

# Docker Hands On

宮島健太



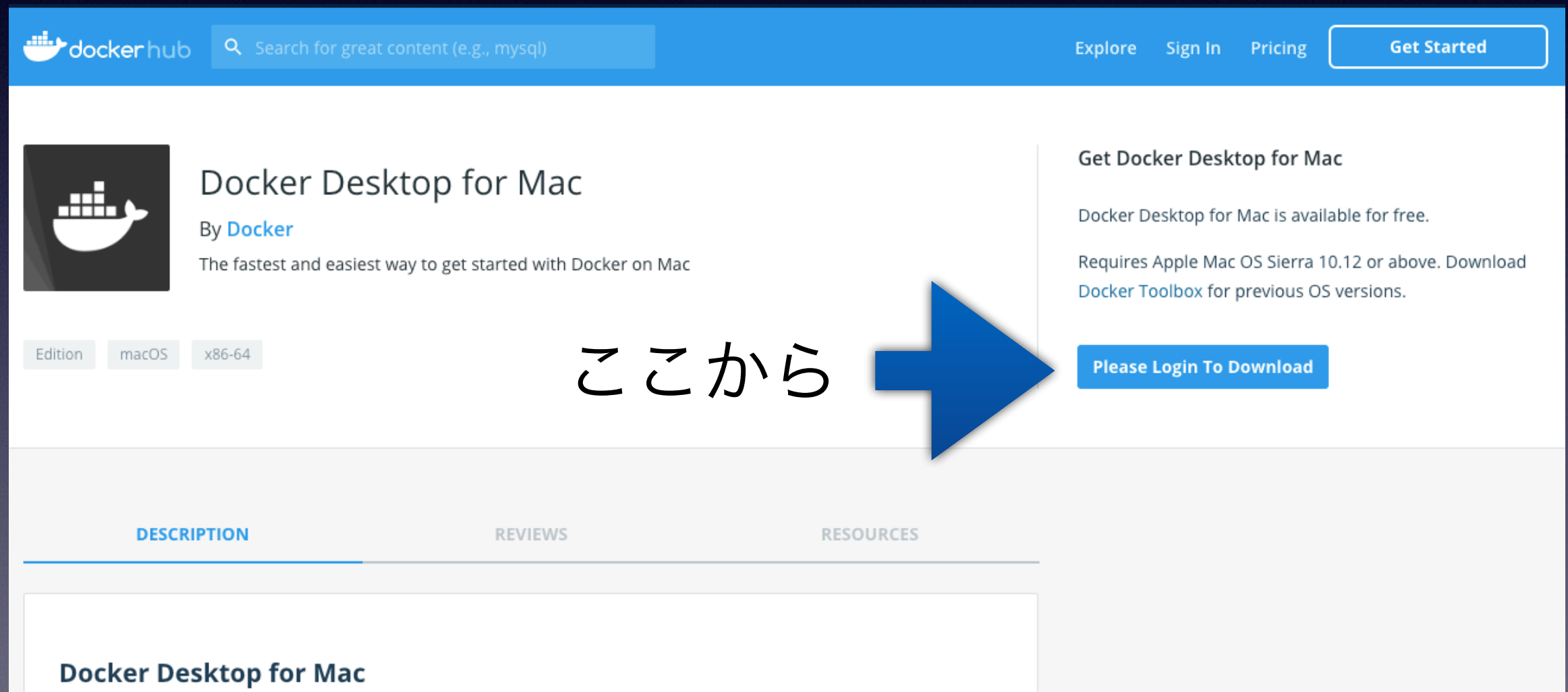
# 行うこと

- ・ Dockerの環境構築
- ・ コンテナの生成
- ・ コンテナの使い方
- ・ コンテナ間通信



# 環境構築

<https://hub.docker.com/editions/community/docker-ce-desktop-mac>



The screenshot shows the Docker Hub interface for 'Docker Desktop for Mac'. The top navigation bar includes the Docker Hub logo, a search bar, and links for 'Explore', 'Sign In', 'Pricing', and 'Get Started'. The main content area features the Docker Desktop for Mac logo, the title 'Docker Desktop for Mac', and the text 'By Docker' and 'The fastest and easiest way to get started with Docker on Mac'. Below this, there are tabs for 'Edition', 'macOS', and 'x86-64'. A large blue arrow points from the text 'ここから' (From here) to the 'Please Login To Download' button. The right sidebar contains the text 'Get Docker Desktop for Mac', 'Docker Desktop for Mac is available for free.', and 'Requires Apple Mac OS Sierra 10.12 or above. Download Docker Toolbox for previous OS versions.' Below the main content area, there are tabs for 'DESCRIPTION', 'REVIEWS', and 'RESOURCES'. The 'DESCRIPTION' tab is selected, showing the title 'Docker Desktop for Mac'.

アカウントを作ってダウンロード



# 動作確認

```
$ docker
```



# どんな仮想環境が作れるのか？

<https://hub.docker.com>

ubuntu, centos,  
mysql, postgresql, redis,  
nginx, apache,  
golang, python, etc...

→ **Image**と呼ばれる



# MySQLのイメージのDL

```
$ docker pull mysql
```

(バージョン指定がない場合は最新)



# コンテナの生成

```
$ docker run --name TEST \  
-e MYSQL_ROOT_PASSWORD=password \  
-d mysql
```

MySQLのコンテナを立てる。

コンテナ名：TEST

MySQL-root-Pass：password



# コンテナが生成できているか

```
$ docker ps
```

Docker上で動いているコンテナのプロセスを確認

```
$ docker ps -a
```

Docker上に存在するコンテナのプロセスを確認



# コンテナの使用 (コンテナにログインする)

```
$ docker exec -it コンテナID bash
```

コンテナが動いているとこのOSのbashに入ります



# MySQLにログイン

```
$ mysql -u root -p
```

先ほど設定したrootのパスワードを入力する。

あとはDBをいじる。



# 使わないなら止めましょう

\$ docker stop コンテナID

\$ docker stop コンテナ名



いらなくなったら  
消しましょう。

\$ docker rm コンテナID



ここまでがDockerの  
基本的な使い方



# コンテナの生成から削除まで 自動でやるには

`docker-compose` コマンドを使用する  
`Dockerfile`, `yaml` を使う



# Docker Compose

建てたいコンテナの設定を  
`docker-compose.yaml`  
に書いておく。

デプロイ設定は`Dockerfile`に記載する。



# デプロイと生成

\$ docker-compose build

\$ docker-compose up -d

それぞれデプロイ・それ以外のコンテナ生成

\$ docker-compose down

で破棄する。



# Docker Compose

1つのコンテナにイメージか実行環境

実行環境がある場合、Dockerfileでデプロイする



# (個人的な) メリット

複数のコンテナの設定を記述することができる。

- 一気にコンテナを立ち上げることが可能
- コンテナ間通信の設定も出来る
- 1つのアプリケーションを全て自動で仮想化

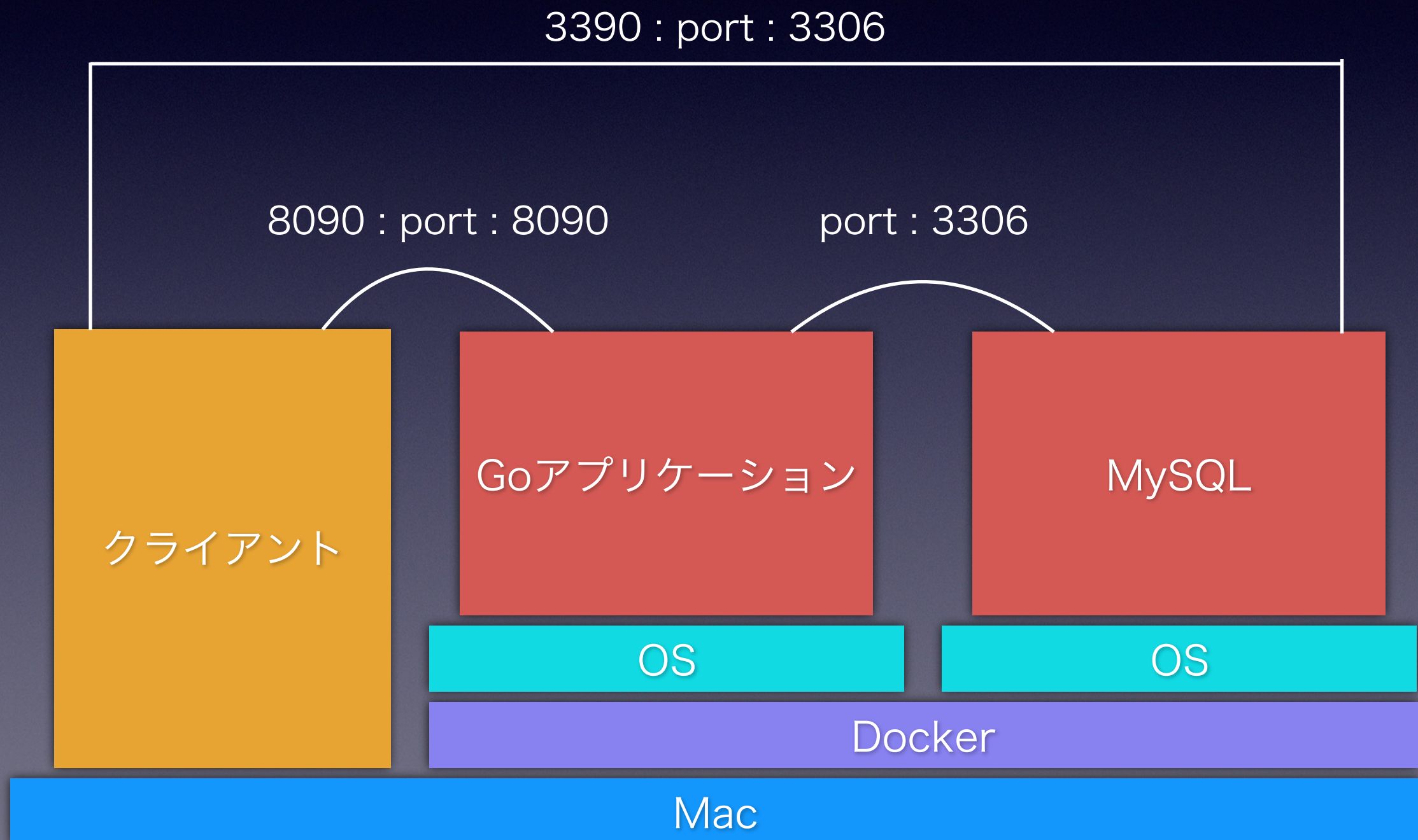


(個人的な) デメリット

非常にややこしい。



# 構築するもの





やってみます。



# まとめ

- docker runで簡単にコンテナを生成
- docker内の通信はdockerのlocalhost
- ホストOSからdockerへはポートフォワーディング
- 仮想環境自体はdocker-compose.yamlに記述
- デプロイの設定はDockerfileに記述