Internet Week 2018

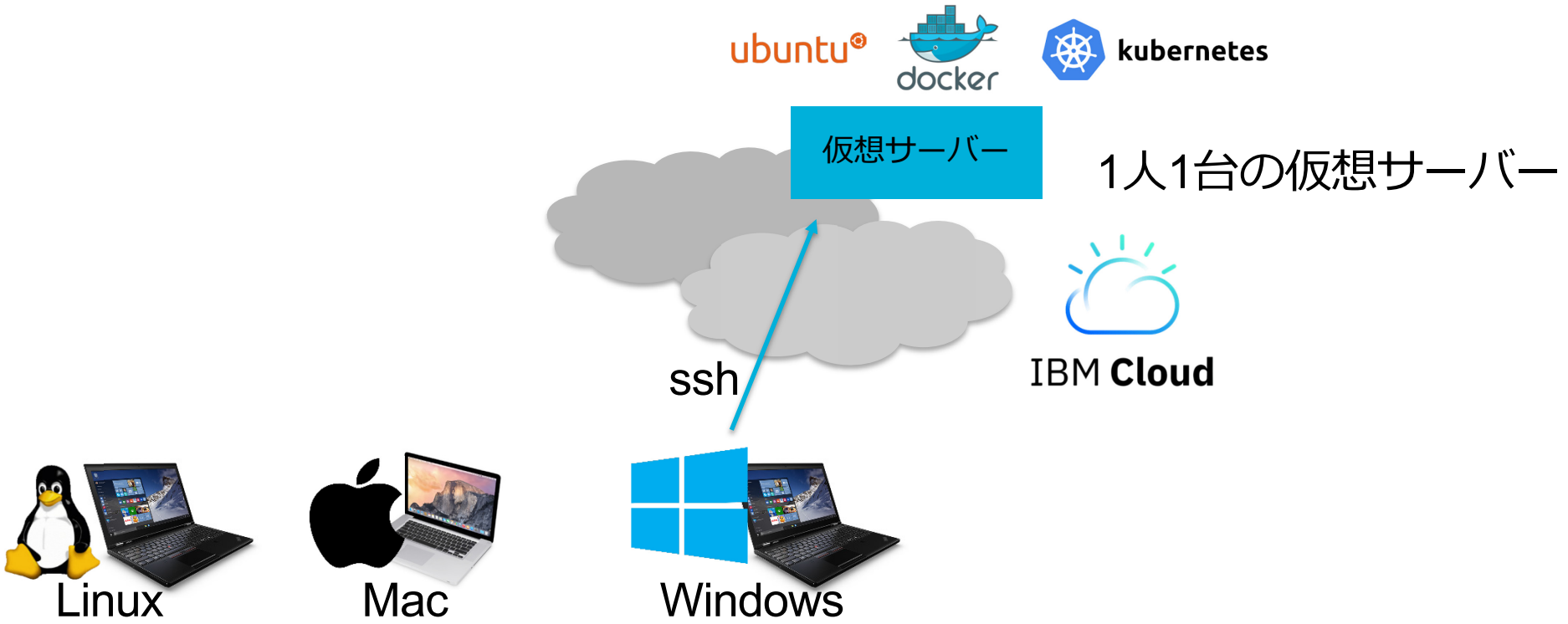
Dockerハンズオン

2018年11月28日

日本アイ・ビー・エム株式会社
クラウド事業本部 高良 真穂

本日のハンズオンの環境

– このハンズオンセッションでは、以下を利用します。
  - 想定クライアント： Windows PC、Mac、Linux
  - 必要なクライアントソフト： sshクライアント（Windowsの場合はTeraTerm,Putty等）



ログイン後の確認

ログインに成功すると、次のようなメッセージが表示されます。

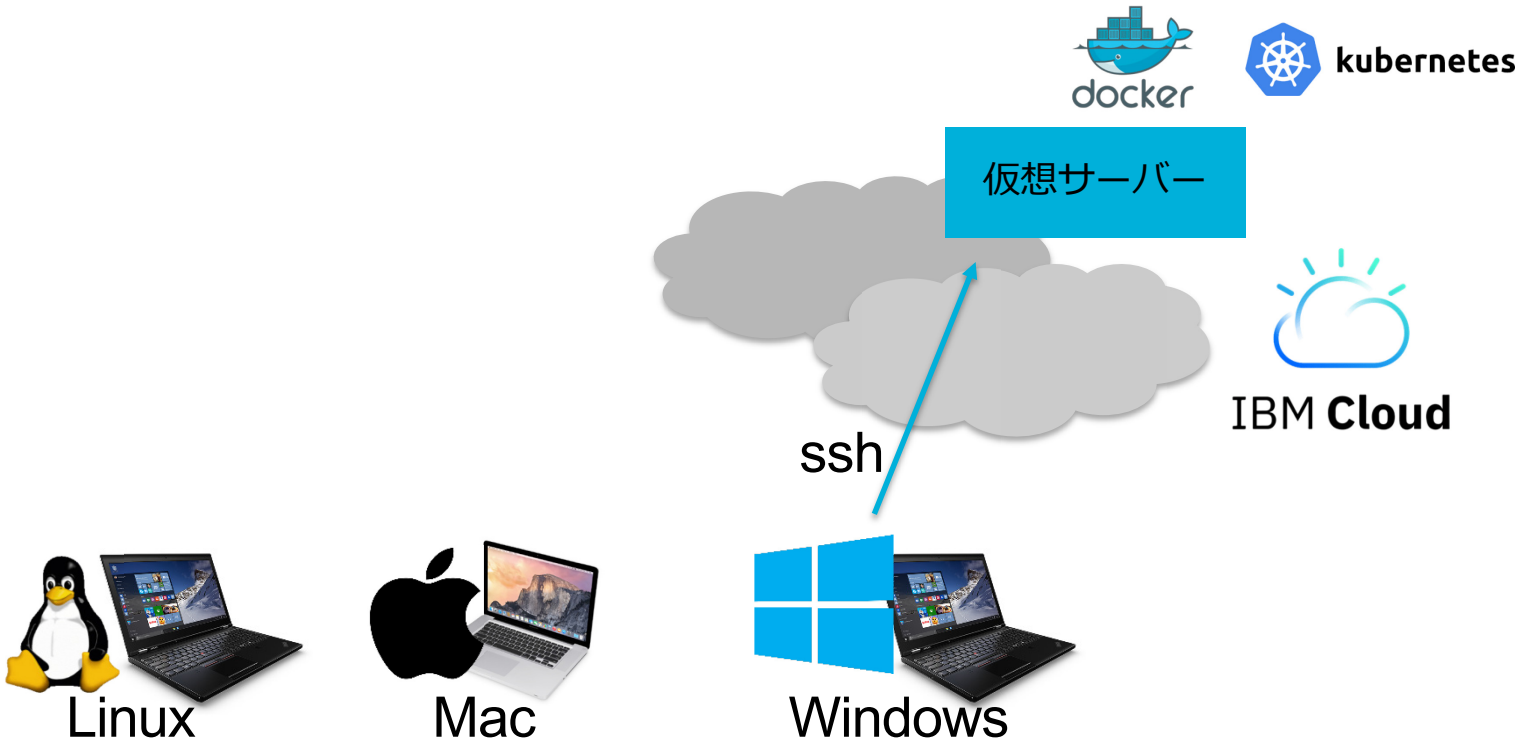
```
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.4.0-135-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
<以下省略>
```

次のコマンドを実行して、自分の仮想サーバーのIPアドレスを確認してください。
一致しない場合は、挙手してTAに知らせてください。

```
minion@iw01:~$ ip addr show dev eth1
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default
qlen 1000
    link/ether 06:fc:94:19:b5:d4 brd ff:ff:ff:ff:ff:ff
    inet 128.168.66.221/28 brd 128.168.66.223 scope global eth1
```

ハンズオンで利用するコードの入手

– 回線容量の都合上、仮想サーバー上でgit clone をお願いします。
  • git clone https://github.com/takara9/k8s-hands-on



レッスン＃1

コンテナの基本動作の理解

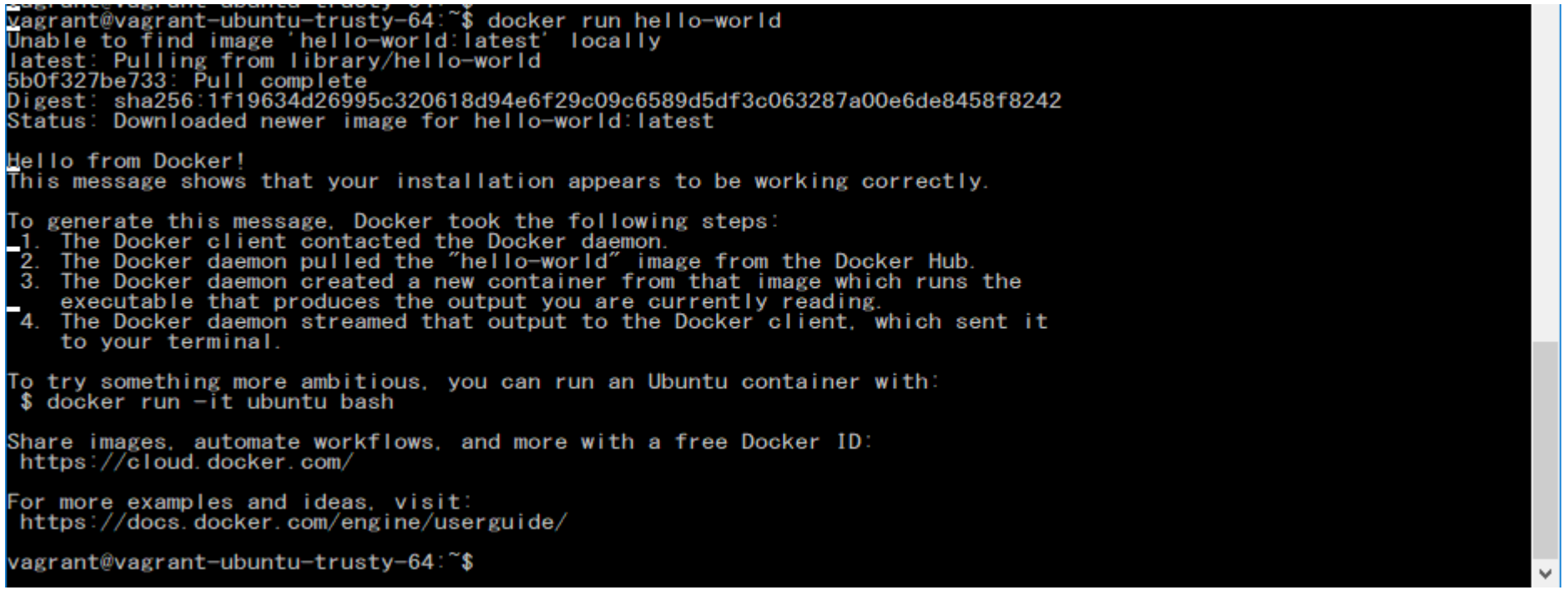
演習

コンテナを実行してメッセージが表示されることを確認してください。

Dockerコンテナ hello-world を実行して、動作を確認します。

```
docker run hello-world
```

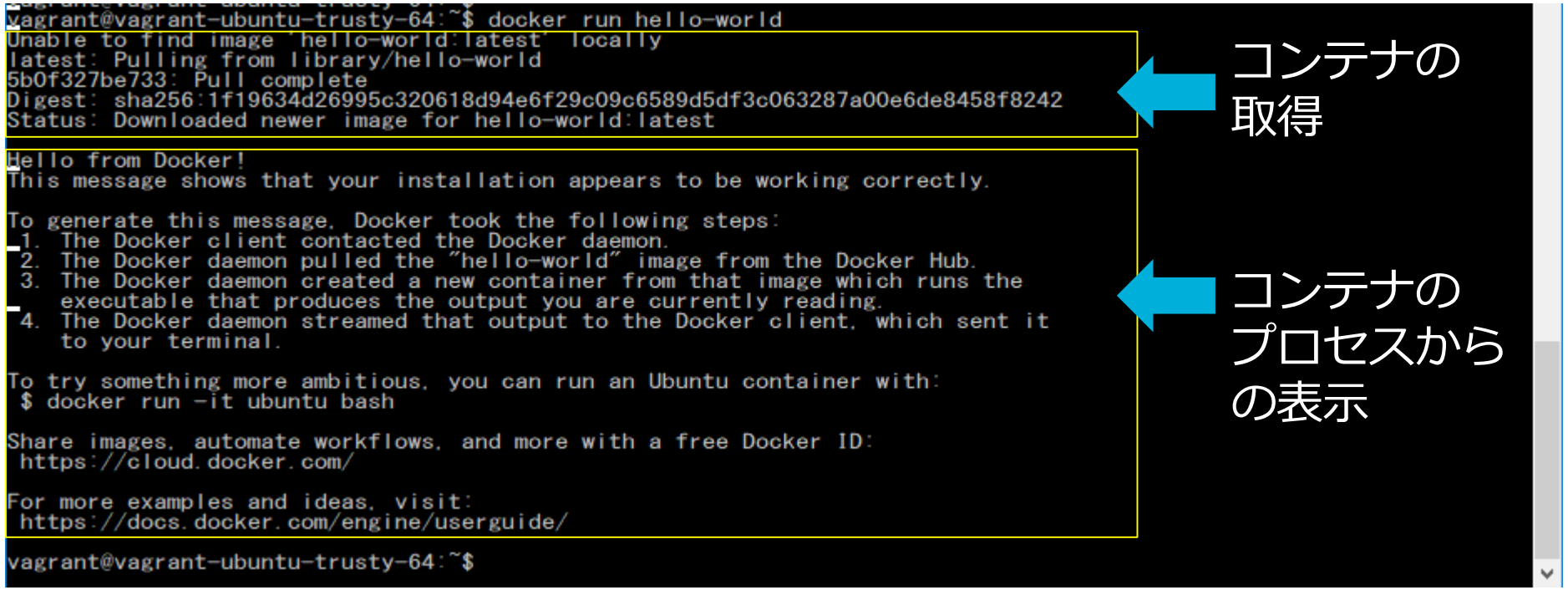
以下のように表示されていれば、コンテナが正常に実行されています。



Dockerコマンドのメッセージを確認してください。

演習

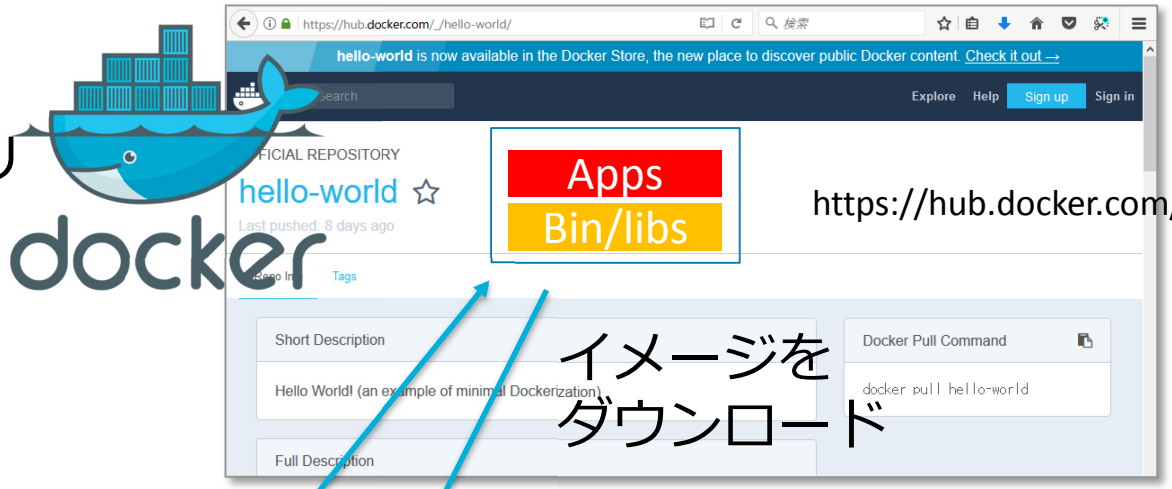
二つの種類のメッセージが表示されています。



Hello-Worldメッセージ表示までの過程

– 画面表示は、①～④の動作によって、得られた表示です。

DockerHub
コンテナレジストリ



https://hub.docker.com/_/hello-world/

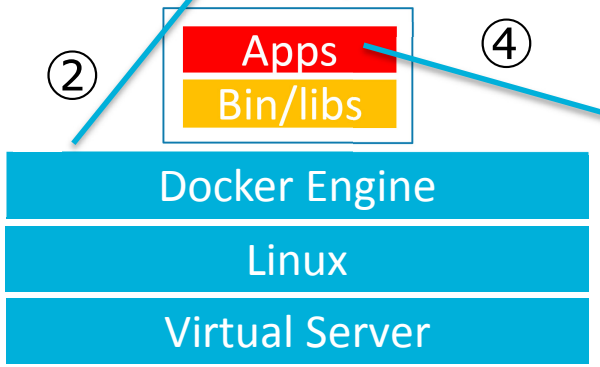
イメージを
ダウンロード

③

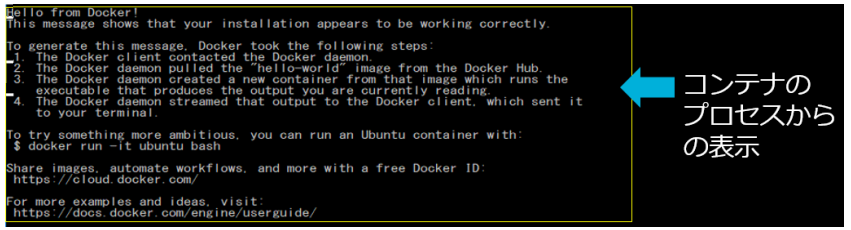
① コマンド実行

```
docker run hello-world
```

②



④ コンテナを実行＆メッセージ出力



コンテナの
プロセスから
の表示

レッスン#1のまとめ

1. コンテナを実行するにはdockerコマンドを利用
2. Dockerエンジンは、Docker Hub リポジトリから、イメージをダウンロードする
3. コンテナを生成してコードを実行する

レッスン＃２

コンテナで対話モードのシェルを実行

コンテナを対話モードで起動

演習

CentOS7の最新バージョンのコンテナを実行して、対話モード(-it) で、bashを実行します。

コマンド実行

```
docker run -it centos:7 bash
```

実行結果

```
vagrant@vagrant-ubuntu-trusty-64:~$ docker run -it centos:7 bash
Unable to find image 'centos:7' locally
7: Pulling from library/centos
d9aaf4d82f24: Pull complete
Digest: sha256:eba772bac22c86d7d6e72421b4700c3f894ab6e35475a34014ff8de74c10872e
Status: Downloaded newer image for centos:7
[root@d41d9525359c /]#
```

コンテナに入ったら、次ページに進む適当にコマンドを実行して、CentOSであることを確かめてください。

コンテナ上でコマンドを実行してCentOSを確認

演習

(1) Red Hat のリリースを表示

```
[root@b0d2bb2cbcd8 /]# cat /etc/redhat-release
```

(2) カーネルのバージョンを確認

```
[root@b0d2bb2cbcd8 /]# uname -a
```

(3) yum update して、ファイルを作っておきます

```
[root@b0d2bb2cbcd8 /]# yum update -y
[root@b0d2bb2cbcd8 /]# date > date.txt
```

(4) シェルから出ます。

```
[root@b0d2bb2cbcd8 /]# exit
```

コンテナホストのOSとカーネルを確認して、再びコンテナを対話モードで起動します。

演習

(5) コンテナホストのリリースを表示

```
minion@iw01:~$ cat /etc/issue
```

(6) コンテナホストのカーネルのバージョンを表示

```
minion@iw01:~$ uname -a
```

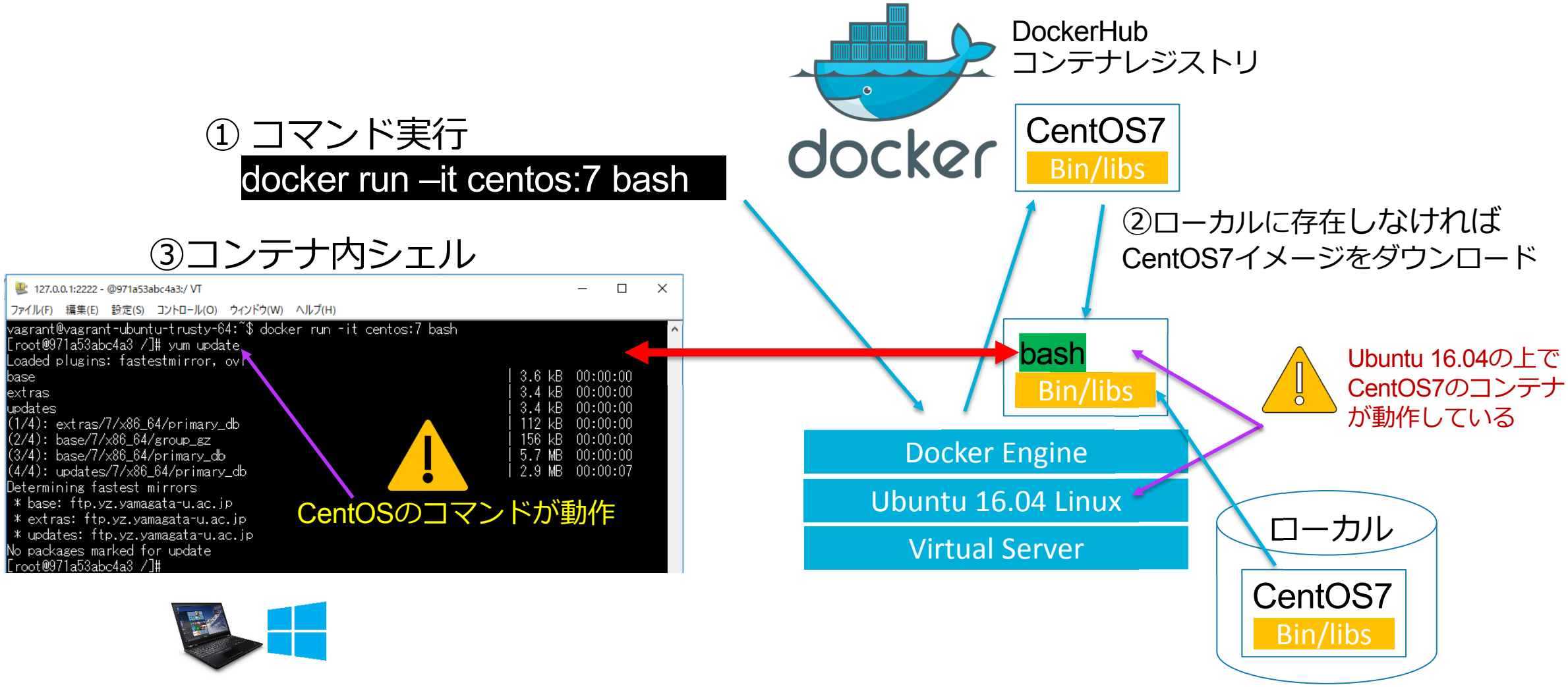
(7) CentOS コンテナをもう一度起動します。

```
minion@iw01:~$ docker run -it centos:7 bash
[root@cf0d13ee33c7 /]#
```

今度は、一瞬で起動しました。

コンテナのレジストリとコンテナ

コンテナレジストリからダウンロードしたイメージはローカルに保存され、再利用される。



最初に作ったファイルは存在していますか？

演習

(8) コンテナのファイルシステムで、ファイルをリストします。

```
[root@cf0d13ee33c7 /]# ls
```

存在しませんね。どこにあるか、
次のレッスンで探して行きましょう！

レッスン#2のまとめ

1. DockerHubのリポジトリを指定してコンテナを実行できる
2. Ubuntu Linux 上で、CentOSのコンテナが動作した
3. カーネルは、コンテナホストとコンテナで同じである
4. ２回目はローカルリポジトリを利用し瞬時に起動する

レッスン＃3

コンテナ とイメージの関係を学ぶ

コンテナのリストを表示して、コンテナを指定して開始します。

演習

(1) これまで作ったコンテナの全てのリストを表示します。

```
minion@iw01:~$ docker ps -a
```

```
minion@iw01:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
minion@iw01:~$
minion@iw01:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS               NAMES
f3935b574547        centos:7            "bash"              About an hour ago   Exited (0) 33 minutes ago                       laughing_nobel
66d1541bf852        centos:7            "bash"              About an hour ago   Exited (0) 31 minutes ago                       nervous_noyce
```

コンテナ に付与された名前

コンテナ名を指定して対話型でシェルを実行してファイルをレッスン#2(3)で作成したファイルを探してください。

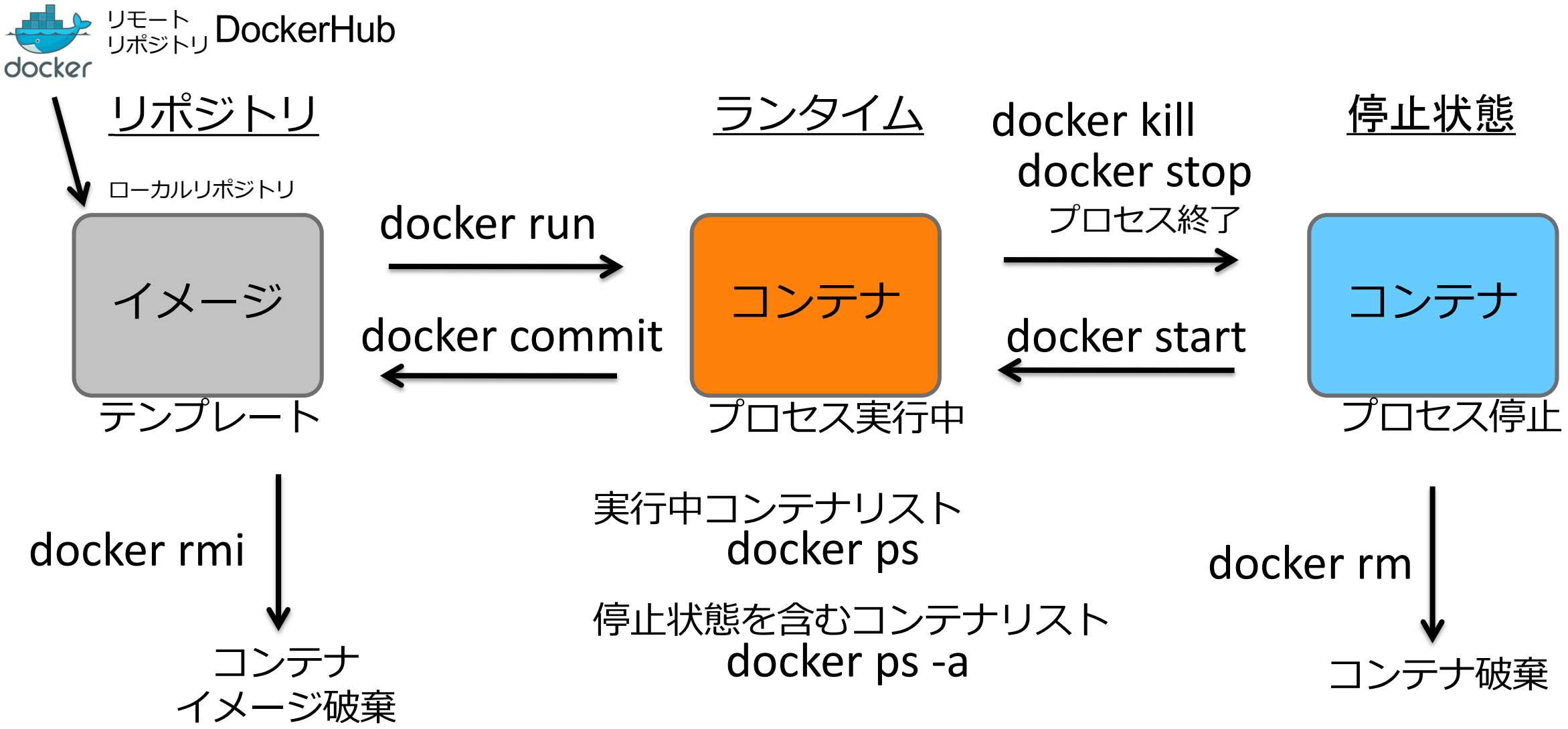
演習

(2) NAME文字列をコンテナ名に置き換えて、シェルを対話型で実行します。
両方のコンテナに入って、作成したファイルを確認しましょう。

```
minion@iw01:~$ docker start -i  NAME bash
```

二つのコンテナは、独立したファイルシステムを持っていることを確認してください。

コンテナの状態遷移とコマンド



コンテナの基本的なコマンド

リポジトリからコンテナを生成して、対話型でコマンドを実行する

```
docker run -it [-name コンテナ名] リポジトリ名 コマンド
```

停止中コンテナを再スタートして、対話型でコマンドを実行する

```
docker start -i コンテナ名 コマンド
```

実行中コンテナ停止する

```
docker stop コンテナ名
```

コンテナ削除する

```
docker rm コンテナ名
```

レッスン#3のまとめ

1. イメージはコンテナの雛形である。
2. 各コンテナ は分離されたファイルシステムを持つ。
3. コンテナは停止しても保持され再び開始できる

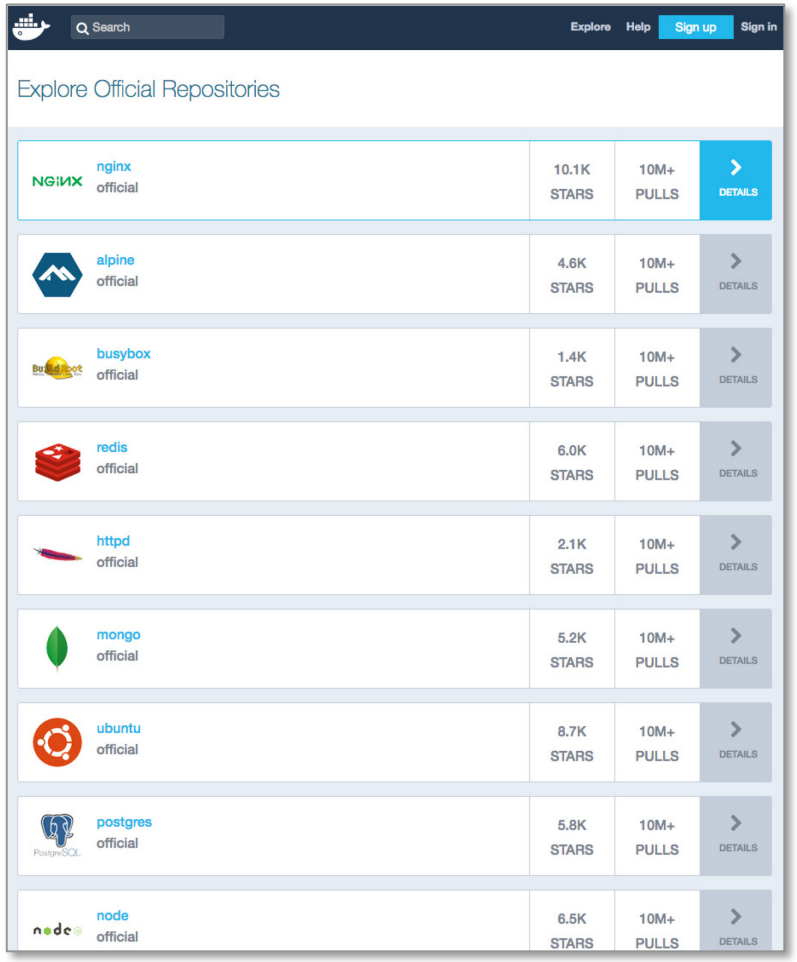
（Kubernetesは違います）

レッスン＃4

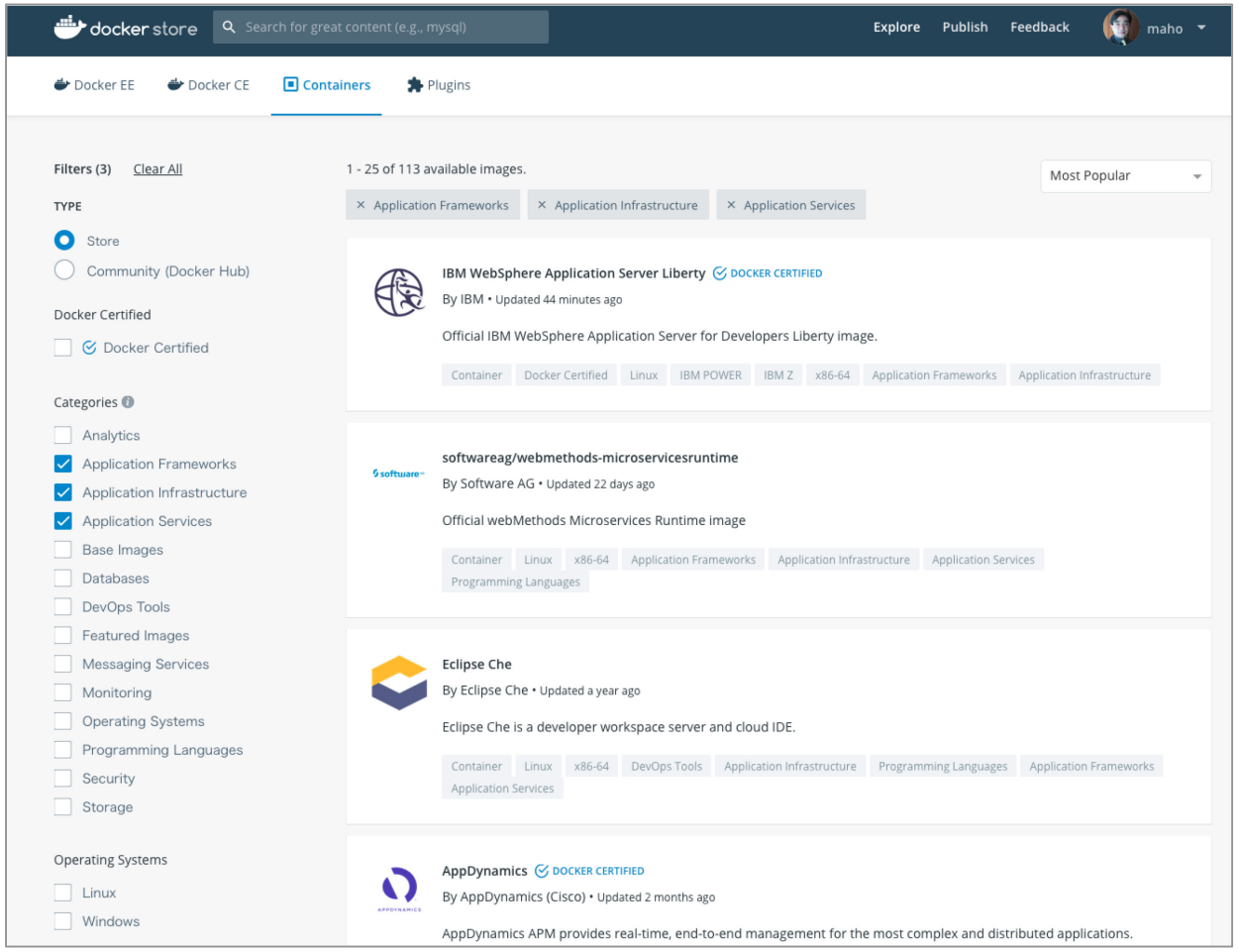
Dockerレジストリを
探検してみましょう

コミュニティとソフトウェア企業のコンテナのイメージをスグに利用できる。

https://hub.docker.com/explore/



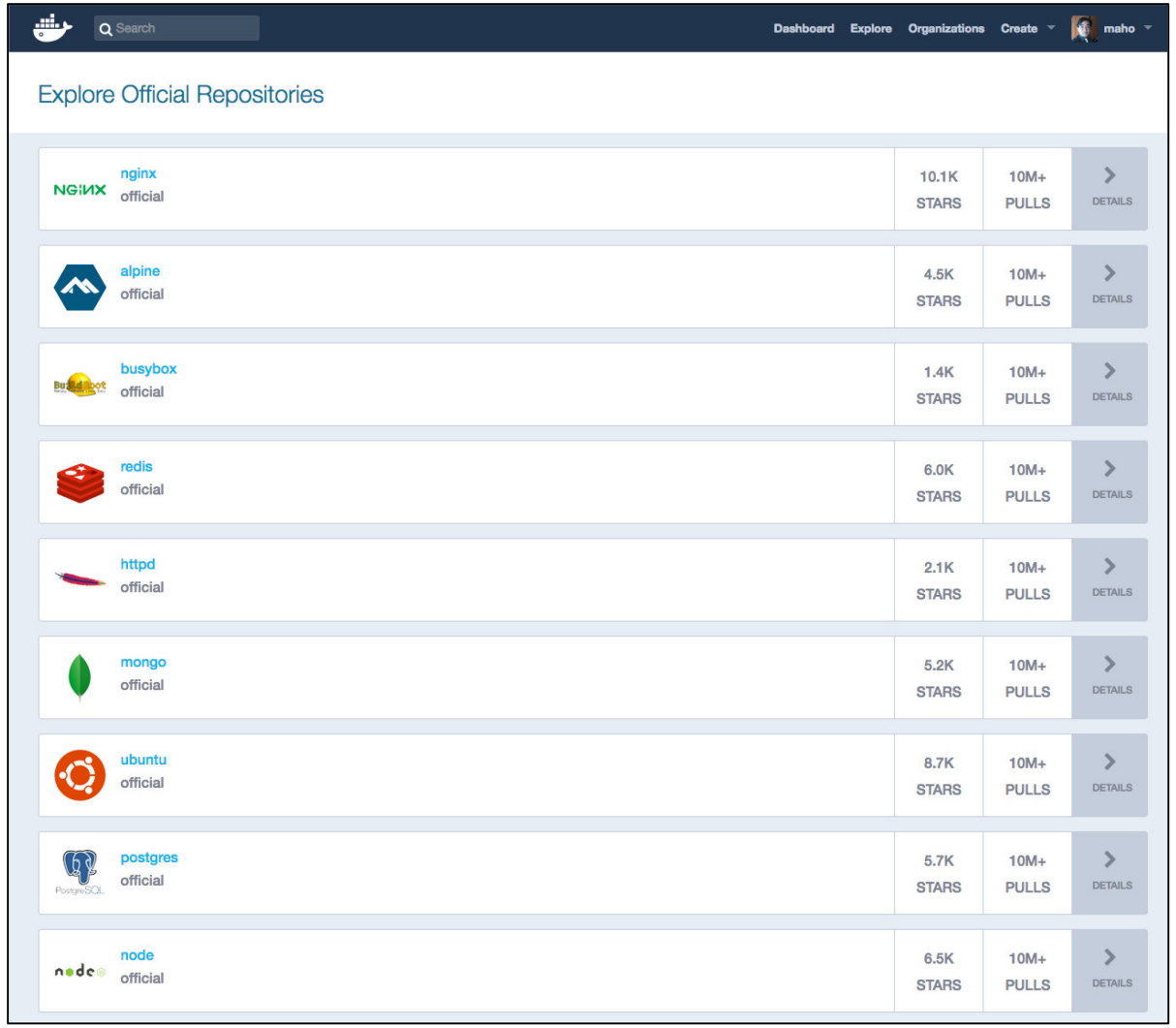
https://store.docker.com/



DockerHubのウェブにアクセスして人気順リストを閲覧してください

演習

https://hub.docker.com/explore/ には pullされた回数順にリポジトリがリストされてます。

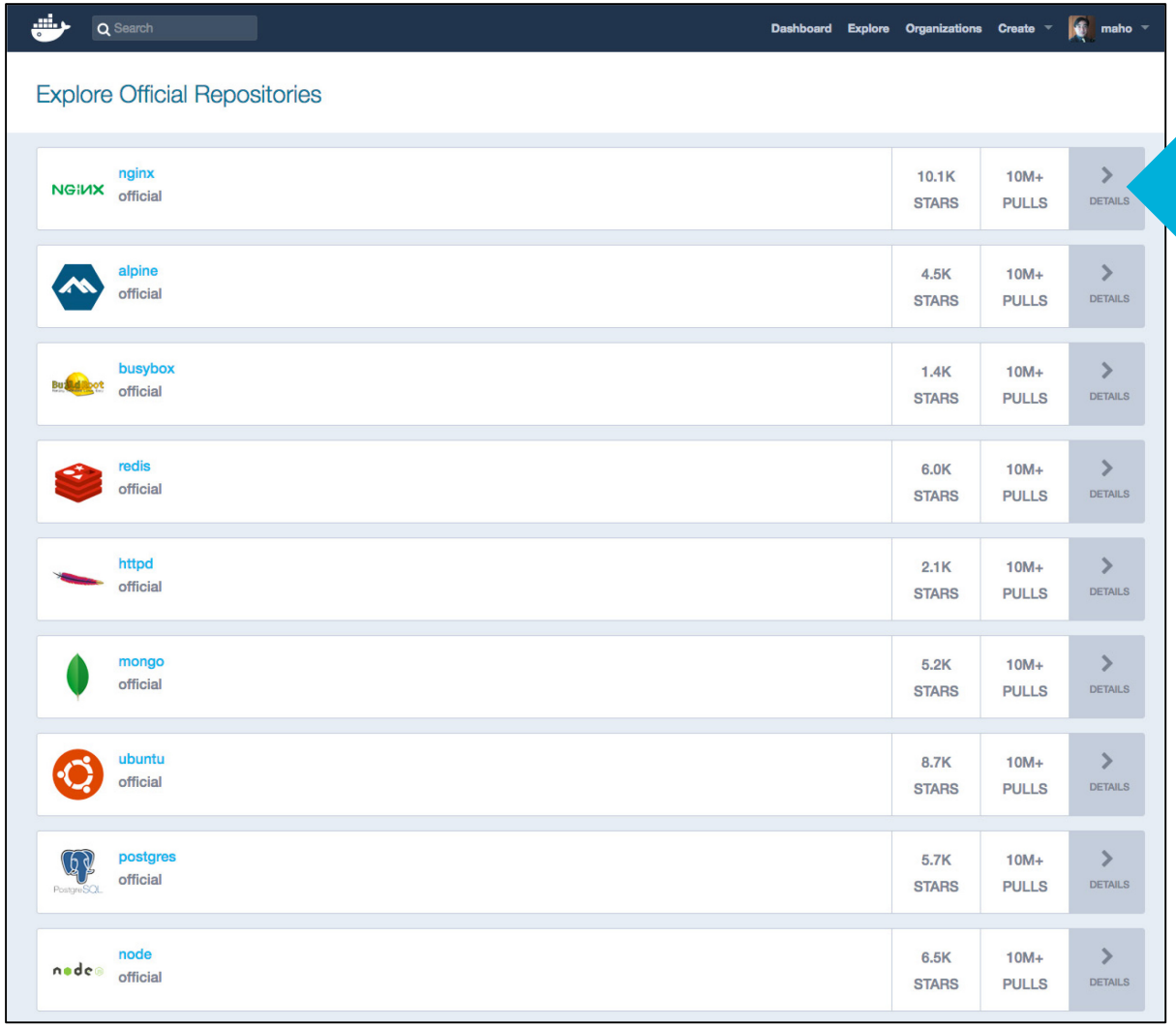


どの様なイメージが登録されているか
ご自身で確認して見てください。

3 分間

一番人気のNginxの詳細に見てきましょう。

https://hub.docker.com/explore/ には戻って、DETAILSをクリックしてください。



クリック

Nginxの詳細内容を見て行きましょう。

ウェブページの上から順に見て行きます。

タグはイメージを識別するために付与されます

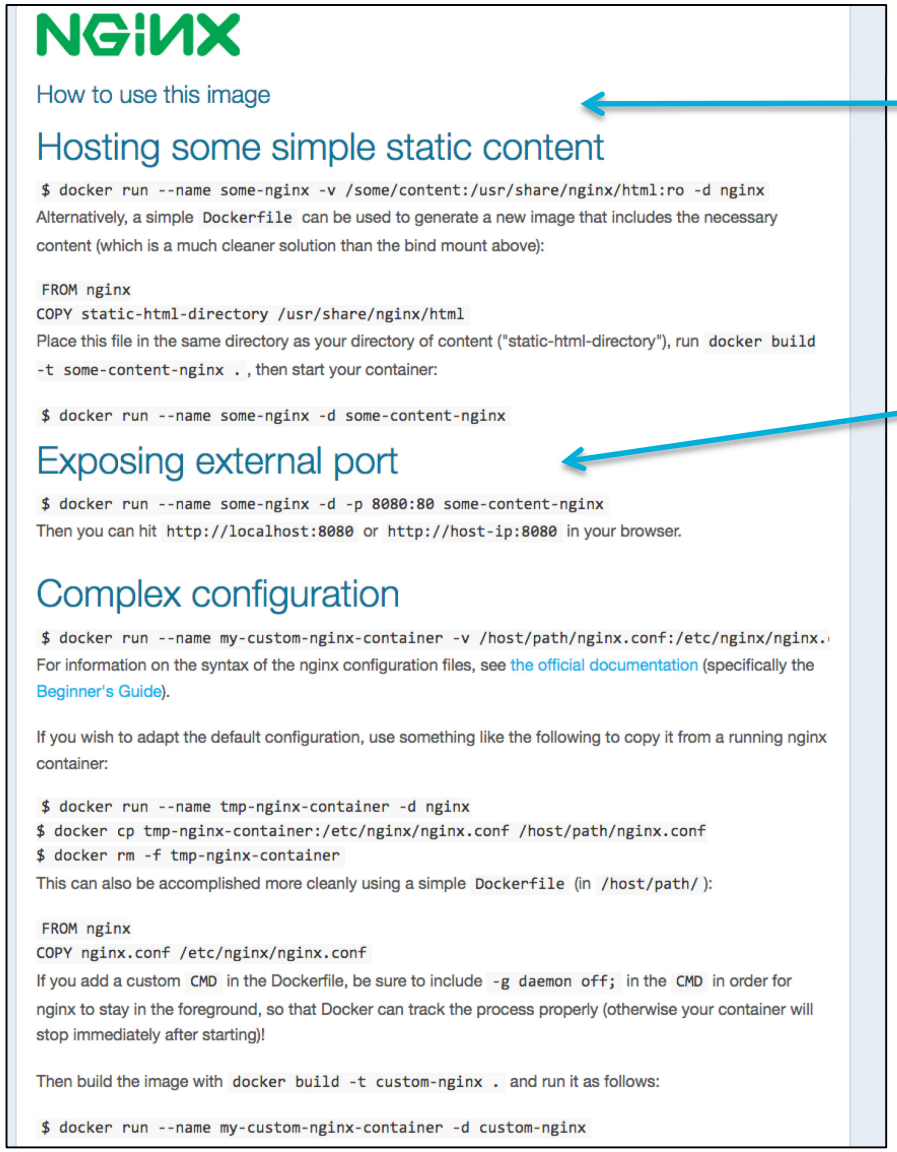
ここにあるイメージが登録されています。
リストを見るには
Tagsをクリック

イメージをビルドするための
Dockerfileへのリンク

ここに挙げられたCPUの
イメージが登録されています

Nginxの詳細内容

コンテナの利用方法が記載されています。



NGINX

How to use this image

Hosting some simple static content

```
$ docker run --name some-nginx -v /some/content:/usr/share/nginx/html:ro -d nginx
```

Alternatively, a simple `Dockerfile` can be used to generate a new image that includes the necessary content (which is a much cleaner solution than the bind mount above):

```
FROM nginx
COPY static-html-directory /usr/share/nginx/html
```

Place this file in the same directory as your directory of content ("static-html-directory"), run `docker build -t some-content-nginx .`, then start your container:

```
$ docker run --name some-nginx -d some-content-nginx
```

Exposing external port

```
$ docker run --name some-nginx -d -p 8080:80 some-content-nginx
```

Then you can hit `http://localhost:8080` or `http://host-ip:8080` in your browser.

Complex configuration

```
$ docker run --name my-custom-nginx-container -v /host/path/nginx.conf:/etc/nginx/nginx.
```

For information on the syntax of the nginx configuration files, see the official documentation (specifically the Beginner's Guide).

If you wish to adapt the default configuration, use something like the following to copy it from a running nginx container:

```
$ docker run --name tmp-nginx-container -d nginx
$ docker cp tmp-nginx-container:/etc/nginx/nginx.conf /host/path/nginx.conf
$ docker rm -f tmp-nginx-container
```

This can also be accomplished more cleanly using a simple `Dockerfile` (in `/host/path/`):

```
FROM nginx
COPY nginx.conf /etc/nginx/nginx.conf
```

If you add a custom `CMD` in the Dockerfile, be sure to include `-g daemon off;` in the `CMD` in order for nginx to stay in the foreground, so that Docker can track the process properly (otherwise your container will stop immediately after starting)!

Then build the image with `docker build -t custom-nginx .` and run it as follows:

```
$ docker run --name my-custom-nginx-container -d custom-nginx
```

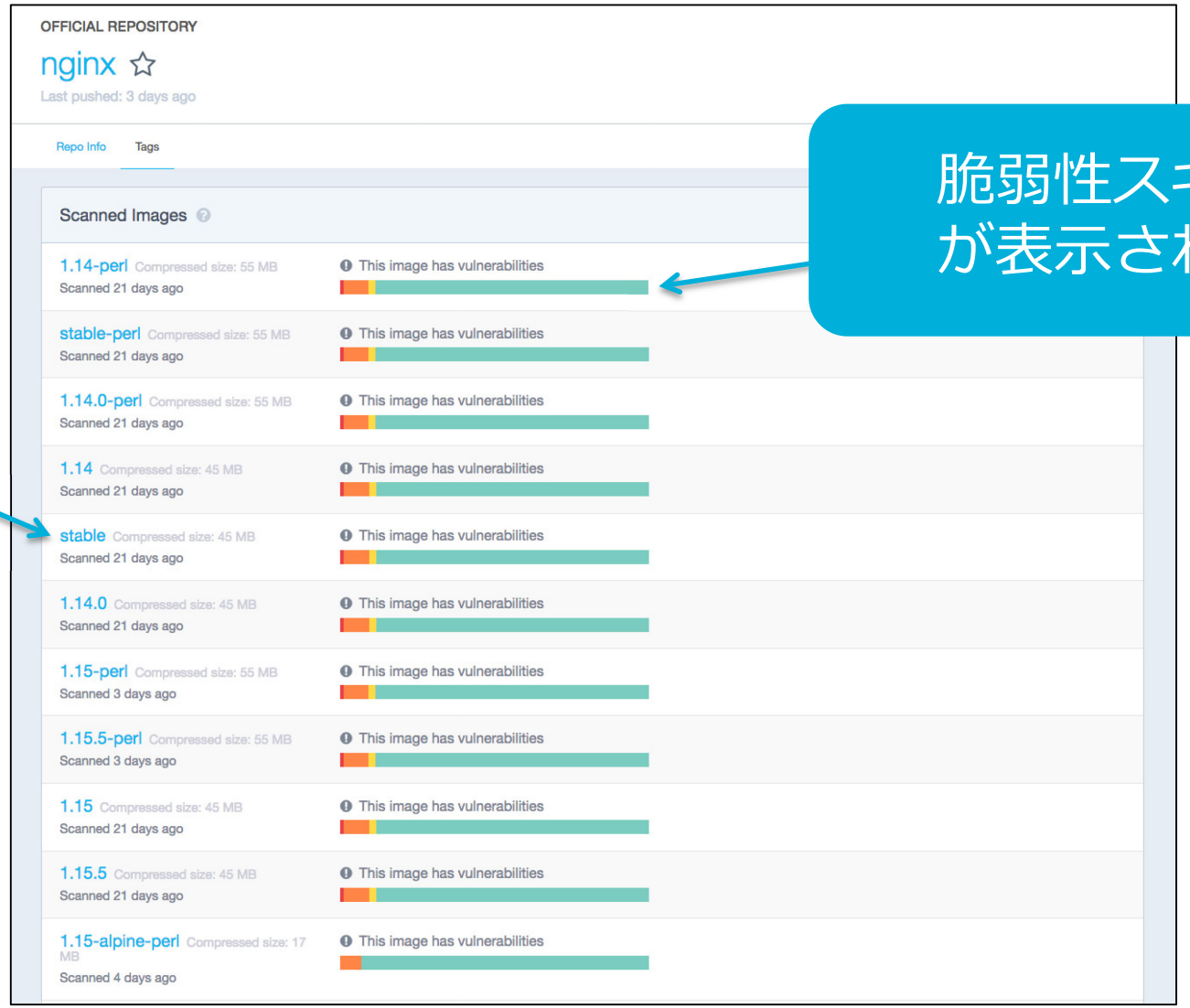
コンテナの利用方法

コンテナに実装されたAPIの説明があります。

1. 環境変数によるパラメータ設定
2. 永続ボリューム指定
3. ベースイメージとして使用する場合のDockerfileの書き方など

上に戻って、Tagsをクリックしてください。

Nginxのリポジトリに、タグの種類だけイメージが登録されいます。



詳細なレポートを参照できます。

脆弱性スキャンの結果が表示されています。

スグに利用できて役立つコンテナイメージ紹介

Nagios®

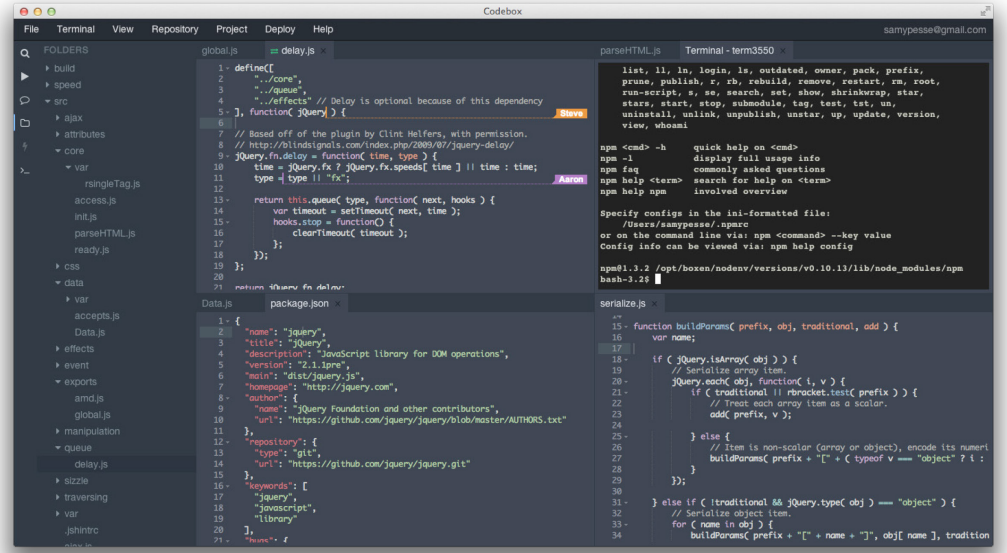
https://hub.docker.com/r/jasonrivers/nagios/

ownCloud

https://hub.docker.com/_/owncloud/

ROCKET.CHAT

https://hub.docker.com/_/rocket.chat/



https://hub.docker.com/r/tangym/codebox/

レッスン#4のまとめ

1. Docker Hubに無料で利用できるイメージが登録されている
2. DETAILSにコンテナの利用法が記載されている
3. 脆弱性スキャンの結果も参考にイメージを決定する

レッスン＃5

DockerHubの資料を読みながら
自分だけのウェブサーバーを起動しましょう。

実際に説明を読みながらNginxを起動して見ましょう。

https://hub.docker.com/r/_/nginx/

NGINX

How to use this image

Hosting some simple static content

```
$ docker run --name some-nginx -v /some/content:/usr/share/nginx/html:ro -d nginx
```

Alternatively, a simple `Dockerfile` can be used to generate a new image that includes the necessary content (which is a much cleaner solution than the bind mount above):

```
FROM nginx
COPY static-html-directory /usr/share/nginx/html
```

Place this file in the same directory as your directory of content ("static-html-directory"), run `docker build -t some-content-nginx .`, then start your container:

```
$ docker run --name some-nginx -d some-content-nginx
```

Exposing external port

```
$ docker run --name some-nginx -d -p 8080:80 some-content-nginx
```

Then you can hit `http://localhost:8080` or `http://host-ip:8080` in your browser.

これを選択して
実行します。

選択の決め手はオプションです。

-v が無い
　HTMLコンテンツが無くても
　NGINXを表示してくれる

-p が有る
　公開用ポートの設定がある

これまでの情報をもとに、Nginxにイメージからコンテナを起動するコマンドを組み立てます。

演習

```
docker run --name some-nginx -d -p 8080:80 some-content-nginx
```

コンテナ名
イメージの雛形から生成される
コンテナ (インスタンス) の名前

リポジトリ名とTagをセットします。
nginx:stableなどに置き換えてください。

バックグランドで実行にします。
画面に何も表示しません。
表示を見るには次のコマンドを利用します。

docker logs コンテナ名

コンテナ ホストのIPアドレスで
公開するためのポート番号です。
次のURLでアクセスできます。

http://仮想サーバーのIPアドレス:8080/

組み立てたコマンドを実行して、ブラウザでアクセスしましょう。

演習

コマンド組み立て例

```
docker run --name my-nginx -d -p 8080:80 nginx:stable
```

コマンド実行例

```
minion@iw01:~$ docker run --name my-nginx -d -p 8080:80 nginx:stable
Unable to find image 'nginx:stable' locally
stable: Pulling from library/nginx
f17d81b4b692: Pull complete
f1998c324db0: Pull complete
e27db20b7cd9: Pull complete
Digest: sha256:989faec2818af256f63bd303aac59a5fdf1e4aaf1d0e7b2c389db26daf59417d
Status: Downloaded newer image for nginx:stable
753e1bd87f1c258a06d556ccc3561afcd79b8eb8b3cba68f58307b8bb0f95ee9
minion@iw01:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx               stable              8552ee3809ee        About an hour ago   109MB
centos              7                   75835a67d134        4 weeks ago         200MB
minion@iw01:~$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                   NAMES
753e1bd87f1c        nginx:stable        "nginx -g 'daemon of…"   8 seconds ago       Up 8 seconds        0.0.0.0:8080->80/tcp    my-nginx
minion@iw01:~$
```

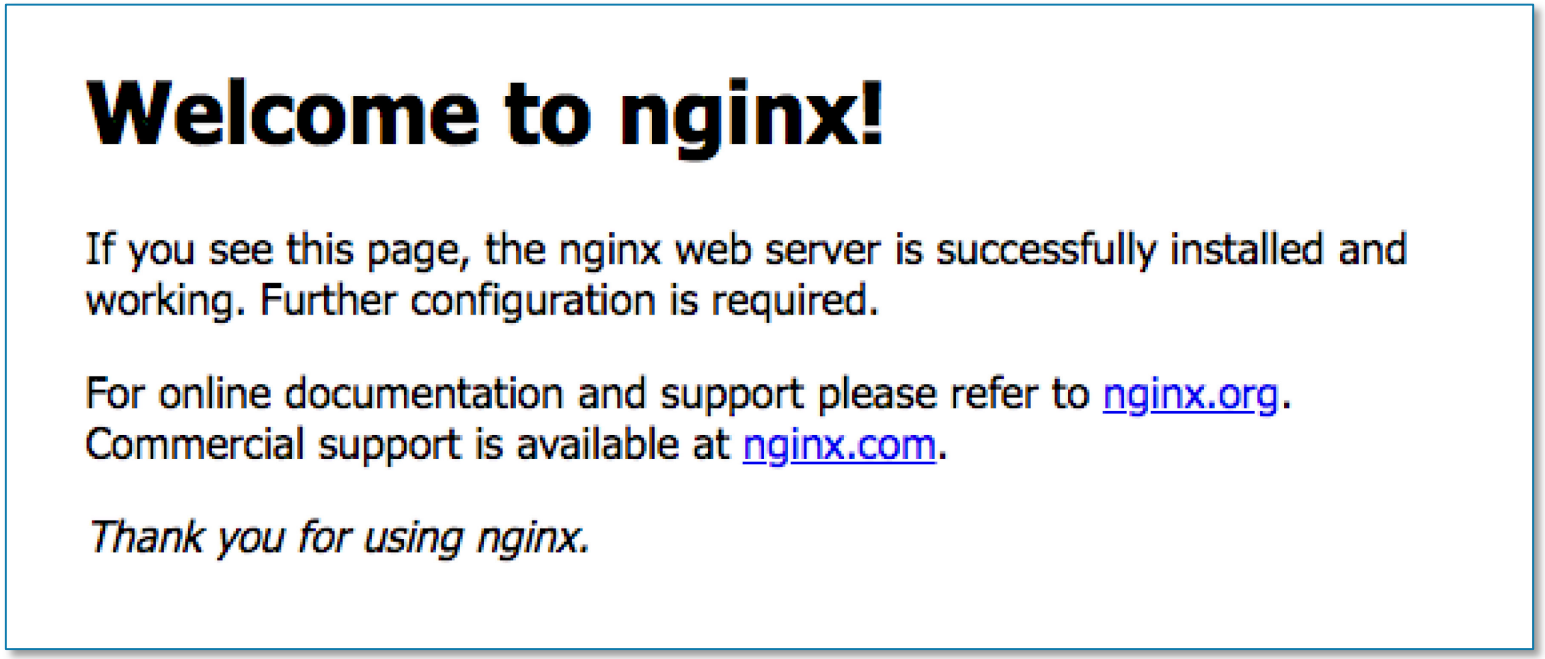
ブラウザで次の画面表示されたら成功です。

演習

アクセスするURL

http://仮想サーバーのIPアドレス:8080/

ブラウザ表示の期待結果



今度は、コンテンツを入れてNginxを起動して見ます。

演習

ディレクトリ html を作成して、コピペでindex.htmlを作成します。

```
minion@iw01:~$ mkdir html
minion@iw01:~$ cat - > html/index.html
```

```
<!DOCTYPE html>
<html lang="ja">
<head>
<meta charset="utf-8" />
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Docker & Kubernetes を運用の現場へ</h1>
<p>Docker & Kubernetes を運用の現場で使って行きましょう！</p>
<img src="https://www.docker.com/sites/default/files/vertical.png">
<img src="https://www.vectorlogo.zone/logos/kubernetes/kubernetes-card.png">
</body>
</html>
```

コンテンツの入ったコンテナ をもう一つ起動して見ます。

演習

再びコマンドを組み立てます。

```
docker run --name my-nginx2 -d -p 9080:80 -v `pwd`/html:/usr/share/nginx/html:ro nginx:stable
```

名前の重複は
許されないので
新たなコンテナ名
を付与します。

ポート番号も
２重利用できない
ので新たな番号
を指定します。

先ほど作った
ディレクトリを
指定します。

-d オプションの書き方

-p 公開ポートポート番号:コンテナ内ポート番号

-v オプションの書き方

-v ローカルディレクトリ:コンテナ内マウント先パス

コンテンツの入ったコンテナ をもう一つ起動して見ます。

演習

実行例

```
minion@iw01:~$ docker run --name my-nginx2 -d -p 9080:80 -v `pwd`/html:/usr/share/nginx/html:ro nginx:stable
3e4d002fdb02e720f4409a440de9bf9fb834b78e85210d5e121231da5ddbf1b2
minion@iw01:~$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
3e4d002fdb02        nginx:stable        "nginx -g 'daemon of…"   4 seconds ago       Up 4 seconds        0.0.0.0:9080->80/tcp     my-nginx2
753e1bd87f1c        nginx:stable        "nginx -g 'daemon of…"   About an hour ago   Up About an hour    0.0.0.0:8080->80/tcp     my-nginx
```

アクセスするURL

http://仮想サーバーのIPアドレス:9080/

ブラウザの表示画面

レッスン#5のまとめ

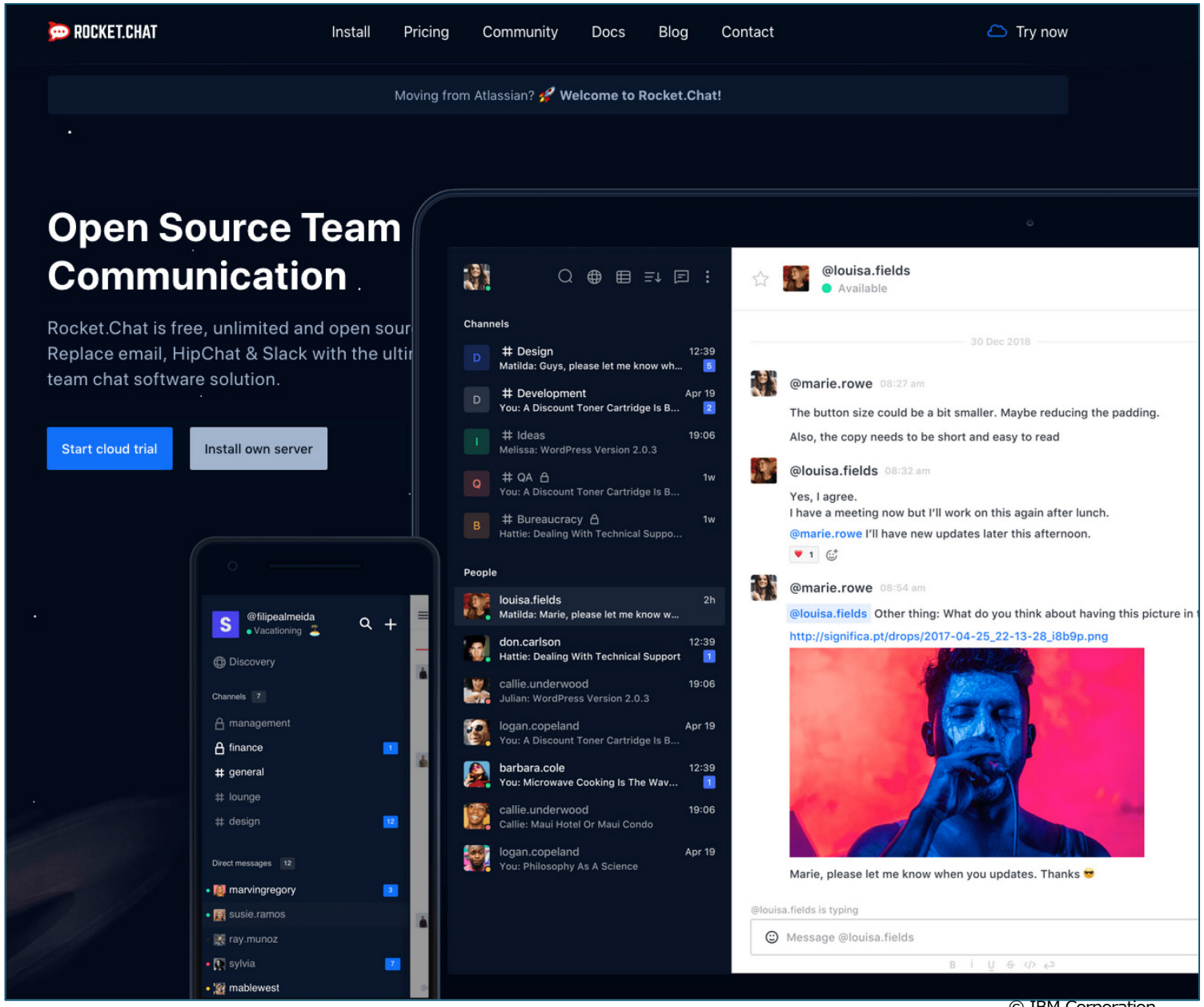
1. リポジトリの説明に従ってコンテナを起動する
2. コンテナ名とポートの重複を避けて、複数のコンテナを起動できる
3. オプション設定で、コンテナの動作を変えられる

レッスン＃6

Slackのようなアプリケーション
ROCKET.CHATを起動して見ましょう。

ROCKET.CHATとは

- オープンソースは無料で制限無しに使えるWeb Chatです。
- Mail、Slack や HipCahtに変わる究極のOSSのチャットソリューションとのこと。
- URL https://rocket.chat/



ROCKET.CHATのコンテナの詳細を読んで起動して見ましょう。

https://hub.docker.com/_/rocket.chat/

Rocket.Chat

Rocket.Chat is a Web Chat Server, developed in JavaScript, using the Meteor fullstack framework.

It is a great solution for communities and companies wanting to privately host their own chat service or for developers looking forward to build and evolve their own chat platforms.

ROCKET.CHAT

How to use this image

First, start an instance of mongo:

```
$ docker run --name db -d mongo:3.0 --smallfiles
```

Then start Rocket.Chat linked to this mongo instance:

```
$ docker run --name rocketchat --link db -d rocket.chat
```

This will start a Rocket.Chat instance listening on the default Meteor port of 3000 on the container.

If you'd like to be able to access the instance directly at standard port on the host machine:

```
$ docker run --name rocketchat -p 80:3000 --env ROOT_URL=http://localhost --link db -d r
```

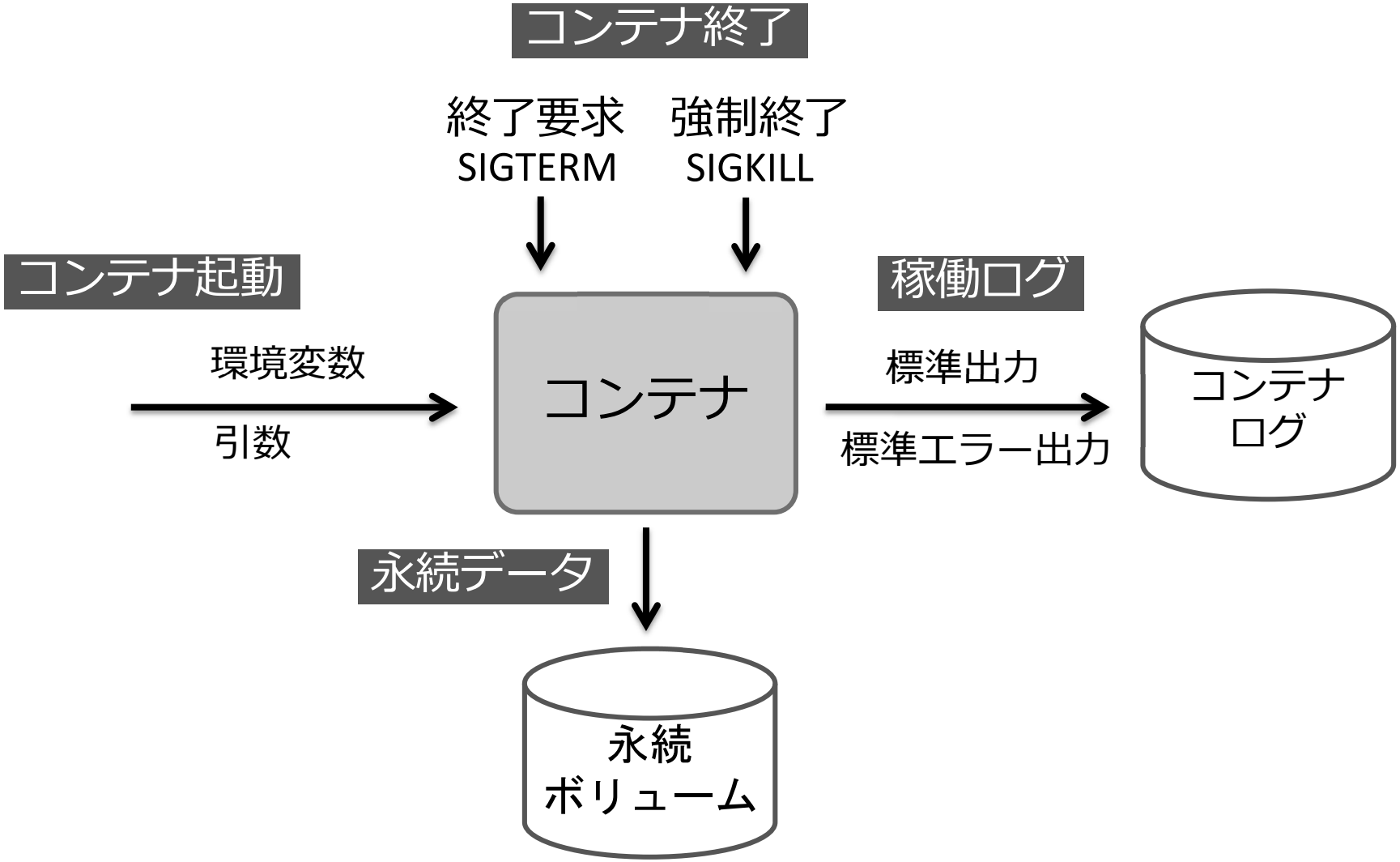
Then, access it via `http://localhost` in a browser. Replace `localhost` in `ROOT_URL` with your own domain name if you are hosting at your own domain.

If you're using a third party Mongo provider, or working with Kubernetes, you need to override the `MONGO_URL` environment variable:

```
$ docker run --name rocketchat -p 80:3000 --env ROOT_URL=http://localhost --env MONGO_UR
```

利用法の二つのコマンドで
ROCKET.CHATが起動します。

詳細にはコンテナに実装されたAPIが記述



２つのコマンドで、以下の構成を作成します。

演習

次の２つのコマンドを各自のIPアドレスに置き換えて、①②の順で、実行をお願いします。

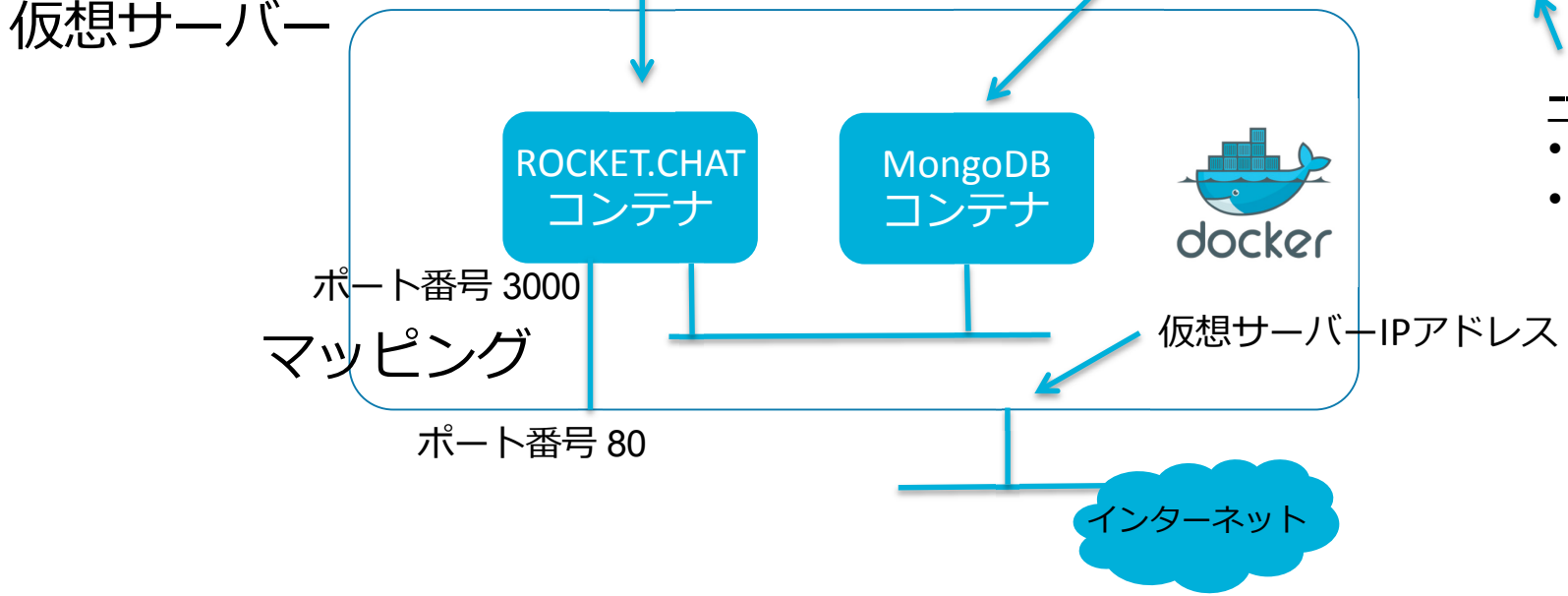
② ROCKET.CHATの起動

コンテナのプロセスへ環境変数をセットします。

```
docker run --name rocketchat -p 80:3000 --env ROOT_URL=http://仮想サーバーIPアドレス --link db -d rocket.chat
```

① MongoDBの起動

```
docker run --name db -d mongo:3.0 --smallfiles
```



--link db

- 内部ネットワークで接続二つを接続
- 名前解決を提供

コマンドの実行例

演習

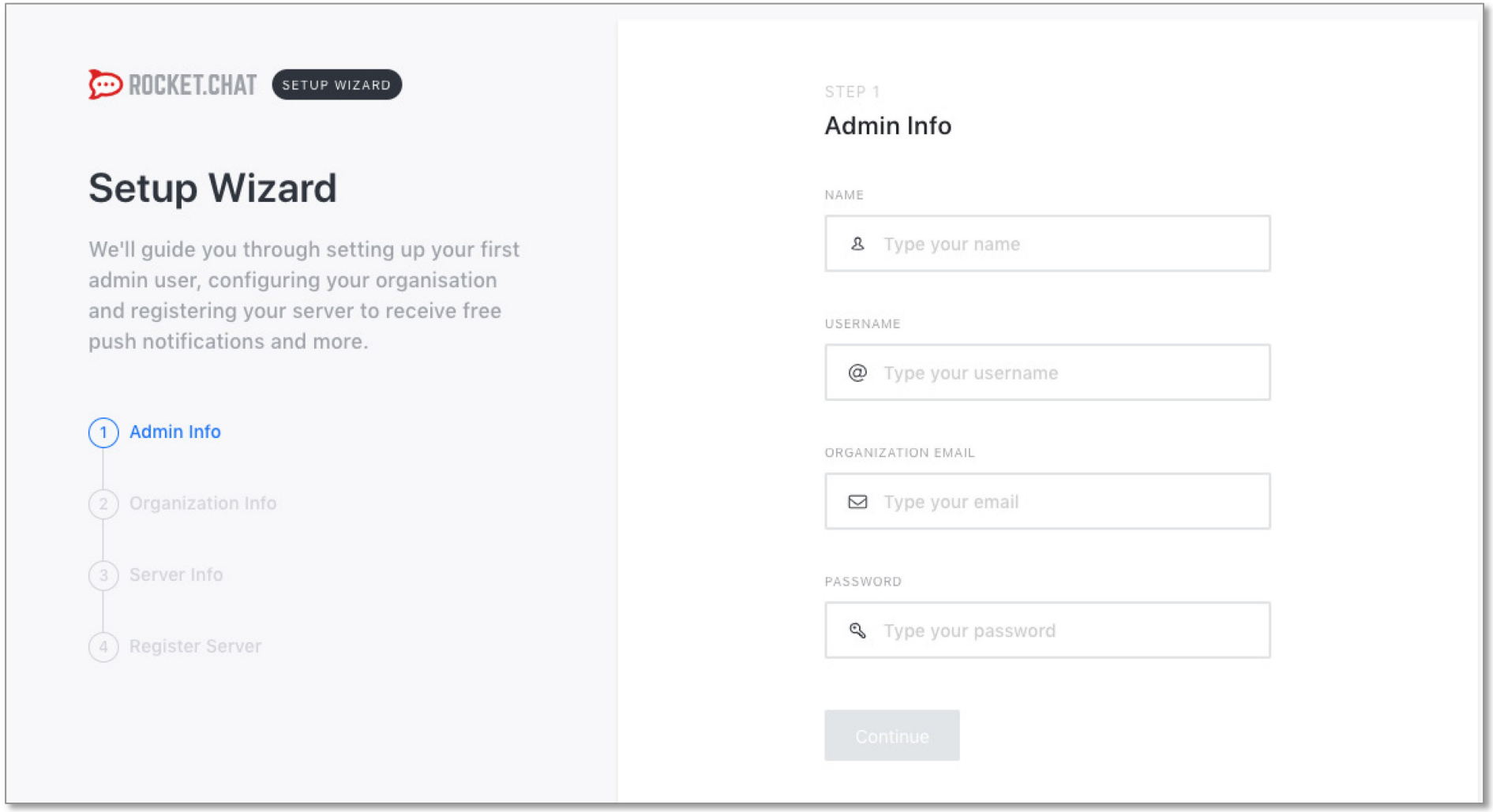
実行例

```
minion@iw01:~$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
3e4d002fdb02        nginx:stable        "nginx -g 'daemon of…"   22 hours ago        Up 22 hours         0.0.0.0:9080->80/tcp     my-nginx2
753e1bd87f1c        nginx:stable        "nginx -g 'daemon of…"   23 hours ago        Up 23 hours         0.0.0.0:8080->80/tcp     my-nginx
minion@iw01:~$ docker run --name db -d mongo:3.0 --smallfiles
Unable to find image 'mongo:3.0' locally
3.0: Pulling from library/mongo
9e518726c72a: Pull complete
5bec5585393f: Pull complete
67b43c55e1d0: Pull complete
ac73d29bb1af: Pull complete
5f055f855756: Pull complete
5ff164565c7e: Pull complete
14cac0308fdb: Pull complete
d64ca7fda324: Pull complete
a785a68ac06d: Pull complete
749ebc08962c: Pull complete
6941183827c4: Pull complete
4fe0aacd6337: Pull complete
Digest: sha256:2eae1f99d5e49286d5e34a153c6ffaf25b038731e1cfbd786712ced672e8e575
Status: Downloaded newer image for mongo:3.0
1486c98cdae51273125f1dbf315132564fde52bb21559b07f43c465ec6c1a3b0
minion@iw01:~$ docker run --name rocketchat -p 80:3000 --env ROOT_URL=http://[redacted] --link db -d rocket.chat
Unable to find image 'rocket.chat:latest' locally
latest: Pulling from library/rocket.chat
57936531d1ee: Pull complete
b186cf19f9ed: Pull complete
eadbf8312262: Pull complete
3f59c2d47106: Pull complete
96ed7d64abe1: Pull complete
b8c605c1dcf6: Pull complete
c374c906f2b0: Pull complete
67089d8324e0: Pull complete
Digest: sha256:ca5c48aeb4a6197c1b3a5ef9115bce1411a24bccaa5fcb08f490a7af8a65c18f
Status: Downloaded newer image for rocket.chat:latest
1b8701e71d2b1053baddbf923b6fc8caca444f90e14a86283213603a5dd8bedf
minion@iw01:~$
```

ROCKET.CHAT セットアップ画面1/6

演習

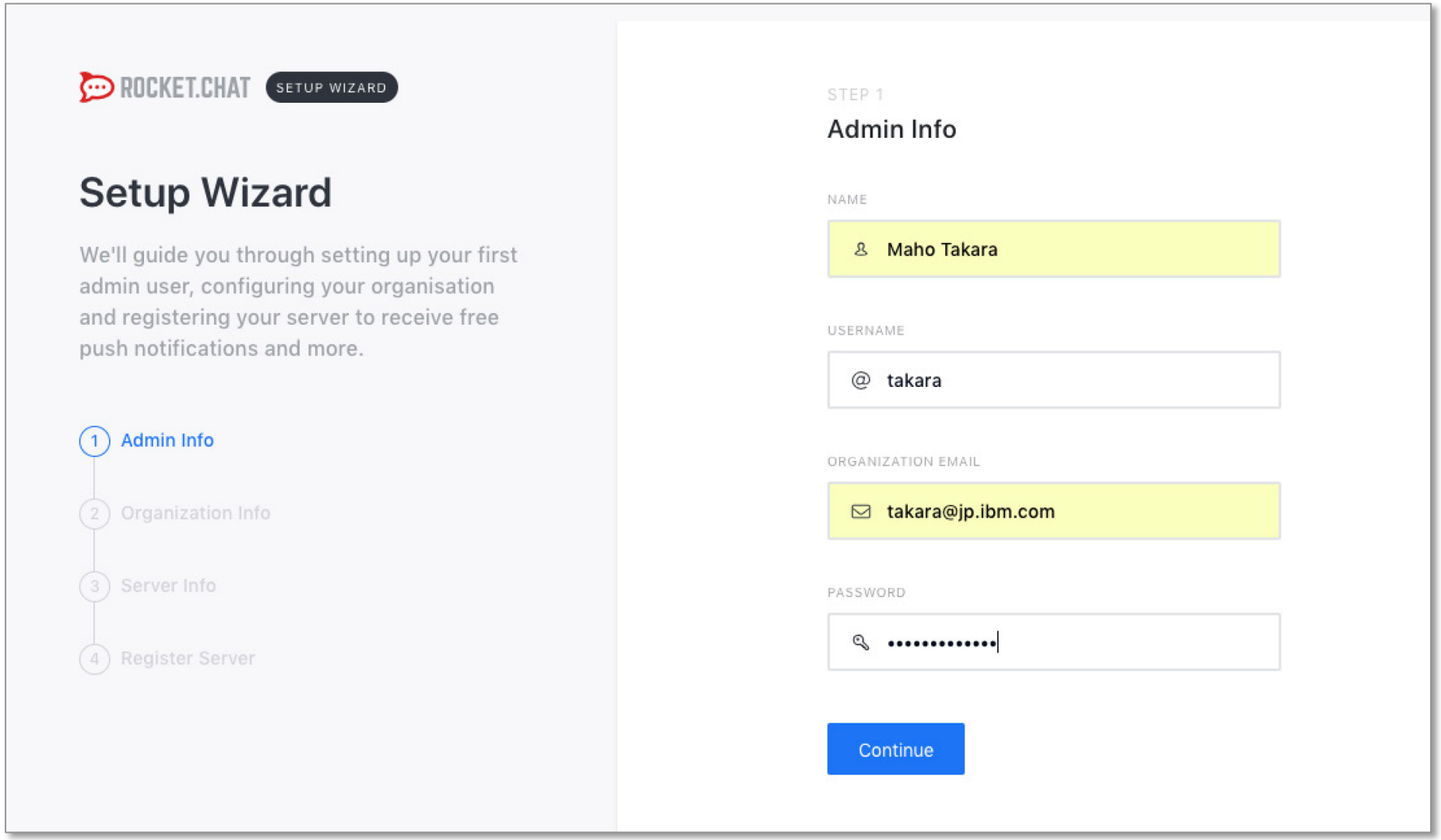
http://仮想サーバーIPアドレス/ でアクセス



ROCKET.CHAT セットアップ画面2/6

演習

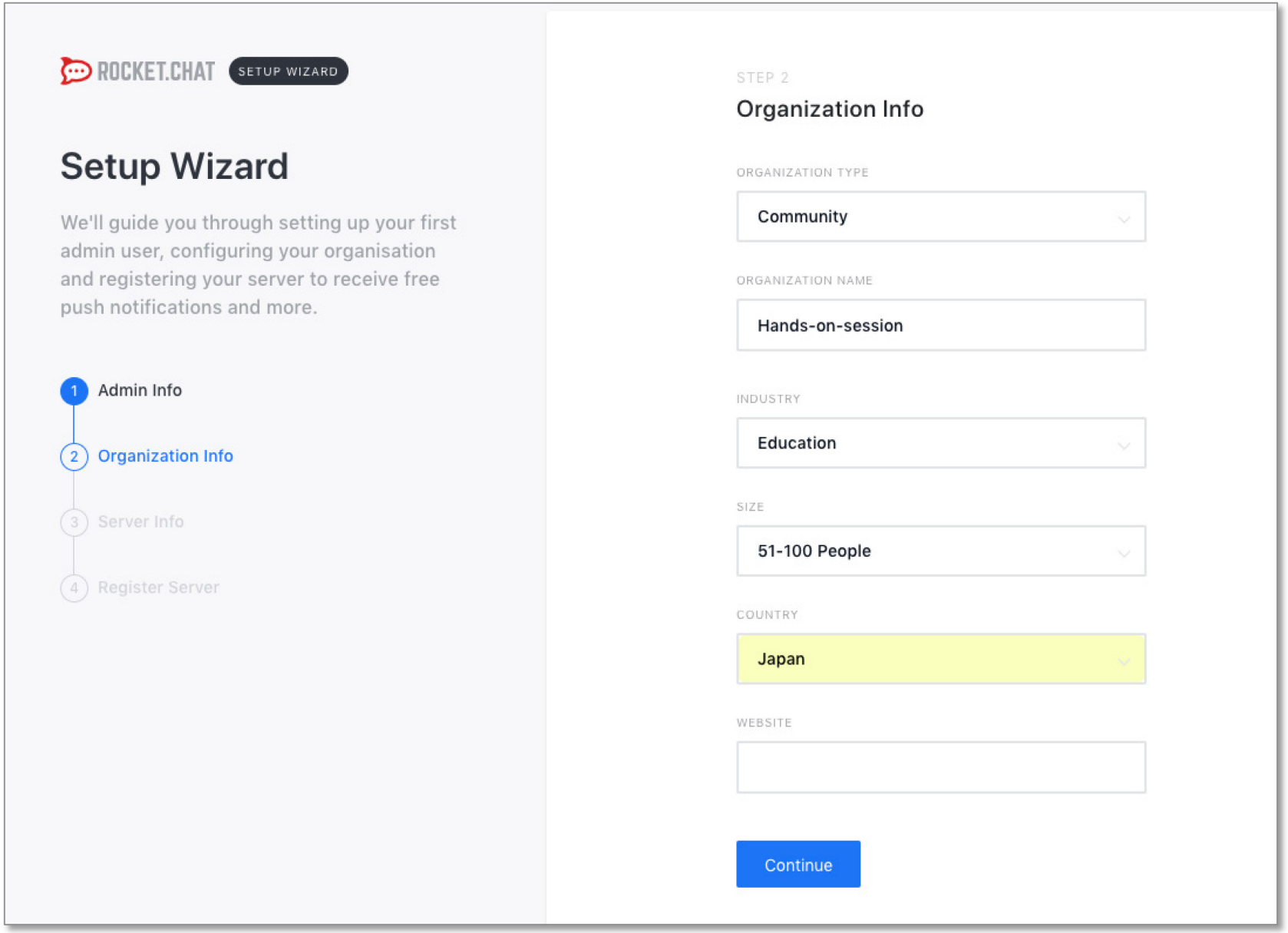
実行例



ROCKET.CHAT セットアップ画面3/6

演習

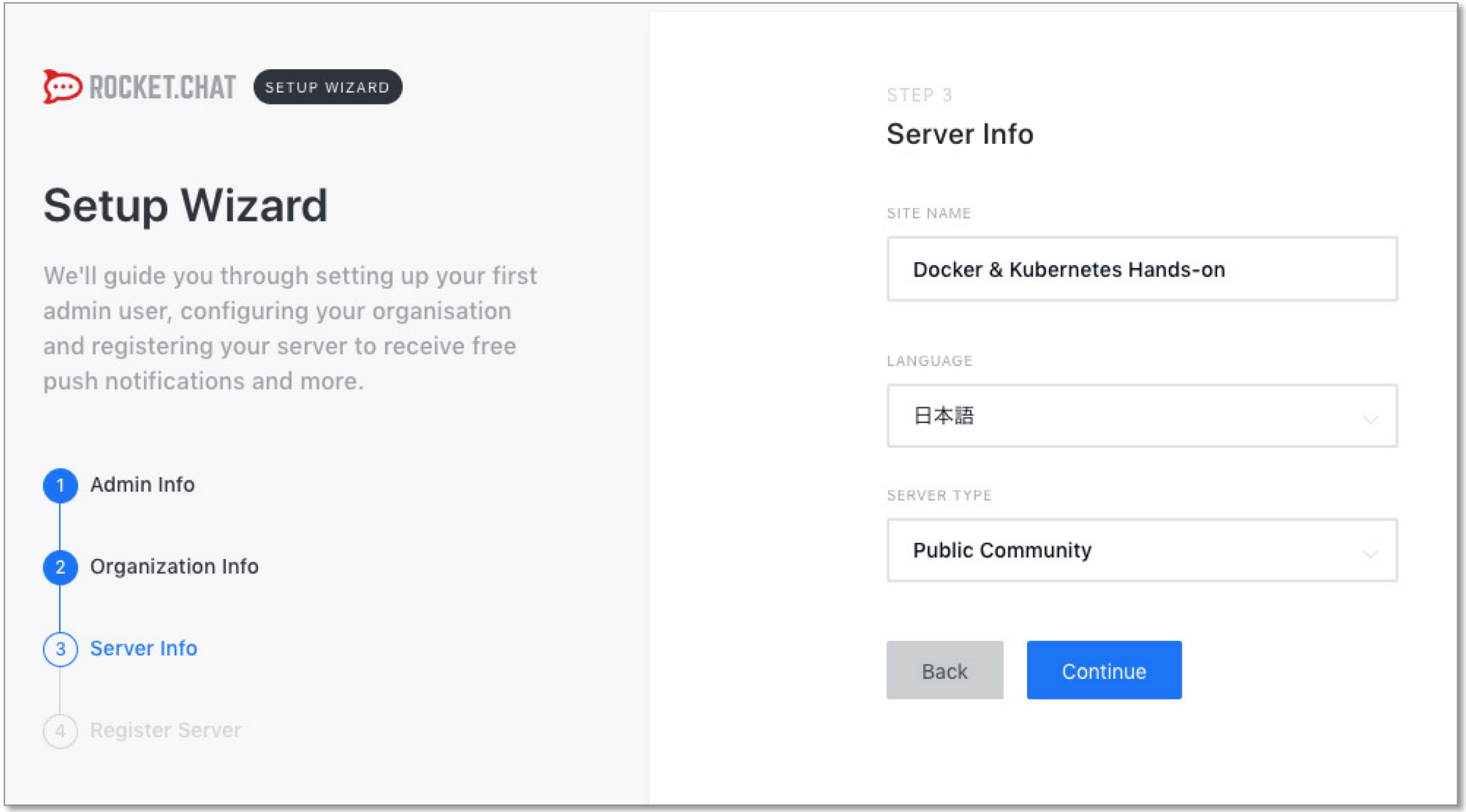
実行例



ROCKET.CHAT セットアップ画面4/6

演習

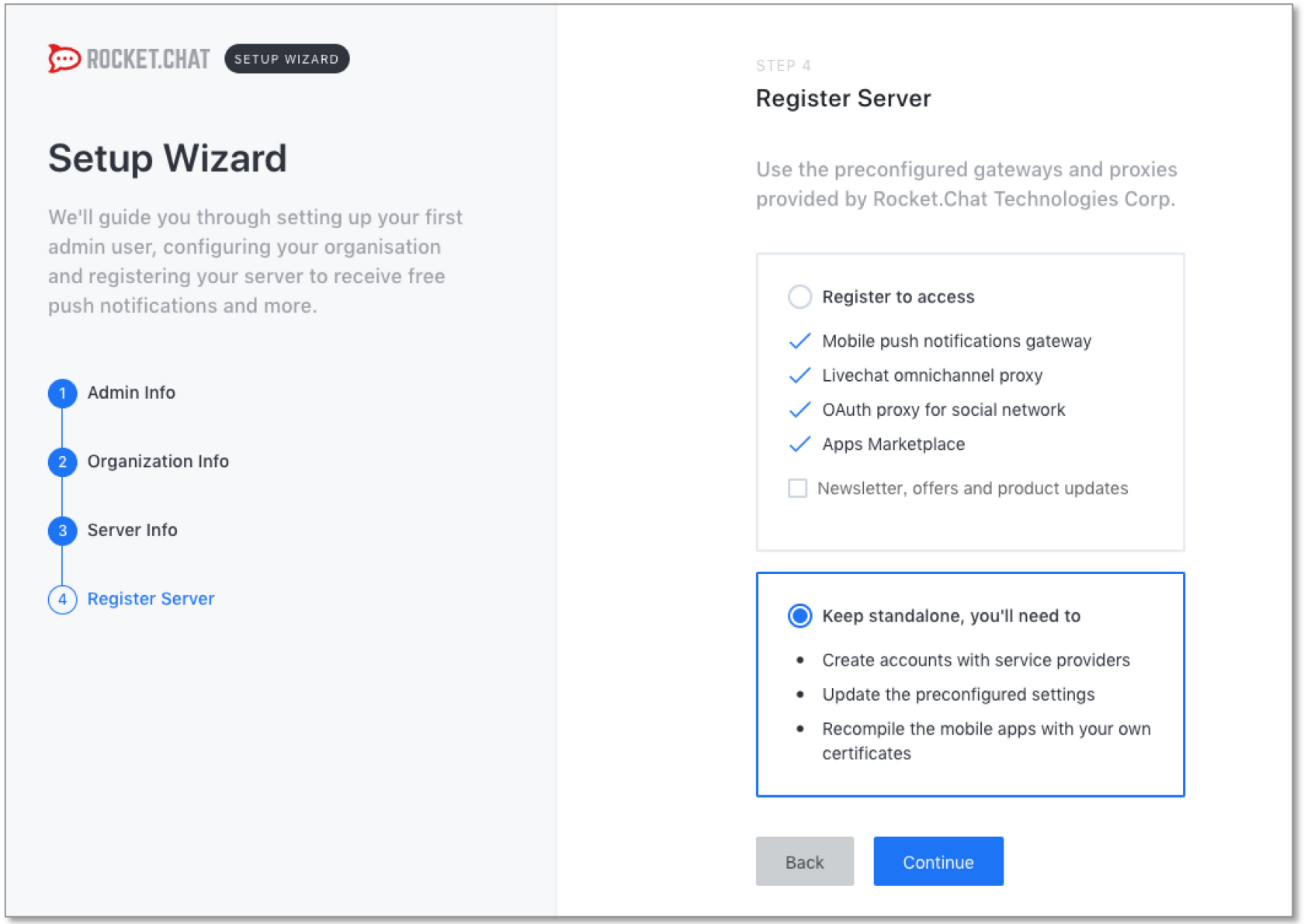
実行例



ROCKET.CHAT セットアップ画面5/6

演習

実行例



ROCKET.CHAT セットアップ画面6/6

演習

実行例

ROCKET.CHAT 利用画面

演習

実行例

レッスン#6のまとめ

1. アプリとDBのコンテナを繋いでシステムを構築
2. DockerHubに登録されたアプリケーションを即利用開始