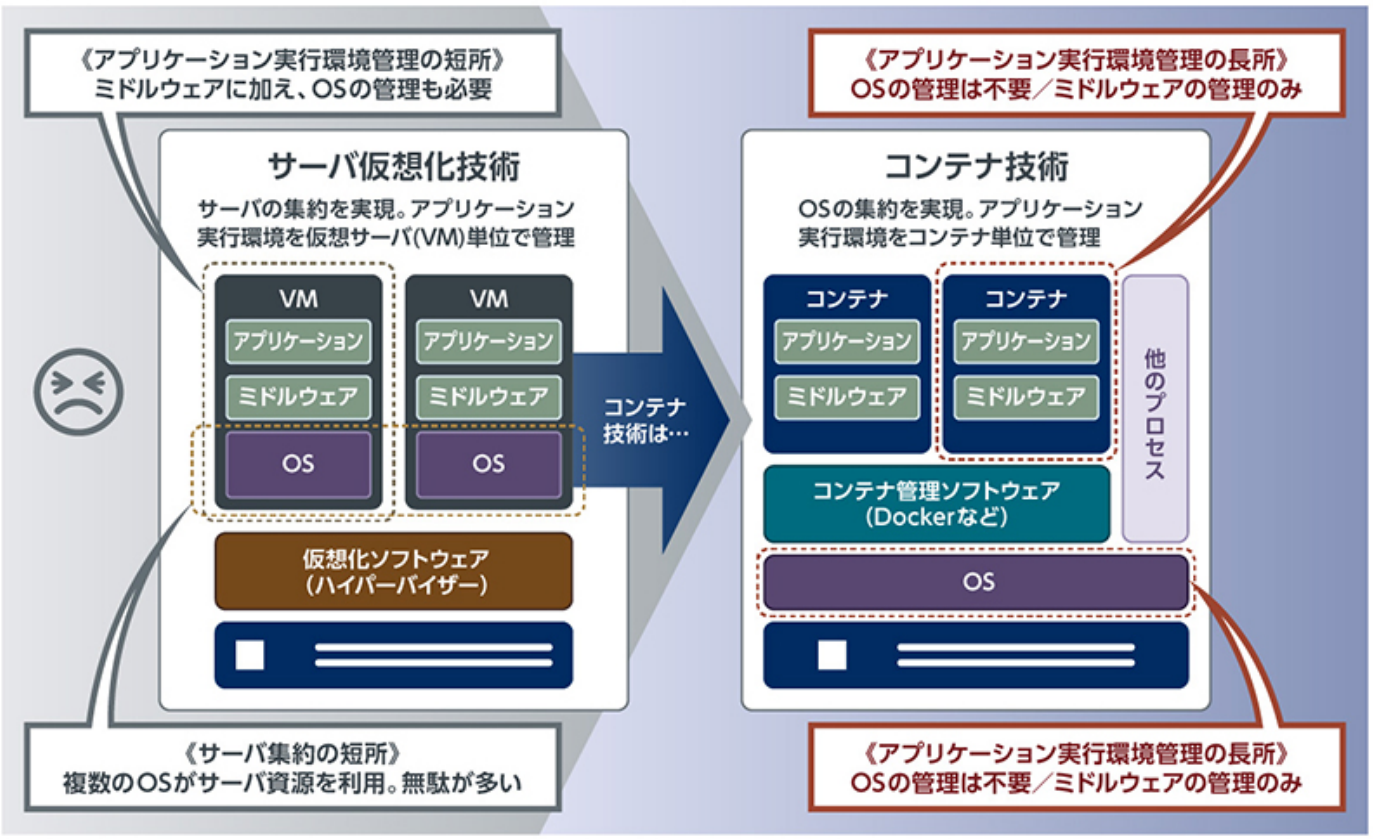
2023-08-22 勉強会資料

コンテナ仮想

コンテナ仮想とは

　コンテナとは、ホストマシンから隔離された仮想空間で実行されるプロセスです。Dockerはこの仮想プロセスを実行するためのコンテナエンジンです。サーバー仮想化と異なり、OS管理が不要であること、コンテナはホストマシンとリソースを共有（カーネルパラメータなど）をしており、1プロセスを1コンテナとして実行するため軽量であることが特徴です。



　コンテナ仮想には２種類あり、１つは `システムコンテナ` と呼ばれもう一つが `アプリケーションコンテナ` と呼ばれます。システムコンテナは、アプリケーションと実行環境ごとコンテナ化したもので従来の仮想マシンに近いものです。代表的なシステムコンテナツールにはLXDがあります。一方、アプリケーションのみをコンテナ化したものはアプリケーションコンテナ[1]と呼ばれ、これは普段我々が利用しているDockerによるコンテナ仮想です。

　本稿では、特に注釈がない限りコンテナはアプリケーションコンテナを指します。

dockerコマンド

dockerには、多種多様なコマンドが用意されています。docker helpでどのようなコマンドがあるか確認しておきます。

```
1
2  Usage:  docker [OPTIONS] COMMAND
3
4  A self-sufficient runtime for containers
5
6  Options:
7        --config string      Location of client config files (default
8                             "/Users/kpu0471/.docker")
9    -c, --context string     Name of the context to use to connect to the daemon
10                            (overrides DOCKER_HOST env var and default context set
11                            with "docker context use")
12   -D, --debug              Enable debug mode
13   -H, --host list          Daemon socket(s) to connect to
14   -l, --log-level string   Set the logging level
15                            ("debug"|"info"|"warn"|"error"|"fatal") (default "info")
16       --tls                Use TLS; implied by --tlsverify
17       --tlscacert string   Trust certs signed only by this CA (default
18                            "/Users/kpu0471/.docker/ca.pem")
19       --tlscert string     Path to TLS certificate file (default
20                            "/Users/kpu0471/.docker/cert.pem")
21       --tlskey string      Path to TLS key file (default
22                            "/Users/kpu0471/.docker/key.pem")
23       --tlsverify          Use TLS and verify the remote
24   -v, --version            Print version information and quit
25
26 Management Commands:
27   builder     Manage builds
28   buildx*     Docker Buildx (Docker Inc., v0.8.2)
29   compose*    Docker Compose (Docker Inc., v2.6.1)
30   config      Manage Docker configs
31   container   Manage containers
32   context     Manage contexts
33   extension*  Manages Docker extensions (Docker Inc., v0.2.7)
34   image       Manage images
35   manifest    Manage Docker image manifests and manifest lists
36   network     Manage networks
37   node        Manage Swarm nodes
38   plugin      Manage plugins
39   sbom*       View the packaged-based Software Bill Of Materials (SBOM) for an image (Anchore
    Inc., 0.6.0)
40   scan*       Docker Scan (Docker Inc., v0.17.0)
```

```
  secret      Manage Docker secrets
  service     Manage services
  stack       Manage Docker stacks
  swarm       Manage Swarm
  system      Manage Docker
  trust       Manage trust on Docker images
  volume      Manage volumes

Commands:
  attach      Attach local standard input, output, and error streams to a running container
  build       Build an image from a Dockerfile
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local filesystem
  create      Create a new container
  diff        Inspect changes to files or directories on a container's filesystem
  events      Get real time events from the server
  exec        Run a command in a running container
  export      Export a container's filesystem as a tar archive
  history     Show the history of an image
  images      List images
  import      Import the contents from a tarball to create a filesystem image
  info        Display system-wide information
  inspect     Return low-level information on Docker objects
  kill        Kill one or more running containers
  load        Load an image from a tar archive or STDIN
  login       Log in to a Docker registry
  logout      Log out from a Docker registry
  logs        Fetch the logs of a container
  pause       Pause all processes within one or more containers
  port        List port mappings or a specific mapping for the container
  ps          List containers
  pull        Pull an image or a repository from a registry
  push        Push an image or a repository to a registry
  rename      Rename a container
  restart     Restart one or more containers
  rm          Remove one or more containers
  rmi         Remove one or more images
  run         Run a command in a new container
  save        Save one or more images to a tar archive (streamed to STDOUT by default)
  search      Search the Docker Hub for images
  start       Start one or more stopped containers
  stats       Display a live stream of container(s) resource usage statistics
  stop        Stop one or more running containers
  tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
  top         Display the running processes of a container
  unpause     Unpause all processes within one or more containers
  update      Update configuration of one or more containers
```

```
  version     Show the Docker version information
  wait        Block until one or more containers stop, then print their exit codes

Run 'docker COMMAND --help' for more information on a command.

[1mTo get more help with docker, check out our guides at https://docs.docker.com/go/guides/
```

よく使うのは次のようなコマンドです。

docker ps | Dockerプロセスを表示する

```
docker ps
```

docker images | Dockerイメージの一覧を表示する

```
docker images
```

docker attach | Dockerコンテナにアタッチする

```
docker attach <container-id>
```

docker run | Dockerコンテナを実行する

```
docker run <docker-image>
```

docker stop | Dockerプロセスを停止する

```
docker stop <container-id>
```

docker start | Dockerプロセスを開始する

```
docker start <container-id>
```

docker restart | Dockerプロセスを再起動する

```
docker restart <container-id>
```

docker rm | Dockerコンテナを削除する

```
docker rm <container-id>
```

docker rmi | Dockerイメージを削除する

```
docker rmi <container-id>
```

docker logs | Dockerログを取得する

```
docker logs <container-id>
```

docker save | Dockerイメージを保存する

```
docker save <docker-image>:<tag>
```

Dockerfile

DockerコンテナはDockerfileと呼ばれるDockerイメージを作成するための命令を記述したファイルから構築します。サンプルとして、次にCentOSにApache2をインストールして、httpdを実行するDockerfileを示します。

```
#ベースイメージに何を使うか
FROM --platform=amd64 centos:7

# メタデータの設定
LABEL Auther=kingprinters.com
```

```
6
7  # RUNコマンドで、dockerコンテナに必要なツールのインストールや設定を行う
8  RUN yum -y update
9
10 # コンテナに必要なファイルをコンテナに取り込む
11 COPY ./hello.sh /hello.sh
12
13 # コンテナで実行するサービスのポートを公開する
14 EXPOSE 80
15
16 #コンテナ起動時の実行コマンド
17 ENTRYPOINT [ "/hello.sh" ]
18
```

```
1  #!/bin/bash
2  echo "Hello World!"
```

任意のディレクトリを作成し、上記のDockerfileとhello.shを保存しビルド・実行します。

```
1  $mkdir docker-test
2  $cd docker-test
3  $vim Dockerfile(上記の内容を編集)
4  $vim hello.sh
5  $docker build . -t test:v1
6  $docker run test:v1
7  Hello World!
```

FROM | ベースイメージの指定

- コンテナのベースイメージをしています。

```
1  FROM [[--platform]] <docker-image>:<tag>
```

LABEL |メタデータの設定

```
LABEL <key>=<value>
```

RUN | コンテナ作成時のコマンド

- コンテナ作成時に実行するコマンドを記述します。

```
RUN <shell-command>
```

CMD | コンテナ実行時のコマンド

- コンテナ実行時に実行するコマンドを記述します。

```
CMD <shell-command>
```

EXPOSE | ポート開放

- ポートをホストから参照できるように開放します。

```
EXPOSE <port-num>
```

VOLUME | VOLUMEの作成

- Dockerボリュームを作成します。
- VOLUME命令で作成したVOLUMEは、別のコンテナから参照することができます。

```
VOLUME <key>=<value>
```

ADD | ファイルの追加

コンテナにファイルをコピーします。

- リモートからもファイル追加できる
- 圧縮ファイルが自動解凍される

```
ADD <ソース>... <送信先>
ADD ["<ソース>",... "<送信先>"]
```

COPY | ファイルのコピー

- コンテナにファイルをコピーします。
- リモートからのファイル追加は出来ない
- 圧縮ファイルは自動解凍されない

```
COPY <ソース>... <送信先>
COPY ["<ソース>",... "<送信先>"]
```

ENV | 環境変数の設定

- 環境変数を設定します。

```
ENV <key>=<value>
```

コンテナイメージとレイヤー

Dockerfileを書く際にはまず、`FROM`命令でベースとなるイメージを指定します。それに続いて、RUN命令やADD命令を使ってアプリのインストールやホストからコンテナ内のファイルシステムへのファイルの追加などの変更を加えていきます。このようにコンテナは**「変更差分(レイヤ)の集まり」**としてみなすことができます[2]。

では実際に、コンテナイメージがレイヤー化されていることを確認して見ます。`docker save`コマンドを使用することでdockerイメージをtar形式にエクスポートできるので、先程のDockerfileをsaveでtar形式にエクスポートしてみます。

```
$mkdir layers
$docker save test:v1 | tar -xC ./layers
$cd layers
.
├── 291eb894538de0baee3beecbbb57ef8668b00974b05062ff0d78c0dc110820ac
│   ├── VERSION
│   ├── json
│   └── layer.tar
├── 83c614c207652768d8b0d66f31a1a554081acb5ad38e7c4f2d3ced749ce96151
│   ├── VERSION
```

```
11  │   ├── json
12  │   └── layer.tar
13  ├── df402f6ea9f5734de6d3eebdeac230563d574d70dedf6a961645c37dfbc7ee68.json
14  ├── e200f71f3d8a87f1ea0ce068e154df75023b54cc4f354585afb221422416c1b6
15  │   ├── VERSION
16  │   ├── json
17  │   └── layer.tar
18  ├── manifest.json
19  └── repositories
20
21  4 directories, 12 files
```

treeコマンドで出力すると、各レイヤーにVERSIONやjsonの他、layer.tarといったデータが格納されています。layer.tarが、コンテナイメージを構成するレイヤとなります。tarファイルの中身を出力すると実際にDockerfileに記載したファイルがレイヤーに追加されていることがわかります。

```
1  tar --list -f 83c614c207652768d8b0d66f31a1a554081acb5ad38e7c4f2d3ced749ce96151/layer.tar| grep
   hello.sh
2  hello.sh
```

コンテナオーケストレーション

一般的にコンテナ運用を進めると、必然的に複数のコンテナを連携させてサービスを構築することになります。例えば、PHPの場合だと、Webサーバーのプロセスとアプリケーションサーバーのプロセスを協調させる必要があります。このように複数のコンテナを１つのアプリケーションとして協調する仕組みをオーケストレーションと呼びます。

オーケストレーションツールを利用することで、コンテナ上で動作しているアプリケーションを複数のサーバーへの展開、あるコンテナがダウンした時に別のコンテナを動作させるなどのクラスター管理、アプリケーションへ高負荷がかかった時にサーバーを増やす、負荷が減った時にサーバーを減らすなどのスケーリング管理など、複数のコンテナの運用管理を効率的にすることができるようになります[3]。

オーケストレーションツールには、コンテナ運用のデファクトスタンダードであるK8S(Kubernetes)や、AWSのコンテナサービスであるECR、Dockerが提供しているdocker-composeなどがあります。

DockerCompose

DockerComposeは、複数のDockerコンテナを束ねるための技術です。docker-composeコマンドをインストールし、YAML形式で`docker-compose.yml`を記述することで、複数のコンテナをまとめて管理することができるようになります。サンプルとして次に、mysqlとwordpresの環境を構築する`docker-compose.yml`を示します。

```
1  version: '3'
2  services:
3    wordpress:
```

```
    image: wordpress
    container_name: some-wordpress
    restart: always
    ports:
      - 8080:80
    environment:
      WORDPRESS_DB_PASSWORD: my-secret-pw

  mysql:
    image: mysql:5.7
    container_name: some-mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: my-secret-pw
```

マルチステージビルド

　マルチステージビルドとは、複数のFROM句を使ってビルドをステージ単位に分割できる機能です。各ステージは、ビルドに必要なツールを読み込んだり、削除したりするという共通なタスクを実行するベースの Docker イメージを経由してそれぞれのステージのイメージが作成されます

　従来の Docker 環境の作り方の場合、開発環境用のDockerfile を用意して開発に必要なものを一式入れたものと、本番環境用にはアプリケーション本体それを動かすためのランタイムのみを含むスリム化した Dockerfile をもう 1 つ用意するのが一般的でした。（今のdocker-kpなどを参照して下さい）

　マルチステージビルドは、開発環境、テスト環境、本番環境にかかわらず、1 つの Dockerfile で共通部分をベースにして、それぞれの環境のイメージを作成することが可能です。そうすることで可読性も保守性も上がります。

---

1. コンテナとは？仮想サーバーとの違いやLXCやDocker、Kubernetesについて | 勉強ログ (log-bennkyou.com) ↩

2. Dockerコンテナのレイヤ構造とは？ - Qiita ↩

3. コンテナオーケストレーションとは？仕組みやメリット、注目のツールについて徹底解説｜サイバーセキュリティ.com (cybersecurity-jp.com) ↩