドエラー以前に実行された命令との
依存関係を考慮した修正が行われていない

| 本研究の動機

- 依存関係の考慮が自動修正手法の性能向上につながる（RQ1）
- 複雑な依存関係が修正時間を引き延ばしている（RQ2）

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

目的 & Research Question

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

RQ1：レイヤレベルの依存関係とイメージレベルの依存関係が原因のビルトエラーはそれぞれどの程度存在するか

RQ2：レイヤレベルの依存関係とイメージレベルの依存関係が原因のビルドエラーの修正時間はそれぞれどの程度か

データセット

| Shipwrightのデータセットをフィルタリングして使用

– スター数10以上で単一のDockerfileを持つ, GitHubプロジェクト

1. 破損していたリポジトリ

– 正確な分析を行うことができない

2. 現在は存在していないリポジトリ

– 修正時間の分析を行うことができない

3. マルチステージビルドを利用したDockerfileを持つリポジトリ

– 本研究が定義する依存関係に当てはまらない

Dockerfile数	ビルドエラー件数	ビルドエラー率
16,099件	4,440件	27.5%

RQ1 | 2つの依存関係が原因のビルドエラー数

| 分析方法

- ビルドログの目視による分類

| 結果

ビルドエラーの原因	Dockerfile数	割合
レイヤレベルの依存関係	621件	14.0%
イメージレベルの依存関係	2,546件	57.3%
その他	1,273件	28.7%
合計	4,440件	100%

- ビルドエラーの主な原因

- ベースイメージレイヤとカスタムイメージレイヤの互換性

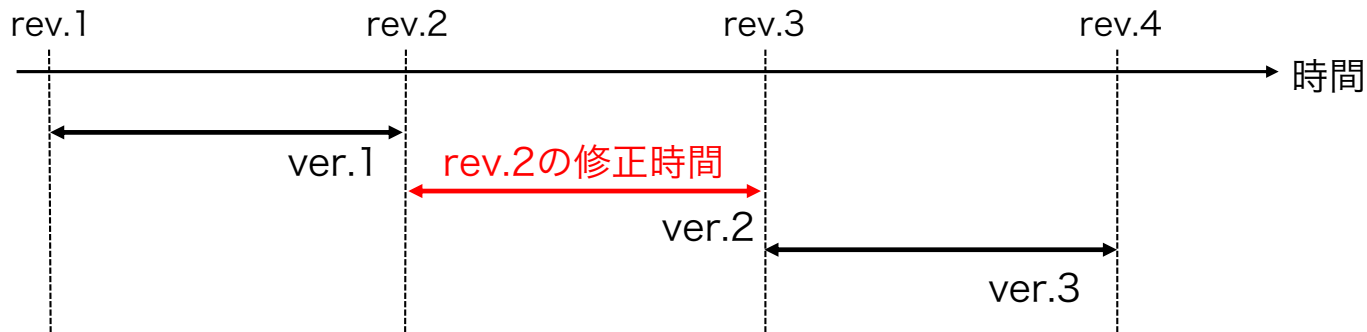
ベースイメージがビルドエラーに与える影響が大きい

RQ2 | 2つの依存関係の修正時間の差

| 分析方法

1. GitHub API v4でビルドエラーの修正時間の取得

- データセットのDockerfileと直後のDockerfileへのリビジョンを取得し、時間差を計算
- ビルドエラーの修正時間が12時間以上のDockerfileを除外^[5]



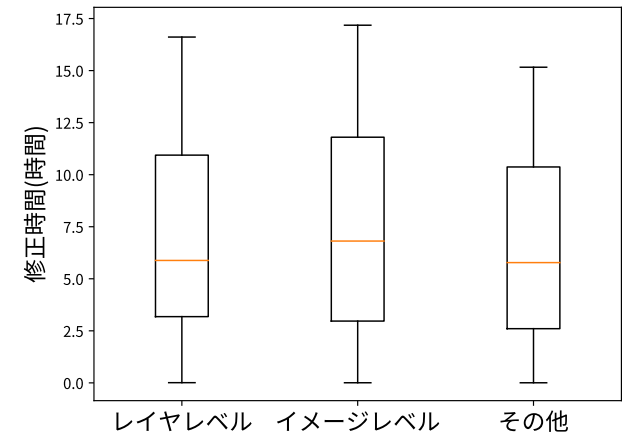
2. 2つの依存関係の修正時間の差について統計的検定

[5] Hyunmin Seo, Caitlin Sadowski, Sebastian Elbaum, Edward Aftandilian, and Robert Bowdidge. Programmers' build errors: A case study (at google). In Proceedings of the 36th International Conference on Software Engineering (ICSE '14), pp. 724–734

RQ2 | 2つの依存関係の修正時間の差

| 2つの依存関係によるビルドエラーの修正時間

- レイヤレベルの依存関係 : 6.9時間
- イメージレベルの依存関係 : 7.5時間
- その他 : 6.3時間



| 2つの依存関係の修正時間に差はある？

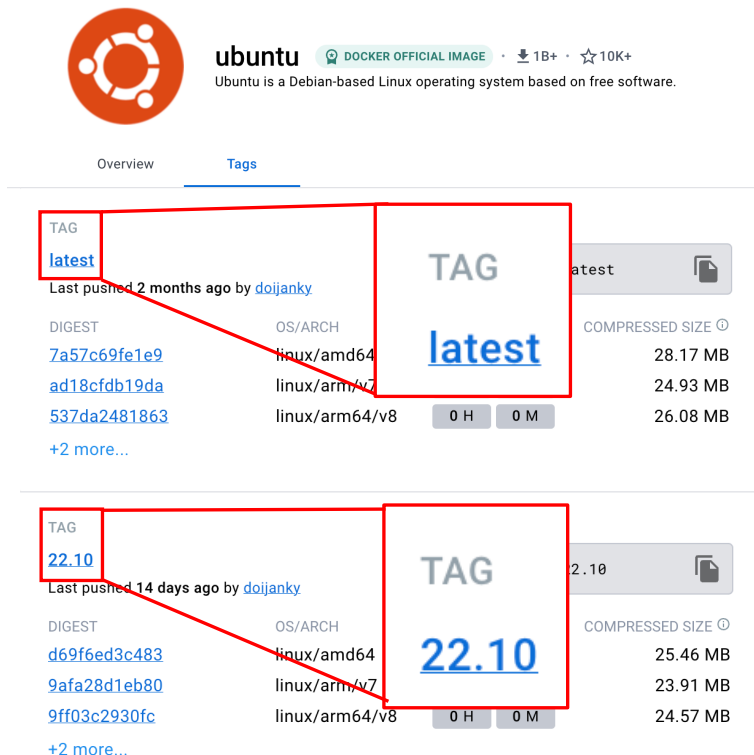
- マンホイットニーのU検定 (有意水準: 0.01)
- 結果: p 値=0.430

2つの依存関係の修正時間に差はない

考察1 | "latest"タグがビルドエラーに与える影響

ベースイメージ継承の際はタグも同時に指定

ー 主に、バージョンなどを管理するために使用される



“latest”タグで継承しない方がいい？

ー 意図しないバージョンでビルドが実行される可能性

“latest”タグはイメージレベルの依存関係を引き起こしやすいか

考察1 | "latest"タグがビルドエラーに与える影響

依存関係の種類	Dockerfile数	ビルドエラーが発生したDockerfile数	割合
レイヤレベル	621	43	6.90%
イメージレベル	2,546	275	10.80%

| 2つの依存関係で"latest"タグを用いて継承した場合のビルドエラー発生割合に違いはあるか

- カイ二乗検定（有意水準：0.01）
- 結果：p値 = 0.003

ベースイメージを"latest"タグで継承すると、
イメージレベルの依存関係によるビルドエラーを引き起こしやすい

考察2 | 自動修正手法性能向上のための修正対象

| ビルドエラーが発生しているが、その後修正（変更）されていないDockerfileが多く存在

－ 依存関係によるビルドエラーの修正はそもそも困難？

依存関係の種類	Dockerfile数	修正されていないDockerfile数	データセット全体に対する割合
レイヤレベル	621件	461件	22.4%
イメージレベル	2,546件	2,053件	46.2%



ベースイメージレイヤにまたがる依存関係の修正は困難

イメージレベルの依存関係が原因のビルドエラーの修正が必要

まとめ & 考えられること

- | ベースイメージがビルドエラーに与える影響が大きい
 - ベースイメージの選定の際に対応するDockerfileも確認すべき
 - "latest"タグでベースイメージを継承しない
- | イメージレベルの依存関係の修正は困難
 - ベースイメージレイヤを可視化することが大事
 - オフィシャルイメージはベースイメージレイヤが見える



ベースイメージの可視化



buildpack-deps:stable

DIGEST: sha256:604948951403ab9dbfe4f9147b15dc2860475b56295959e280b0e2eb1b7004cd

OS/ARCH

linux/amd64

COMPRESSED SIZE

307.32 MB

LAST PUSHED

4 days ago by [dojianky](#)

TYPE

Image

VULNERABILITIES

2 H

4 M

106 L

Image hierarchy

FROM	debian:11, 11.7, bullseye, bullseye-20230502, latest	🔗 !
FROM	buildpack-deps:bullseye-curl, curl, stable-curl	🔗 !
FROM	buildpack-deps:bullseye-scm, scm, stable-scm	🔗 !
ALL	buildpack-deps:stable	🔗 !

Layers (5)

0	ADD file:fc290cf8ddb984325474583faa79c5a98c5e...	55.05 MB	!
1	CMD ["bash"]	0 B	✓
2	/bin/sh -c set -eux; apt-get update; apt-get install -y -...	15.76 MB	!
3	/bin/sh -c apt-get update && apt-get install -y --no-in...	54.58 MB	!
4	/bin/sh -c set -ex; apt-get update; apt-get install -y -...	196.85 MB	!

Images (4)

Vulnerabilities (112)

Packages (543)

[Give feedback](#)

FROM debian:11

[🔗](#) DOCKER OFFICIAL IMAGE

Packages: 161

created 6 days ago

[View in Github](#)

FROM buildpack-deps:bullseye-curl

[🔗](#) DOCKER OFFICIAL IMAGE

Packages: 205

created 5 days ago

[View in Github](#)

FROM buildpack-deps:bullseye-scm

[🔗](#) DOCKER OFFICIAL IMAGE

Packages: 256

created 5 days ago

[View in Github](#)

buildpack-deps:stable

[🔗](#) DOCKER OFFICIAL IMAGE

Packages: 543

[View in Github](#)

大学院での研究について

機械学習を用いたDockerfileの ビルドエラー自動修正手法の提案

自動修正の必要性

| Shipwright^[4]

- ルールベース手法のビルドエラー自動修正ツール
 - エラーメッセージを分類し、分類結果を基に修正パッチを生成
 - 手作業で10個のパッチ作成ルールを作成

| Shipwrightの課題

- 手作業によるルール作成・メンテナンスが手間
- 実用的ではない修正
 - ベースイメージのバージョンダウン

目的 & アプローチ

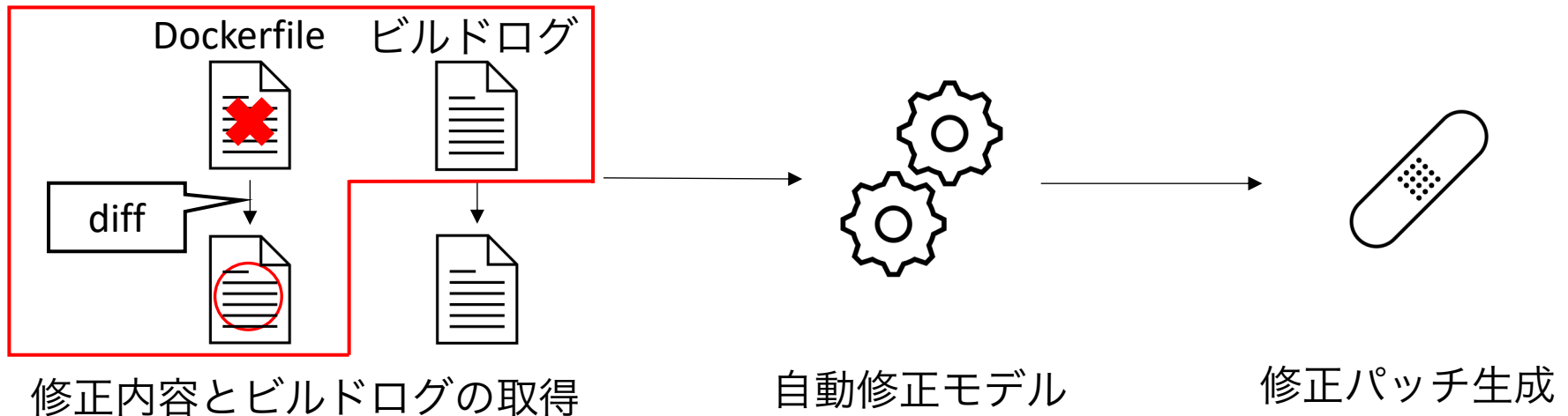
目的

Dockerfileへのパッチ生成, メンテナンスの手間削減
実用的な修正パッチの生成

| アプローチ

- 機械学習ベースで自動修正手法の構築
 - パッチ生成・メンテナンスの手間削減
- 実際の変更内容を学習
 - 実用的な修正パッチを生成

アプローチ



1. ビルドエラーが発生しているDockerfileの修正内容そのビルドログを取得
2. 機械学習を用いた自動修正モデルの構築
3. 生成された修正パッチをビルドエラーが発生しているDockerfileへ適用
4. パッチの評価

これからの予定

| 問題定義

| データ収集

- ビルドの成否が取得可能
- ビルドログが取得可能

補足資料

補足資料 | Dockerfileの命令

| FROM

- ベースイメージの継承

| RUN

- 任意のシェルコマンドを使用

| COPY

- ホストマシンのファイルをイメージへコピー

| WORKDIR

- 作業ディレクトリの指定