

Handling duplicates in Dockerfiles families : Learning from experts

Mohamed A. Oumaziz, Jean-Rémy Falleri, Xavier Blanc
University of Bordeaux
LaBRI, UMR 5800
F-33400, Talence, France
Email: {moumaziz, falleri, xblanc}@labri.fr

Tegawendé F. Bissyandé, Jacques Klein
University of Luxembourg
SnT Interdisciplinary Center
L-1855, Luxembourg
Email: {tegawende.bissyande, jacques.klein}@uni.lu

紹介者：福井

Docker

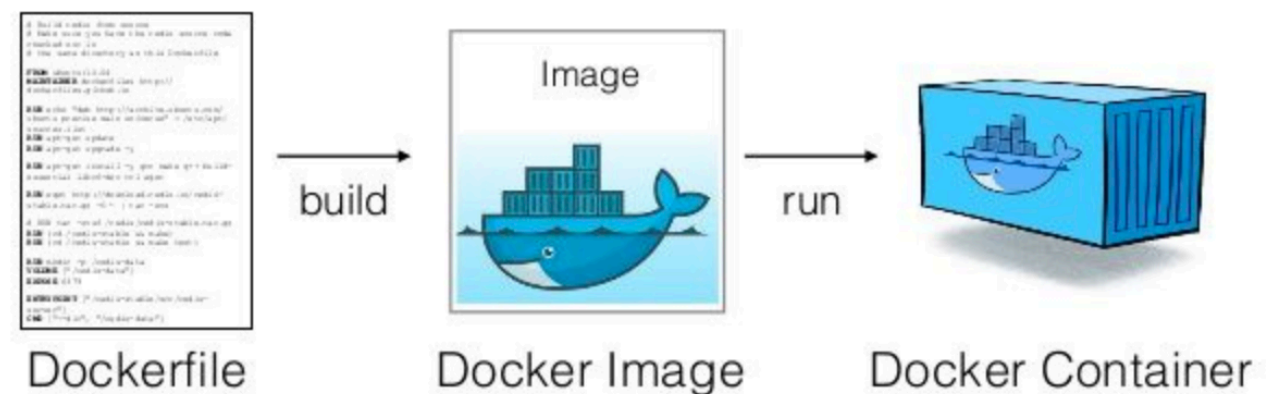
- Dockerはソフトウェアアプリケーションをデプロイおよび実行するために、開発者やエンドユーザーが使用する一般的なツールになりつつある
- Dockerの仕様
 - Dockerイメージと呼ばれるバイナリユニットにアプリケーションの依存関係をパッケージ化する
 - DockerイメージはDockerfileをもとに作成される

DockerfileとDockerイメージ

- Dockerfileはドメイン固有言語(DSL)で記述される
- その他のソースファイルと一緒にgitなどで管理されている
- DockerfileからDockerイメージを作成する

```
1  # Use the official python base image
2  FROM python:2.7-slim
3  # Set the working directory to /app
4  WORKDIR /app
5  # Copy current directory content into the
   container at /app
6  ADD . /app
7  # Install dependencies
8  RUN apt-get update && apt-get install -y
   apache2
9  # Make port 80 available to the world outside
   this container
10 EXPOSE 80
11 # Define environment variable
12 ENV NAME Hello World
13 # Run "python app.py" when the container is
   executed
14 CMD ["python", "app.py"]
```

Dockerfile



Dockerコンテナ構築の流れ

背景

- Dockerを使用するいくつかのソフトウェアプロジェクトは、複数の依存関係と復習のバージョンをサポートしている
 - 複数のDockerfile (DockerFamily:DF) を管理しているプロジェクトが存在する
 - 例：Python Projectは43個のDockerfileを含んでいる
- Dockerfileを対象とするパッチは、DockerFamilyの一部（または全て）に適用する必要がある

Dockerfileに適用するパッチの例

- Python3プロジェクトで検出されたパッチの例

Listing 1. A bugfix of the same duplicate across the Alpine family of Dockerfiles in the official Python project

```
1 RUN ...
2 && apk add --no-cache --virtual .build-deps \
3 bzip2-dev ... \
4 && make -j "$(nproc)"
    EXTRA_CFLAGS="-DTHREAD_STACK_SIZE=0x100000"
```

AlpineLinuxディストリビューションを使用すると起こる
セグメンテーションフォールトを修正するパッチ

Listing 2. An extract of the dicussed version number update across the whole Dockerfile family

```
1 ENV PYTHON_PIP_VERSION 19.0.2 19.0.3
```

新しいバージョンのpipを使用するためのパッチ

- DockerFamily間でパッチが伝播する
 - 影響するDockerfileに重複した命令が記述されているため起こる

課題と事例

- 課題

- パッチを安全に伝播するには，影響を受けるDockerfileを特定し，均等にファイル変更する必要がある
 - ▶ 時間がかかりエラーが発生しやすくなる

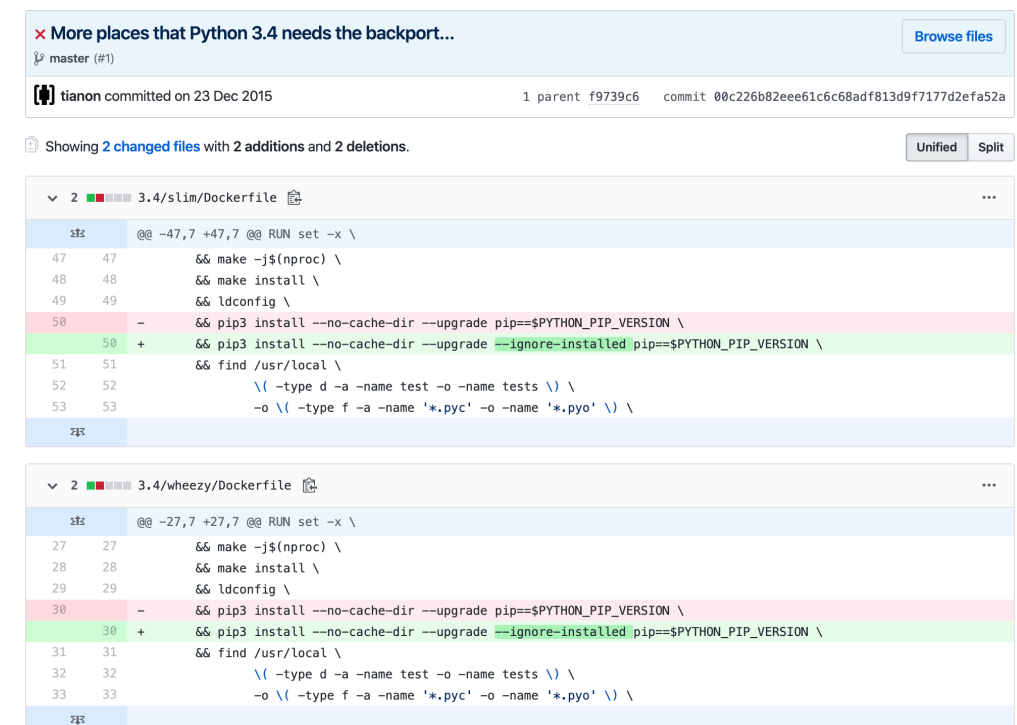
- 事例

- 2015年に適用されてた左のパッチは同日に別のファイルにも適用されている



```
Backport pip fix from 3.5 to 3.4
master (#1)
tianon committed on 23 Dec 2015
1 parent 4388fdb commit f9739c6da575c450aaed8628c1e0bfa97bf1ba18
Showing 1 changed file with 1 addition and 1 deletion.
3.4/Dockerfile
@@ -27,7 +27,7 @@ RUN set -x \
27 27 && make -j$(nproc) \
28 28 && make install \
29 29 && ldconfig \
30 - && pip3 install --no-cache-dir --upgrade pip==$PYTHON_PIP_VERSION \
30 + && pip3 install --no-cache-dir --upgrade --ignore-installed pip==$PYTHON_PIP_VERSION \
31 31 && find /usr/local \
32 32 \(\ -type d -a -name test -o -name tests \) \
33 33 -o \(\ -type f -a -name '*.pyc' -o -name '*.pyo' \) \
```

適用されたパッチ



```
More places that Python 3.4 needs the backport...
master (#1)
tianon committed on 23 Dec 2015
1 parent f9739c6 commit 00c226b82eee61c6c68adf813d9f7177d2efa52a
Showing 2 changed files with 2 additions and 2 deletions.
3.4/slim/Dockerfile
@@ -47,7 +47,7 @@ RUN set -x \
47 47 && make -j$(nproc) \
48 48 && make install \
49 49 && ldconfig \
50 - && pip3 install --no-cache-dir --upgrade pip==$PYTHON_PIP_VERSION \
50 + && pip3 install --no-cache-dir --upgrade --ignore-installed pip==$PYTHON_PIP_VERSION \
51 51 && find /usr/local \
52 52 \(\ -type d -a -name test -o -name tests \) \
53 53 -o \(\ -type f -a -name '*.pyc' -o -name '*.pyo' \) \
3.4/wheezy/Dockerfile
@@ -27,7 +27,7 @@ RUN set -x \
27 27 && make -j$(nproc) \
28 28 && make install \
29 29 && ldconfig \
30 - && pip3 install --no-cache-dir --upgrade pip==$PYTHON_PIP_VERSION \
30 + && pip3 install --no-cache-dir --upgrade --ignore-installed pip==$PYTHON_PIP_VERSION \
31 31 && find /usr/local \
32 32 \(\ -type d -a -name test -o -name tests \) \
33 33 -o \(\ -type f -a -name '*.pyc' -o -name '*.pyo' \) \
```

同日に適用されたパッチ

研究の貢献とRQ

- 貢献

- 公式Dockerプロジェクトを分析した結果を提供し、
DockerFamilyで起こる重複の問題を明らかにする

- リサーチクエスチョン

- 1.公式プロジェクトでDockerfileFamilyをメンテナンスしているか？また、なぜメンテナンスしているか？
- 2.DockerFamilyで重複が発生しているか？また、なぜ重複が起こるか？
- 3.公式プロジェクトは重複を管理するためにツールを利用しているか？また、ツールの長所と短所はなにか？

データ収集

- リポジトリ
 - 公式のDockerリポジトリを収集
- アンケート調査
 - コミットを分析してメールアドレスを収集
 - 900人にメールを送り, 29人から返答(9%)があり, 25人が複数のDockerfileを扱っている.
 - 25人のアンケート結果
- Dockerfileの重複項目の抽出
 - gitリポジトリを入力として, Dockerfileの重複項目を出力するツールの構築

リポジトリ

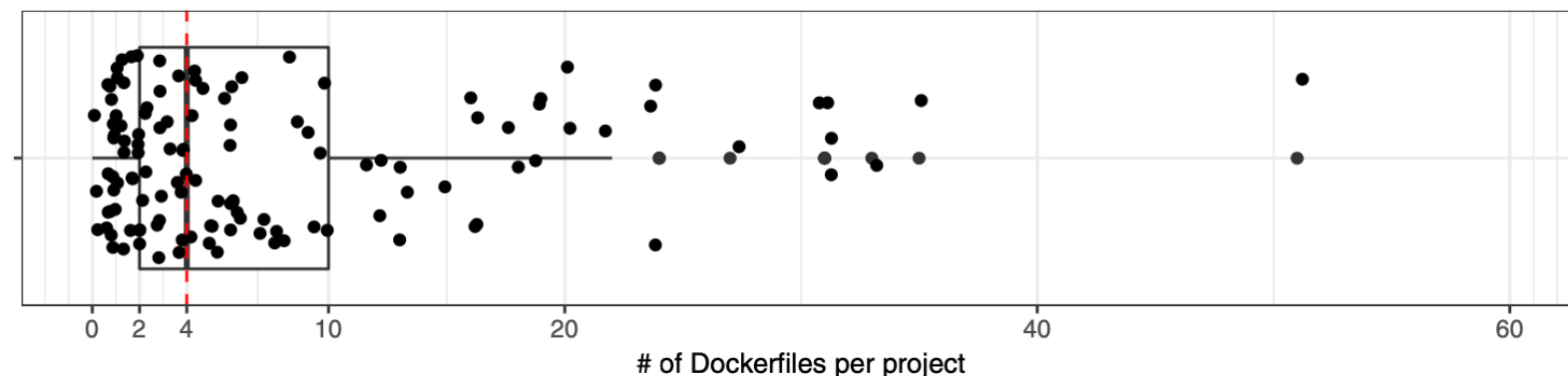
- 対象データ

- 公式Dockerプロジェクト：99プロジェクト
- Dockerfile：1,300ファイル

- フィルタリング

[公式のプロジェクト数：138プロジェクト]

- Dockerfileが1つのプロジェクトは削除
- 同じGithubリポジトリで管理されているプロジェクトは削除（InfluxDBとChronograf）



プロジェクトに含まれるDockerfileの数

アンケート調査

- アンケート項目

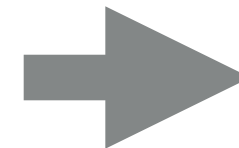
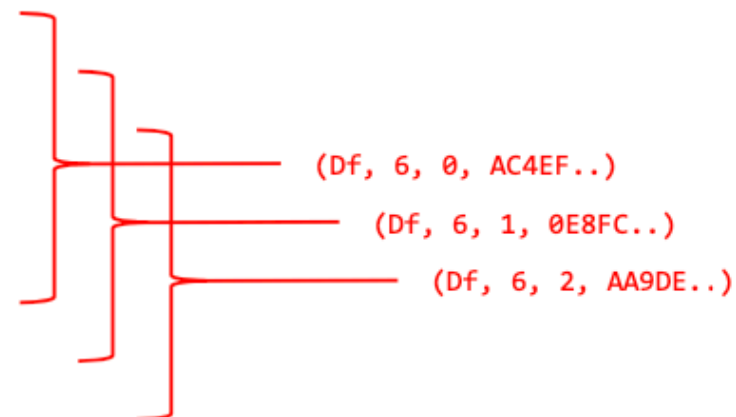
Questions

- How many Dockerfiles do you manage?
- This survey is for maintainers managing more than a single Dockerfile, do you still want to continue the survey?
- Do your Dockerfiles share duplicate instructions (see the following screenshot)?
- Have you ever updated several Dockerfiles in one commit to perform the same changes?
- How do you qualify such updates (the ones that perform several Dockerfiles)?
- You never updated several Dockerfiles in one commit. Can you explain why?
- Would you like to have a tool to detect the duplicate instructions?
- Why don't you want to have a tool to detect duplicate instructions?
- In your opinion, what should be the minimal size for a duplicate bloc of instructions to be detected?
- Across how many Dockerfiles should these instructions/blocs of instructions be duplicated to be detected?
- Do you have tools to avoid these duplicate instructions across your Dockerfiles?
- What tools do you use?
- Are you satisfied with your actual tool?
- Why don't you use a tool to avoid these duplicate instructions?
- Would you like to have a tool built to avoid duplicates in Dockerfiles?
- Why don't you want to have a specific tool to avoid these duplicates?
- Do you think the Dockerfile DSL (Domain Specific Language) should have reuse mechanisms?
- What should these mechanisms help you reuse?
- Why do you think the Dockerfile DSL shouldn't have reuse mechanisms?
- If you want to have feedbacks about the experiment, please leave your e-mail (will be deleted when the study is over)

Dockerfileの重複項目の抽出

- index-based duplicate detection technique [1] を用いて重複項目を特定
- 手法を適用するために、各命令をindexを付ける

```
FROM python:2.7-slim
WORKDIR /app
ADD . /app
RUN apt-get update
RUN apt-get install -y apache2
EXPOSE 80
ENV NAME Hello World
CMD ["python", "app.py"]
```



手法に適用

Figure 3. Dockerfile presenting an example of duplicate index with chunk size set to 6.

ChunkSizeを6にした例
(ChunkSizeは可変)

Dockerfileの重複項目の抽出

- 重複項目の抽出例
 - 1~3行目 (chunksize 3)
 - 5行目 (chunksize 1)

Listing 4. Extract of a Dockerfile duplicate (Bash shell v3.1)

```
1 FROM alpine:3.6
2 ENV _BASH_GPG_KEY 7C0135FB088AAF6 \
3 C66C650B9BB5869F064EA74AB
4 ENV _BASH_VERSION 3.1
5 ENV _BASH_PATCH_LEVEL 0
6 ENV _BASH_LATEST_PATCH 23
```

Listing 5. Extract of a Dockerfile duplicate (Bash shell v4.0)

```
1 FROM alpine:3.6
2 ENV _BASH_GPG_KEY 7C0135FB088AAF6 \
3 C66C650B9BB5869F064EA74AB
4 ENV _BASH_VERSION 4.0
5 ENV _BASH_PATCH_LEVEL 0
6 ENV _BASH_LATEST_PATCH 44
```

●

RQ1:調査方法

公式プロジェクトでDockerfileFamilyをメンテナンスしているか？また、なぜメンテナンスしているか？

- 調査方法

- Dockerfileのpathに着目して、Dockerfileの存在する目的をカテゴリ化する。最終的に4つにカテゴリー化

例：3.6/alpine/Dockerfile, 3.7/alpine/Dockerfileのpathを分割

- Level1 : 3.6, 3.7 softwareのverに対応
- Level2 : alpine baseイメージがAlpineLinuxを使用

RQ1:結果

Figure4 : UpsetPlot(Size:プロジェクト数)

- 最も重要なカテゴリーはVersion
- 2番目はとBaseImage
(どのDistributionを選択するか)
- 3番目はFlavor
 - 例 : Perlは normal or Multi Tread flavorプロジェクト)

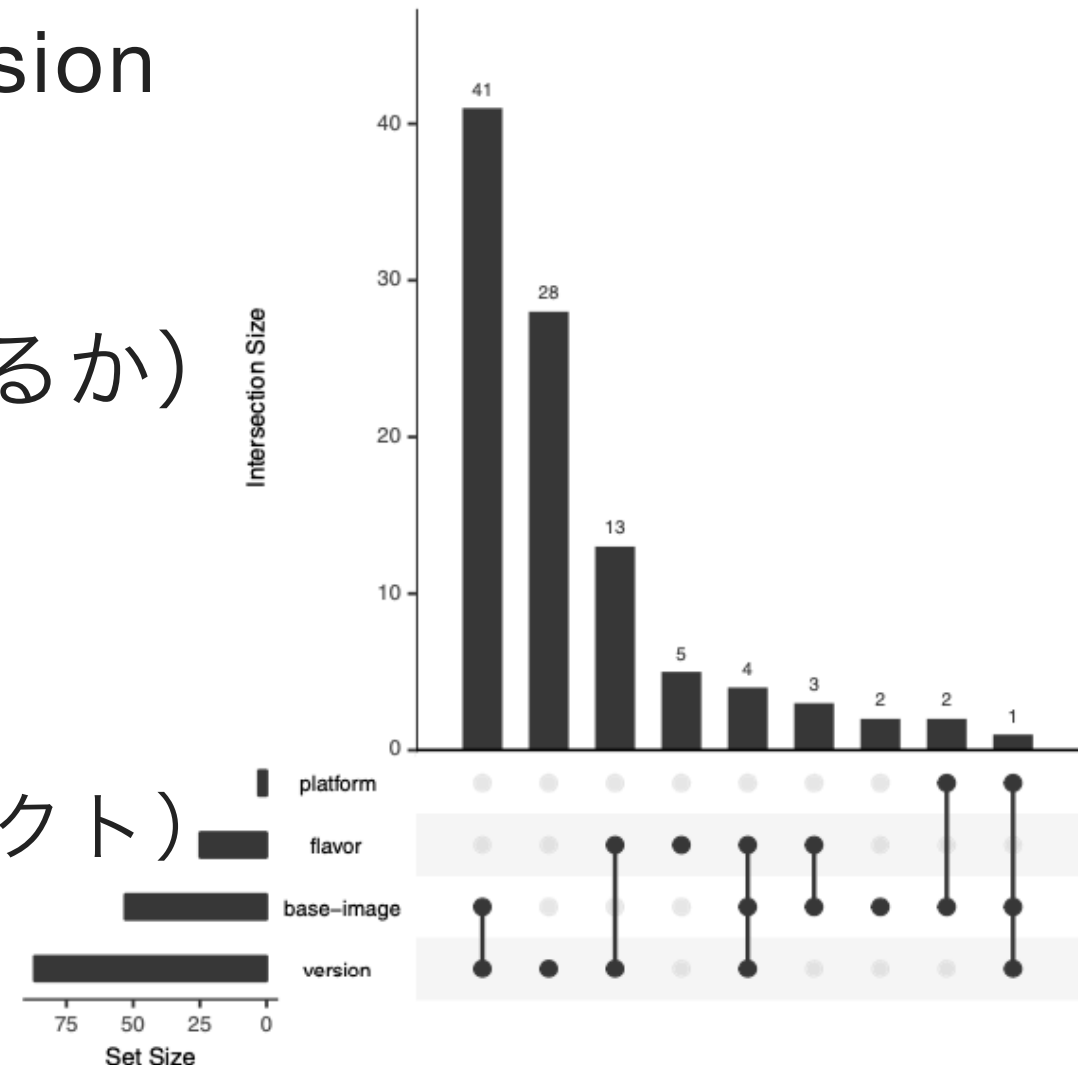


Figure 4. UpSet plot showing the relationships between versions, flavours, base images and platforms across our repositories.

Dockerfileをメンテする理由は, バージョン・ベースイメージ etcに
対応させるため

RQ2：調査方法

DockerFamilyで重複が発生しているか？

- 調査方法
 - 重複に関する統計量を調査
 - Dockerfileの重複した命令が，他のDockerfileにも影響を与えているかを調査
 - ・ Diggit Toolを同じ修正で2つ以上のDockerfileが変更されているコミットを自動的に調査

RQ2 : 結果

DockerFamilyで重複が発生しているか？

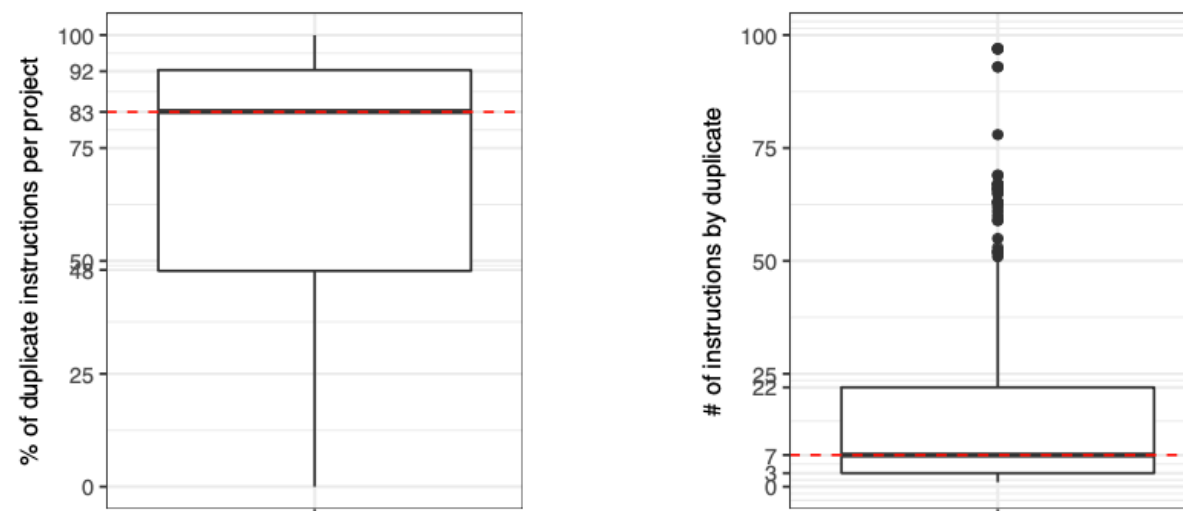


Figure 5. Left plot: Boxplot for the percentage of duplicate instructions per project. Right plot: Boxplot for the number of instructions by duplicate.

図左：各プロジェクトの重複の割合

図右：各プロジェクトの重複chunkの種類の数

多くのプロジェクトで重複したDockerfileが存在する

またメンテナーへのアンケート調査では、68%が重複に
直面したことがあると回答した

RQ2 : 結果

DockerFamilyで重複が発生しているか？

- Dockerfileが同時に編集されるかの調査

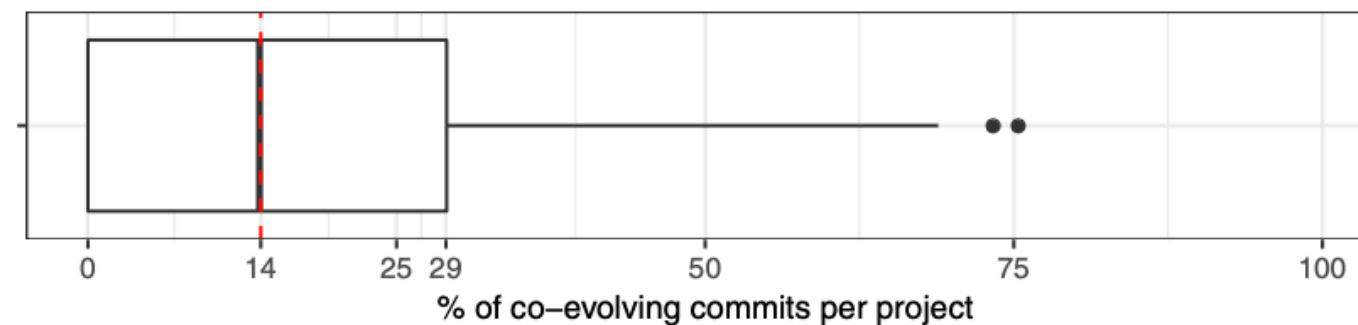


Figure 7. Boxplot for the percentage of co-evolving commits per project.

14%（中央値）のコミットで同時にDockerfileが編集されている

RQ2:なぜ重複が起こるかの調査

なぜ重複が起こるか？

- 調査方法
 - 重複したデータから，50の重複をランダムサンプリング
 - 筆者ら（3人）が，重複を目視して対応する理由と考えられるものを3つに分類

RQ2 : 結果

Listing 6. Duplicate due to identical software installation process

```
1 ENV XWIKI_VERSION=8.4.5
2 ENV XWIKI_URL_PREFIX "http://maven.xwiki.org
  /.../${XWIKI_VERSION}"
3 ENV XWIKI_DOWNLOAD_SHA256 52ed122c...
4 RUN rm -rf /usr/local/tomcat/webapps/* && \
5 mkdir -p /usr/local/xwiki/data && \
6 curl -fSL "${XWIKI_URL_PREFIX}/xwiki-ent-web-${
  XWIKI_VERSION}.war" -o xwiki.war \&& \
7 echo "${XWIKI_DOWNLOAD_SHA256} xwiki.war" |
  sha256sum -c - && \
8 unzip -d /usr/local/tomcat/webapps/ROOT xwiki.
  war && \
9 rm -f xwiki.war
```

ソフトウェアのインストールと環境設定

- 1~4行目 : 環境変数の設定
- 5~7行目 : Xwiki実行のための設定

Listing 7. Duplicate due to dependency manager installation

```
1 RUN apt-get update
2 RUN apt-get install -y mysql-client-core-5.7
  openjdk-8-jre-headless postgresql-client
3 RUN rm -rf /var/lib/apt/lists/*
```

依存関係のインストール

- パッケージマネージャーを構成して、依存関係をインストールする必要がある（たいてい同じ処理となる）

Listing 8. Duplicate due to runtime configuration

```
1 VOLUME /spiped
2 WORKDIR /spiped
3 COPY *.sh /usr/local/bin/
4 ENTRYPOINT ["docker-entrypoint.sh"]
5 CMD ["spiped"]
```

実行手順の設定

- ENDPOINT命令によりコマンドを実行する

RQ3：調査方法

公式プロジェクトは重複を管理するためにツールを利用しているか？

- 調査方法
 - 手動で99個のリポジトリにDockerfile管理ツールが含まれているかを調べる
 - アンケート調査を行う

RQ3 : 結果

- 66%のプロジェクトがツールを使用している
 - 通常はupdate.shとしてルートに配置されている
 - ツールに種類が存在する
 - 75%がコマンドライン上で対象fileなどを選択する形式
 - 25%がWebサイトから自動で取得（一部はbotを用いて定期的に更新）
- アンケート結果
 - 56%(14/25)がツールを使用していないと回答
 - 57%(8/14)がツールがほしいと回答
 - 63%(7/11)が使用していて、ツールに満足していると回答

RQ3 : 結果

- ツールの種類
 - 検索と置換
 - ・ 書き換える箇所を指定して、まとめて置換する
 - ・ バージョン更新の伝播をうまく処理する。バージョン更新以外はできない
 - テンプレート
 - ・ 関連する一連の変更をテンプレートに記述して、Dockerfileを再生成する
 - ・ テンプレートでは一部は手動で変更する必要がある
 - ジェネレーター
 - ・ テンプレートを用いずに、Dockerfileを再生成するためのソースを書く
 - ・ ソースが複雑になる

まとめ

- 貢献

- Dockerfileの重複に関する知見を提供
 - ・ 公式のリポジトリはどのように保守しているか

- 今後

- より広いニーズを特定するために、幅広いメンテナーコミュニティによって維持されるプロジェクトに調査を拡大
- メンテナーがすぐに使用できる管理ツールの構築