# PythonBeginners沖縄 Docker編

@shimoJ.vol9

## ハンズオン内容

- 1. hello-worldの実装
- 2. docker/whalesayの実装とtag付け
- 3. Dockerfileを使用したwhalesayの拡張
- 4. Django REST Framework

## 本日

#### やること

- Dockerの基本と使い方
- Django-REST-Frameworkを使用したapiの実装

## imageとcontainerと DockerHub

- Image・・・・containerを構成する時に元になるファイル
- Container・・imageを基に作成され、実行されている
- Docker hub・gitのようなもの

詳しくは下記サイトを参照

#### Docker初めての人向け説明メモ

https://qiita.com/miyasakura\_/items/87ccb6d4a52d4a00a999

## 資料使い方

#### 入力コマンド

説明

### 1.hello-worldの取得

#### docker image pull hello-world

imageの取得

## 1.hello-worldの実行

docker container run hello-world

runコマンドの実行

### runコマンドについて

- docker pull・・・イメージの取得
- docker create ・・コンテナの作成
- docker start・・・コンテナの起動

#### 上記3つをまとめて実行してる

# 2.whalesayの取得&実行

docker container run docker/whalesay cowsay yahho-!

whalesayのimageを取得し実行する

# imageの確認

#### docker images

REPOSITORY 名称

TAG 特定のimageを識別するため

IMAGE ID imageを一意に識別するために利用するID

CREATED imageが作成されてからの時間

SIZW imageのfileサイズ

# docker tagについて1

docker tag [既存Dockerイメージ名] [付与する Dockerイメージ名]

Dockerイメージに別名を付けれる

Tag・・タグ付けするサブコマンド

# docker tagについて2

docker tag docker/whalesay my\_whalesay

docker/whalesayを元に新しく作成

元のimageと同じIMAGE ID

# docker tagについて3

docker tag docker/whalesay my\_whalesay:ver1

tag名を指定してる

指定しやすくなるのでつけた方が良い

# imageの削除

#### docker rmi 'IMAGE ID'

#### 指定したREPOSITORYの削除

- tagの指定をしないとlatestが指定される
- 強制削除する場合は rmi -f とする

## containerの確認

#### docker ps

実行中のcontainerを表示する

## containerの確認

#### docker ps -a

処理済みのcontainerも表示する

## containerの確認

CONTAINER ID	コンテナに付与一意のID
IMAGE	コンテナ作成に使用されたDocker image
COMMAND	<u>コンテナで実行されているアプリケーションプロセス</u>
CREATED	コンテナが作成されてから経過した時間
STATUS	Up(実行中),Exited(終了)などコンテナの実行状況
PORTS	ホストのポートとコンテナポートの紐付け
NAMES	コンテナにつけられた名前

## containerの削除①

#### docker rm 'コンテナ ID'

指定したcontainer IDを削除する

## containerの削除②

docker rm \$(docker ps -q -f status=exited)

status=exited状態のcontainerをす べて削除する

## 実際に削除しよう

my\_whalesay:ver1のimageを削除してください

exsitedになっているcontainerを削除してください

## 3.whalesayに機能を追加①

カレントディレクトリに移動

docker build -t docker-whale .

Dockerfileを元にimageをbuild

## 3.whalesayに機能を拡張②

#### docker run docker-whale

作成したimageを実行する

## 今回のDockerfile構造

docker/whalesay:latestの imageを元に作成 ・image buildの際に実行されるコマンド

-apt-getでパッケージをリストを更新

-fortunesをインストールするコマンドを指定

※-yは各コマンドが実行確認のメッセージで処

理が止まらないようにつけてる

```
最初に実行される
コマンド
```

```
FROM docker/whalesay:latest

RUN apt-get -y update && apt-get install -y fortunes

CMD /usr/games/fortune | cowsay

CMD /usr/games/fortune | cowsay
```

# Djnagoの構築に入って いきます

### docker-composeについて1

- マルチコンテナをドッカーアプリケーションを定義して実 行するためのツール
- dbサーバーweb サーバーなどを1つのymlファイルに定 義してまとめて実行できる
- ymlは構造化されたデータを表現するためのフォーマット

### docker-composeについて2

- 1. Dockerfileを用意する
- 2. docker-compose.ymlを定義する
- 3. docker-compose upを実行する

# 4.Django開発環境構築1

## カレントディレクトリに移動

docker-compose run web django-admin.py startproject django\_rest\_framework\_test.

Dockerfileを元にコンテナを作成し、 Djangoのプロジェクトを生成する

# 4.Django開発環境構築2

#### Postgresの設定

django\_rest\_framework\_test/settings.pyの以下を変更

- 1. ALLOWED\_HOSTS = ['localhost'] (28行目)
- 2. DATABASESを変更(84行目)

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'postgres',
        'USER': 'postgres',
        'HOST': 'db',
        'PORT': '5432',
    }
}
```

# 4.Django開発環境構築3

-d:back-groundで作業する

### docker-compose up -d

↑起動! localhost/8000番を確認

#### docker-compose stop

↑停止!

## dockerfileの説明

#### python3の実行環境を指定

境変数の設定(pythonの標準出力、標準エラー出力を溜め込まない設定)

#### /codeを作成してる

#### 作業場所を/codeに指定

codeディレクトリにtxtファイル を置く

```
pipインストールを実施
-rで指定しrequirements.txtの
インストールを実行
```

```
Dockerfile docker-compose

1 FROM python:3

2 ENV PYTHONUNBUFFERED 1

3 RUN mkdir /code

WORKDIR /code

5 COPY requirements.txt /code/
RUN pip install -r requirements.txt

COPY . /code/
```

build contextの内容を/code内 においている

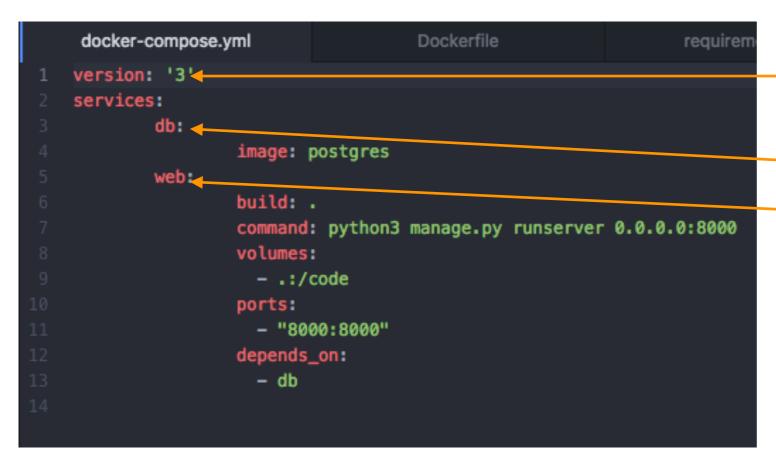
## requirements.txtの説明

```
requirements.txt

Django==2.0
djangorestframework
spsycopg2
psycopg2-binary
django-filter
```

pipインストールコマンド でインストールするパッケー ジを指定している

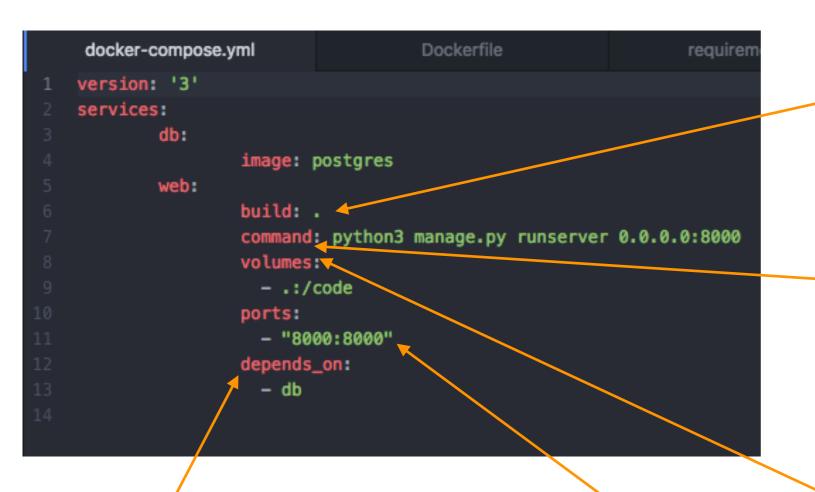
### docker-compose.ymlの説明1



docker-composeの versionを指定

servicesには2つの containerを指定。実行 時にこの2つの containerが起動する

### docker-compose.ymlの説明2



docker-compose.ymlと同じフォルダをbuild contextとしている.Dockerfileを同じフォルダに作成する必要がある

container起動時に 実行されるコマンド

dbを先に立ち 上げる依存関係 を指定 containerを8000 番で公開して8000 番に転送している カレントディレクトリ(.)を/codeにバイントマウントしている

#### コンテナをに入るコマンド

### docker exec -it コンテナ名 bash

↑containerに入り作業するコマンド

#### control + p & control + q

↑bashから抜けるコマンド

### 下記サイトを参考にapiを実装

• Docker コマンドチートシート

https://qiita.com/wMETAw/items/ 34ba5c980e2a38e548db

 REST Frameworkを使って爆速でAPIを実装する https://qiita.com/kimihiro\_n/items/ 86e0a9e619720e57ecd8

## 手順1

- 1. bashでcontainer内部に入る
- 2. python manage.py startapp blog

editerツールを使用して編集が可能

- 3. blog/models.pyを変更
- 4. django\_rest\_framework\_test/settings.py を変更
- 5. python manage.py makemigrations
- 6. python manage.py migrate
- 7. admin用のユーザー作成

http://localhost:8000/admin に入り動作確認

## 手順2

- 1. django\_rest\_framework\_test/settings.py の編集
- 2. blog/serializer.py を作成する
- 3. blog/views.py を編集する
- 4. django\_rest\_framework\_test/urls.pyを編集
- 5. blog/urls.pyを作成