Docker セミナー 2023

Docker によるコンテナ利用入門

渡場　康弘

福井大学 工学系部門工学領域 情報・メディア工学講座 准教授

大阪大学サイバーメディアセンター 招へい教員

コンテンツ

1. コンテナ技術概要

2. Docker 概要

3. Dockerの基本的な利用方法

4. Squid におけるコンテナ利用

コンテンツ

1. コンテナ技術概要

2. Docker 概要

3. Dockerの基本的な利用方法

4. Squid におけるコンテナ利用

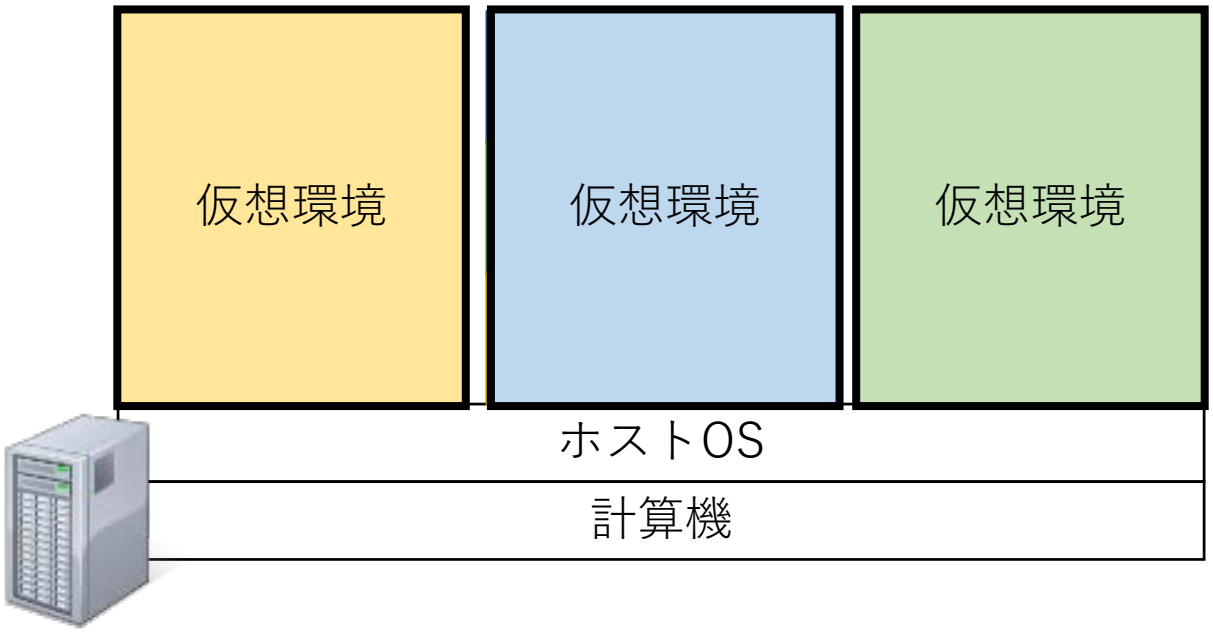
計算環境への要求の変化

- 計算機利用の多様化
  - 従来：自作プログラムによる利用
    => OSS のような公開プログラムの活用の増加
    - ビッグデータ解析、AI関係のライブラリ、フレームワーク等において開発が活発
- 公開プログラム活用における問題
  - デバイスドライバやライブラリの特定のバージョンを必要とする場合がある
  - スパコンのような高性能計算機環境はいわゆる「枯れた技術」で構成
    => 最新のライブラリ等が使用できない場合あり

利用するソフトウェアの要件を満たした環境の構築が必要

仮想化技術の活用

- 仮想化技術の利点
  - 単一環境で異なるソフトウェア環境を実行可能
  - マルチコア・大容量メモリのようなリソースの有効活用



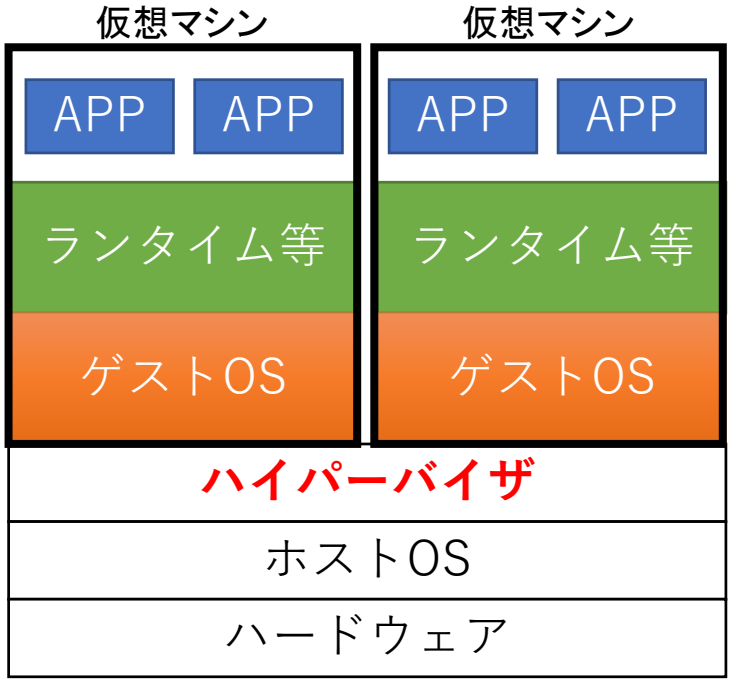
コンテナ

- コンテナとは
  - 隔離された実行環境を構築するための技術
    - 隔離されたプロセスとして実行
  - アプリケーション実行に必要な要素だけをパッケージ化

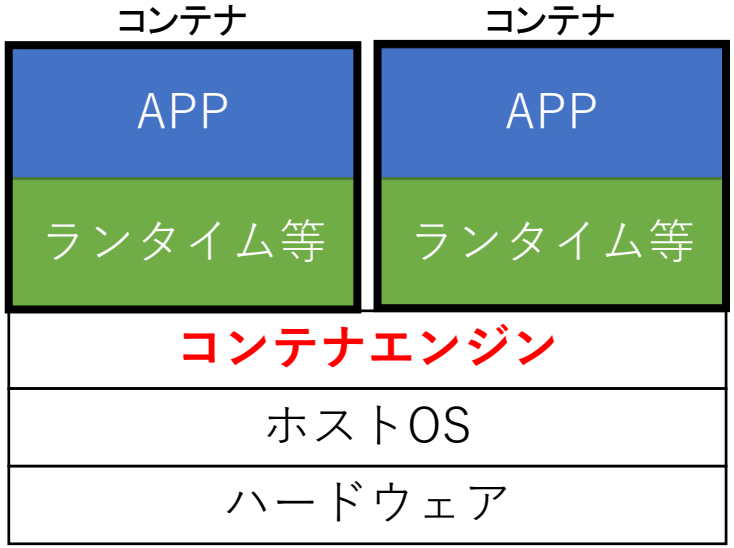
- コンテナの仕組み
  - Linuxカーネルの機能を利用して実現
    - namespaces:
      他のnamespace のプロセスからリソースを隔離
    - cgroups:
      アクセス可能なリソースを制限

仮想マシンとコンテナ

- 主な違いはゲストOSの有無
  - コンテナ：ホストOSの機能（カーネル）を利用
    => 軽量な環境構成



仮想マシン（ホストOS型）

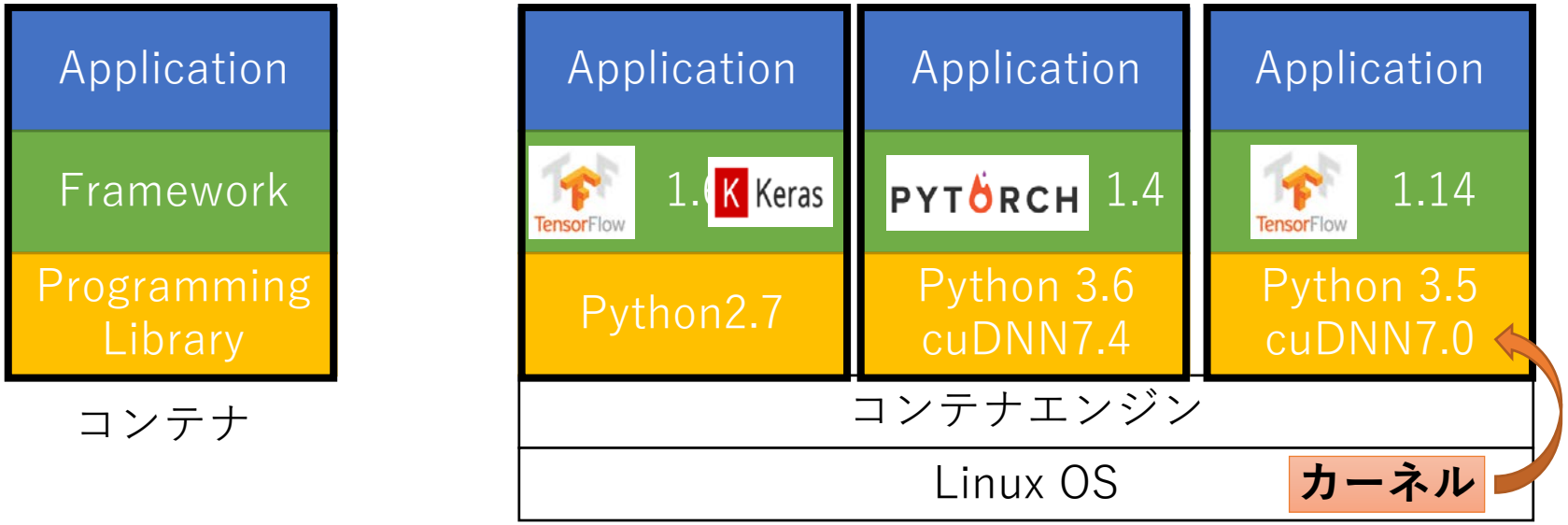


コンテナ

コンテナ型仮想化技術の活用

- コンテナ利用の利点
  - 動作が軽量
  - ポータビリティが高い
  - 実行可能検証済みのソフトウェア環境を利用可能

=> ユーザは適切な環境構成を手間をかけずに利用可能



コンテンツ

1. コンテナ技術概要

2. Docker 概要

3. Dockerの基本的な利用方法

4. Squid におけるコンテナ利用

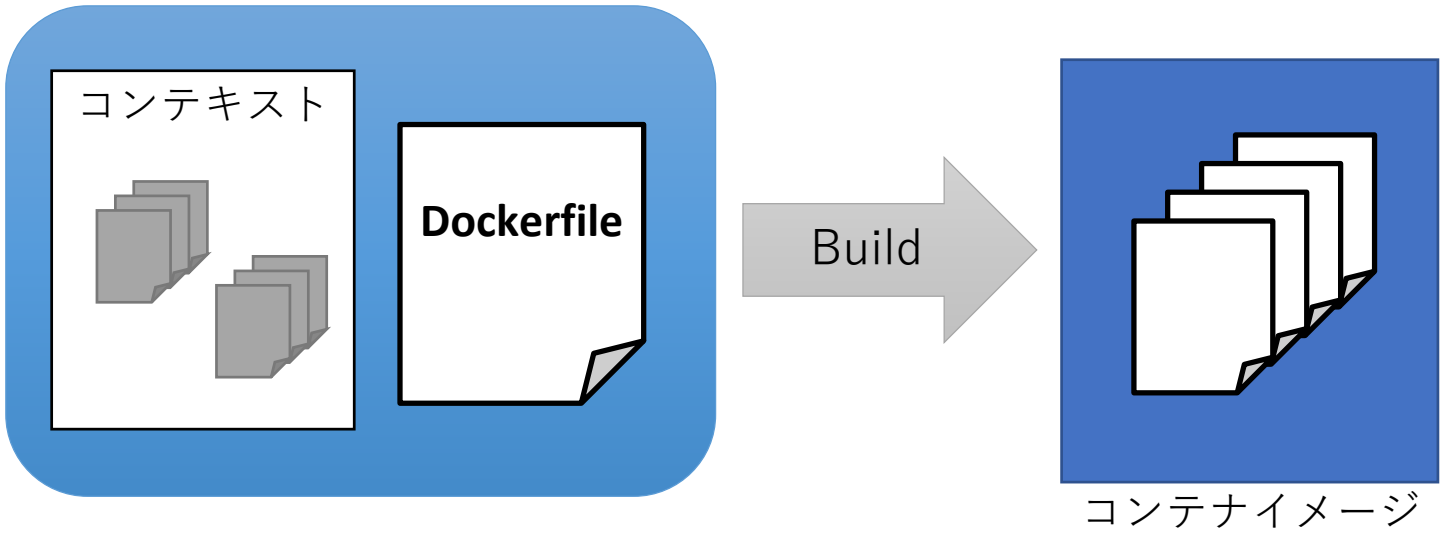
Docker とは

- コンテナ型仮想環境の作成・配布・実行を行うためのプラットフォーム
  - コンテナの操作を容易に

- Dockerにおける全体的な流れ
  - Build：コンテナのイメージを作成
  - Ship：コンテナの共有（配布）
  - Run：コンテナの実行

Build

- コンテキストとDockerfileからコンテナイメージを作成
  - コンテキスト：
    アプリケーションに必要なファイル群
  - Dockerfile：
    コンテナイメージの設計書（テキストファイル）



Dockerfile の例

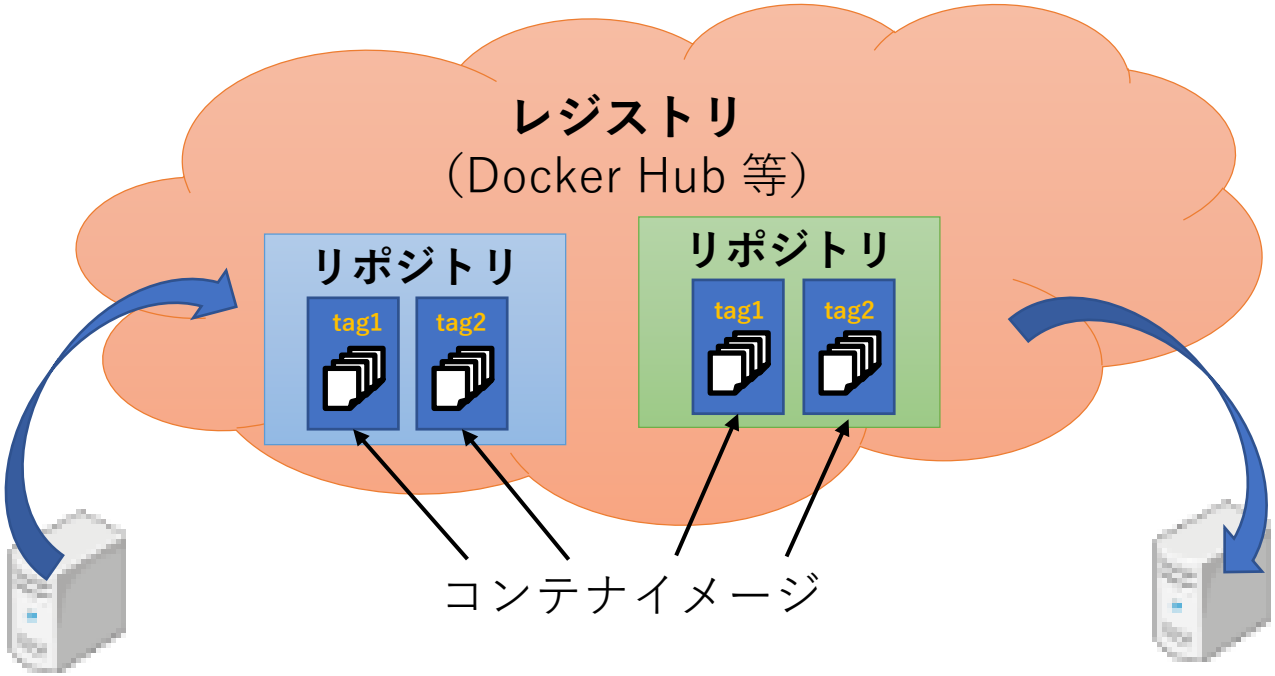
```
FROM centos:7
RUN yum -y update
COPY ./readme.txt /readme.txt
CMD ["echo","Welcome Docker"]
```

- 主な命令
  - FROM：ベースとなるイメージの指定
  - RUN：コマンドの実行
  - COPY：ファイルを追加
  - ADD：ファイルを追加（tar は自動的に展開）
  - CMD：コンテナ生成時に自動的に実行
  - ENTRYPOINT：基本的にCMD と同様
    コマンドの上書き方法が異なる

Ship

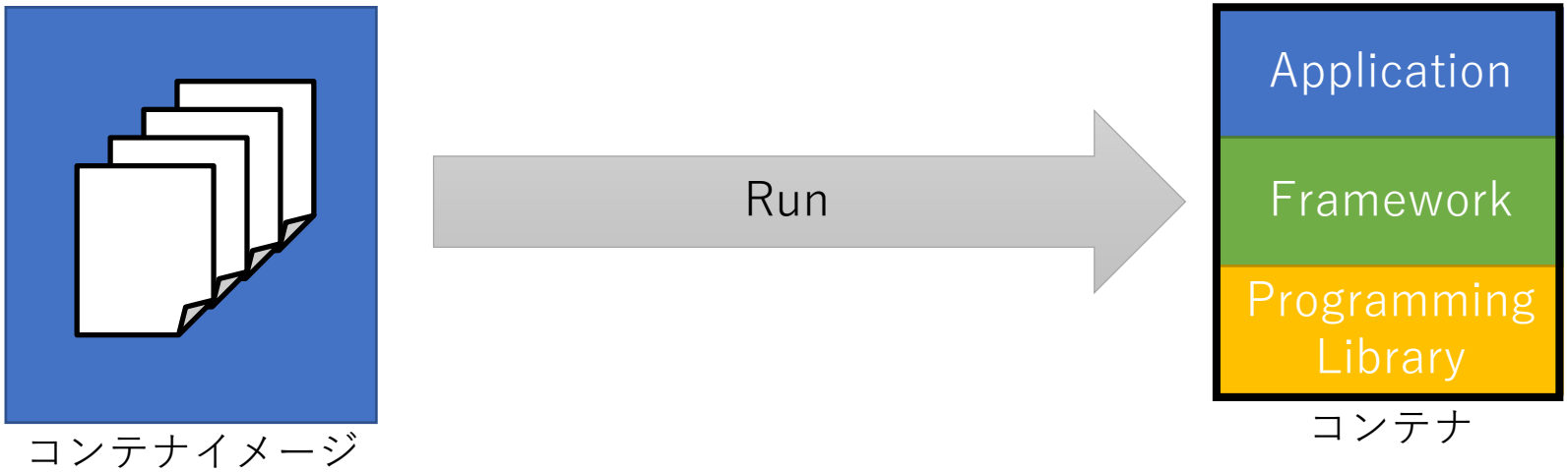
コンテナイメージの共有

- レジストリ：コンテナイメージを保存するサービス
- リポジトリ：同じ名前のイメージの集合
  タグによりイメージを識別



Run

取得したイメージからコンテナを生成

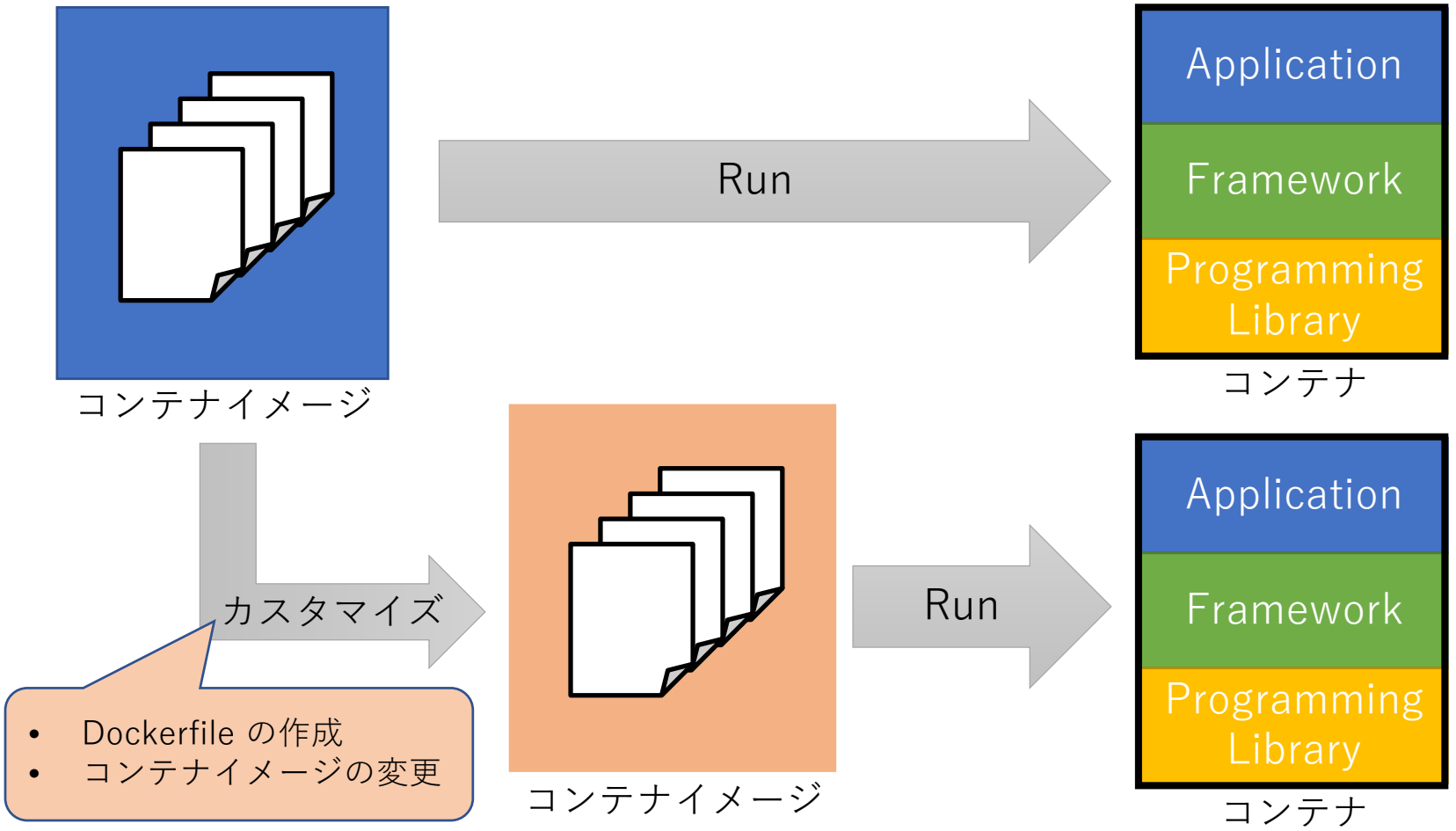


コンテナイメージ

コンテナ

Run

取得したイメージからコンテナを生成

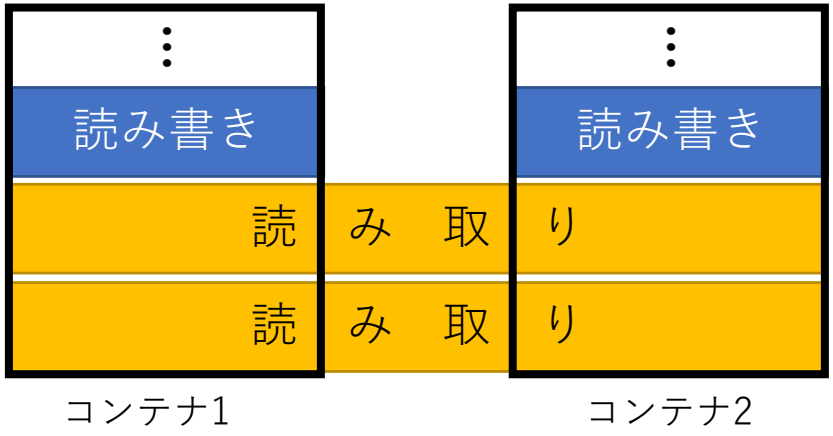


コンテナイメージのレイヤ構造

- Dockerfile内の記述による変更差分（RUN、ADD等）をレイヤとして格納
  => 再度イメージのBuild を行った際には
  更新された変更部分だけが実行

コンテナ内のファイルの扱い

- イメージに含まれているレイヤ(ファイル)は読み取り専用
  - 変更はコンテナに追加される「読み書き可能レイヤ」で反映
- 同じイメージから生成されたコンテナは読み取りレイヤを共有



コンテンツ

1. コンテナ技術概要

2. Docker 概要

3. Dockerの基本的な利用方法

4. Squid におけるコンテナ利用

Dockerコマンド

- コマンド形式：
  docker <command> [option] ...

- Dockerイメージの操作
  - docker pull [options] <name>[:tag] ：
    イメージのダウンロード
  - docker push [options] <name>[:tag] ：
    イメージのアップロード
  - docker images [option] [repository[:tag]]:
    イメージの一覧表示
  - docker search [options] <name> ：
    Docker Hub のイメージを検索

Dockerコマンド

- Dockerコンテナの実行
  - docker run [options] <image> [command] [arg...] :
    コンテナを起動状態で作成
    - オプション
      - -i :ホストの入力をコンテナの標準出力と繋げる
      - -t :コンテナの標準出力とホストの出力を繋げる
      - -d :バックグラウンドで動作
      - --name :コンテナに名前を付ける
      - -p :コンテナのポートをホストに公開
  - docker create [options] <image> [command] [arg...] :
    コンテナを停止状態で作成
  - その他
    - start(起動)、stop(停止)、kill(強制停止)、restart(再起動)

Dockerコマンド

- Dockerコンテナの操作
  - docker ps :
    起動中のコンテナー覧(-a で停止中も表示)

```
[root@localhost ~]# docker ps
CONTAINER ID   IMAGE     COMMAND     CREATED          STATUS          PORTS     NAMES
dec18c0362aa   alpine    "/bin/sh"   5 minutes ago    Up 5 minutes              boring_antonelli
[root@localhost ~]#
```

  - docker stats [container id] :
    コンテナの稼働確認

  - docker rm [options] [container id] :
    コンテナの削除
    - -f :強制削除

Dockerコマンド

- Dockerイメージの作成・削除
  - docker build –t <repository[:tag]> <path> :
    Dockerfile の場所を指定
    （カレントディレクトリにあるなら<path>は<path> ”.”）
  - docker build –t <repository[:tag]> –f <Dockerfile> <path> :
    Dockerfile 名を指定する場合
  - docker build – < <Dockerfile> :
    標準入力から作成
  - docker rmi <image id> :
    Dockerイメージの削除
  - docker tag <source_image>[:tag] <target_image>[:tag] :
    対象イメージに元イメージを参照するタグを作成

Dockerコマンド

- Dockerイメージの作成・削除
  - docker commit [options] <container id> [name]>[:tag]] : 現時点でのコンテナの状態からイメージ作成
  - docker export :コンテナをtar に保存
  - docker import :tar からイメージを生成
  - docker save :イメージを保存
  - docker load :イメージの読み込み

Docker利用の基本的な流れ

1. （docker search でイメージを検索）

2. docker pull でイメージを取得

3. docker run で実行

4. （イメージをカスタマイズ）
   - Dockerfile の変更、docker commit 等

コンテンツ

1. コンテナ技術概要

2. Docker 概要

3. Dockerの基本的な利用方法

4. Squid におけるコンテナ利用

利用可能なコンテナ技術

- Dockerと Singularity が導入
  - 標準サービスはSingularity

	Docker	Singularity
バージョン	20.10.3-3	3.7-2
利用方法	特殊なコンテナ起動(標準利用不可)	フロントエンド上で利用可能
利用可能イメージ	管理者による事前登録が必要	利用者自身で持ち込みと実行が可能

大阪大学サイバーメディアセンター2021年度講習会「コンテナ入門」資料より

Singularity

- Singularity とは
  - HPC環境向けのコンテナ技術
    - HPC系ソフトウェアスタックが利用可能
  - 公式レジストリ：https://cloud.sylabs.io/library

- 特徴
  - ユーザコマンドによりコンテナエンジンを起動可能
    ⇔ Dockerのコンテナエンジンはサービスとして常駐
  - ユーザ権限でのコンテナ起動
    ⇔ Docker ではroot 権限でコンテナが起動
    <= サービスとしては提供が困難
  - Dockerイメージの流用が可能

Singularity の利用方法

詳細は下記のページ/資料を参照

- https://sylabs.io/singularity/

- http://www.hpc.cmc.osaka-u.ac.jp/system/manual/squid-use/singularity/

大阪大学サイバーメディアセンター大規模計算機システムのトップページ より
[システム] => [利用方法] => [SQUIDの利用方法] => [コンテナの利用方法]

- サイバーメディアセンター2022年度講習会「コンテナ入門」 資料

SQUID でのユーザ作成Docker イメージの利用

- Docker Hub に登録したイメージを利用
  1. 構築したDockerイメージをDocker Hub に登録
  2. Singularity コマンドでDocker Hub からイメージ取得
     $ singularity build [イメージファイル名].sif docker://[イメージ情報]

- ローカルにSingularity を導入して変換
  1. Singularity コマンドでローカルレジストリから変換
     $ sudo singularity build [イメージファイル名].sif docker-daemon://[イメージ情報]
  2. 作成したsif ファイルをSQUID にファイル転送

まとめ

1. コンテナ技術概要

2. Docker 概要

3. Dockerの基本的な利用方法

4. Squid におけるコンテナ利用