

Internet Week 2018

Dockerハンズオン

2018年11月28日

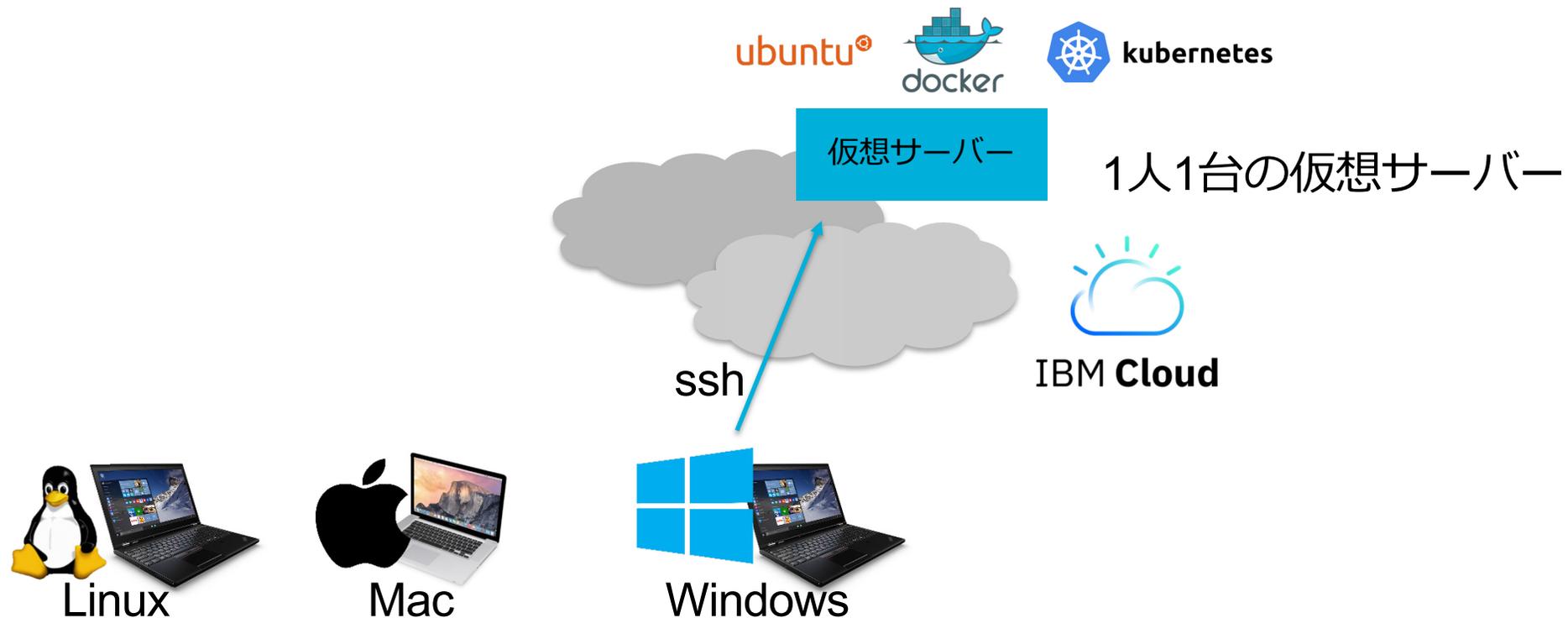
日本アイ・ビー・エム株式会社
クラウド事業本部 高良 真穂



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)

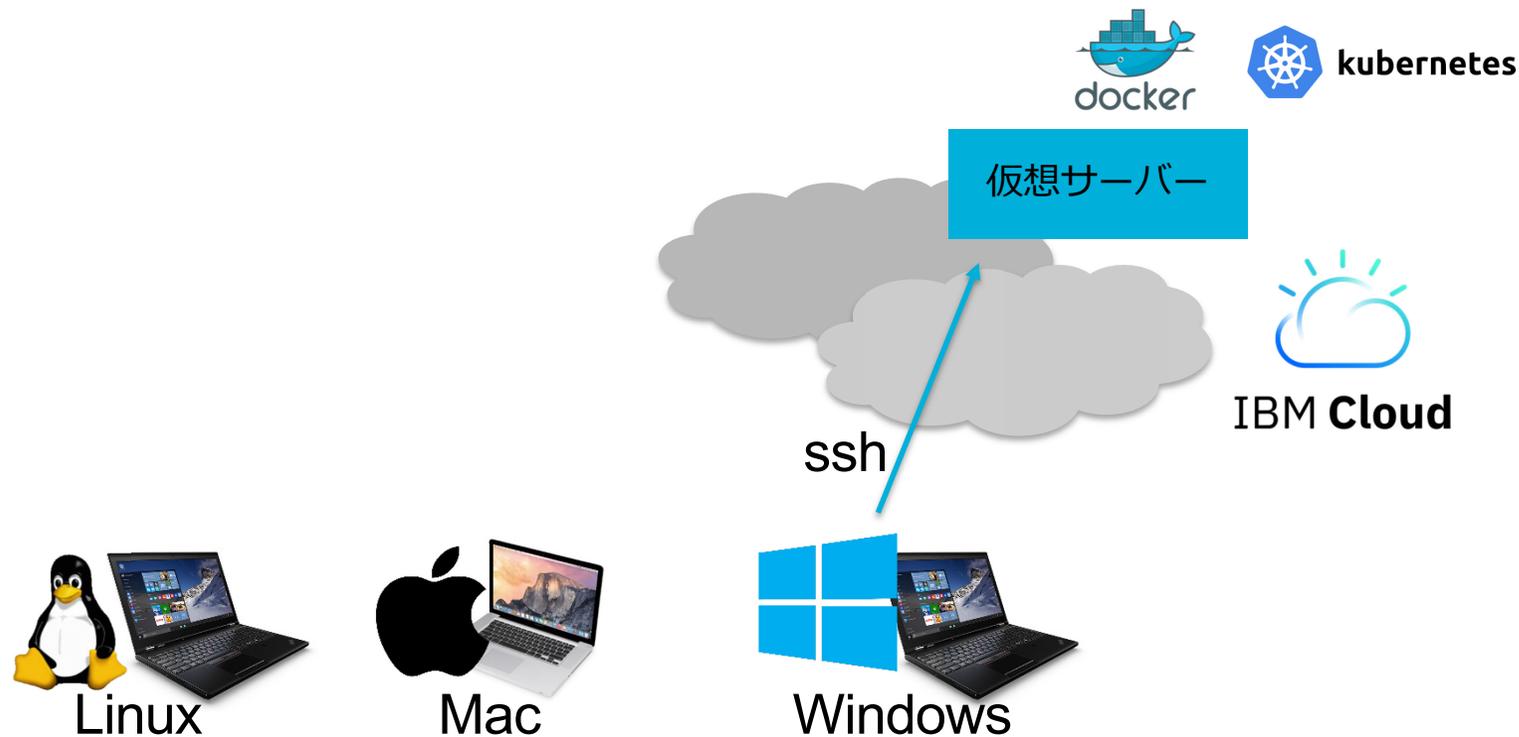
* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage
<以下省略>
```

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

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。

```
vagrant@vagrant-ubuntu-trusty-64:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
5b0f327be733: Pull complete
Digest: sha256:1f19634d26995c320618d94e6f29c09c6589d5df3c063287a00e6de8458f8242
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://cloud.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/engine/userguide/

vagrant@vagrant-ubuntu-trusty-64:~$
```

Dockerコマンドのメッセージを確認してください。

二つの種類のメッセージが表示されています。

```
vagrant@vagrant-ubuntu-trusty-64:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
5b0f327be733: Pull complete
Digest: sha256:1f19634d26995c320618d94e6f29c09c6589d5df3c063287a00e6de8458f8242
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://cloud.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/engine/userguide/

vagrant@vagrant-ubuntu-trusty-64:~$
```

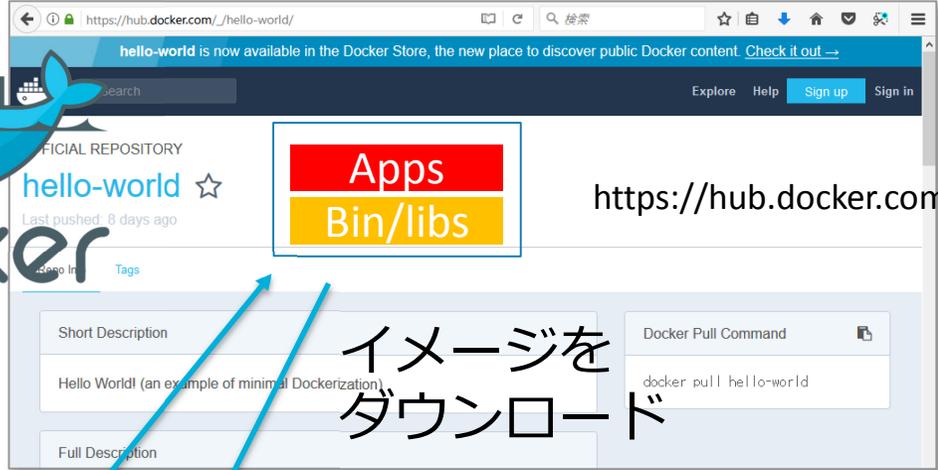
← コンテナの
取得

← コンテナの
プロセスから
の表示

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



hello-world is now available in the Docker Store, the new place to discover public Docker content. Check it out →

hello-world ★
Last pushed: 8 days ago

Apps
Bin/libs

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

Short Description
Hello World! (an example of minimal Dockerization)

Full Description

Imagesをダウンロード

Docker Pull Command
docker pull hello-world

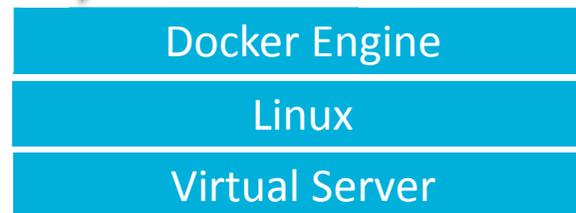
① コマンド実行

```
docker run hello-world
```



②

Apps
Bin/libs



④ コンテナを実行&メッセージ出力

```
Hello from Docker!  
This message shows that your installation appears to be working correctly.  
  
To generate this message, Docker took the following steps:  
1. The Docker client contacted the Docker daemon.  
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.  
3. The Docker daemon created a new container from that image which runs the  
   executable that produces the output you are currently reading.  
4. The Docker daemon streamed that output to the Docker client, which sent it  
   to your terminal.  
  
To try something more ambitious, you can run an Ubuntu container with:  
$ docker run -it ubuntu bash  
  
Share images, automate workflows, and more with a free Docker ID:  
https://cloud.docker.com/  
  
For more examples and ideas, visit:  
https://docs.docker.com/engine/userguide/
```

← 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

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

(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。

(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。



DockerHub
コンテナレジストリ

① コマンド実行

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

③ コンテナ内シェル

```
127.0.0.1:2222 - @971a53abc4a3:/ VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
vagrant@vagrant-ubuntu-trusty-64:~$ docker run -it centos:7 bash
[root@0971a53abc4a3 ~]# yum update
Loaded plugins: fastestmirror, ovl
base                               3.6 kB 00:00:00
extras                             3.4 kB 00:00:00
updates                            3.4 kB 00:00:00
(1/4): extras/7/x86_64/primary_db 112 kB 00:00:00
(2/4): base/7/x86_64/group_gz     156 kB 00:00:00
(3/4): base/7/x86_64/primary_db   5.7 MB 00:00:00
(4/4): updates/7/x86_64/primary_db 2.9 MB 00:00:07
Determining fastest mirrors
 * base: ftp.yz.yamagata-u.ac.jp
 * extras: ftp.yz.yamagata-u.ac.jp
 * updates: ftp.yz.yamagata-u.ac.jp
No packages marked for update
[root@0971a53abc4a3 ~]#
```

CentOSのコマンドが動作

② ローカルに存在しなければ
CentOS7イメージをダウンロード



Ubuntu 16.04の上で
CentOS7のコンテナ
が動作している



最初に作ったファイルは存在していますか？

(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。

(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。

(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。

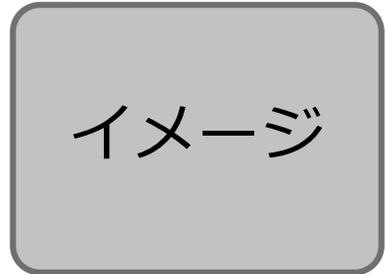
コンテナの状態遷移とコマンド



リモートリポジトリ DockerHub

リポジトリ

ローカルリポジトリ



テンプレート

`docker rmi`

コンテナ
イメージ破棄

`docker run`

`docker commit`

ランタイム



プロセス実行中

実行中コンテナリスト
`docker ps`

停止状態を含むコンテナリスト
`docker ps -a`

`docker kill`
`docker stop`
プロセス終了

`docker start`

停止状態



プロセス停止

`docker rm`

コンテナ破棄

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起動して、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)

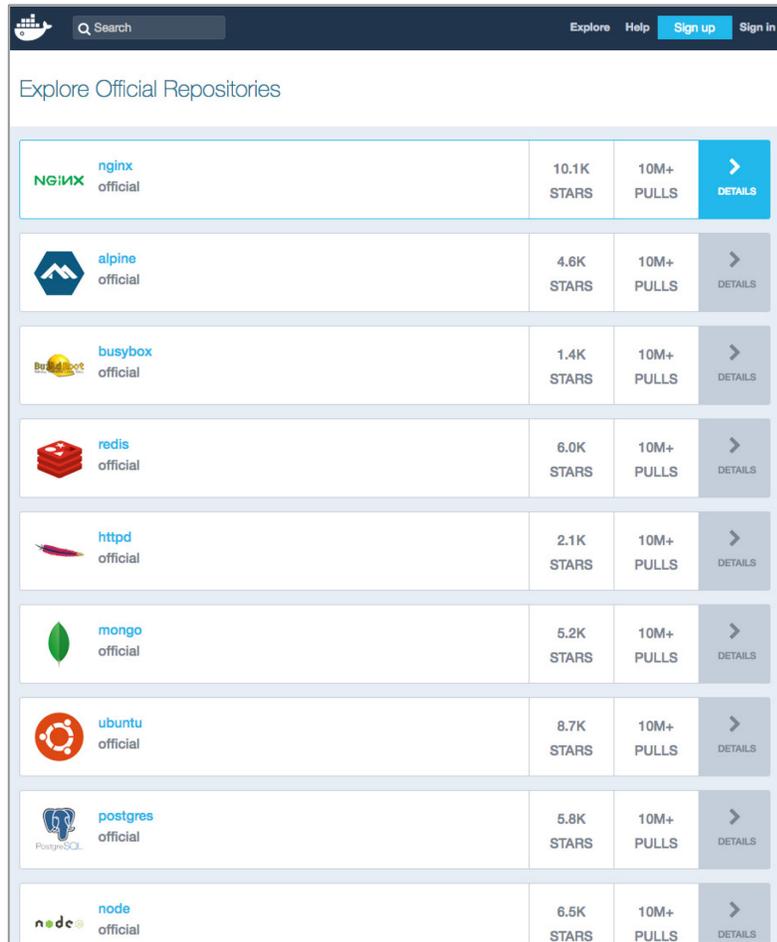


レッスン #4

Dockerレジストリを
探検してみましよう

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。

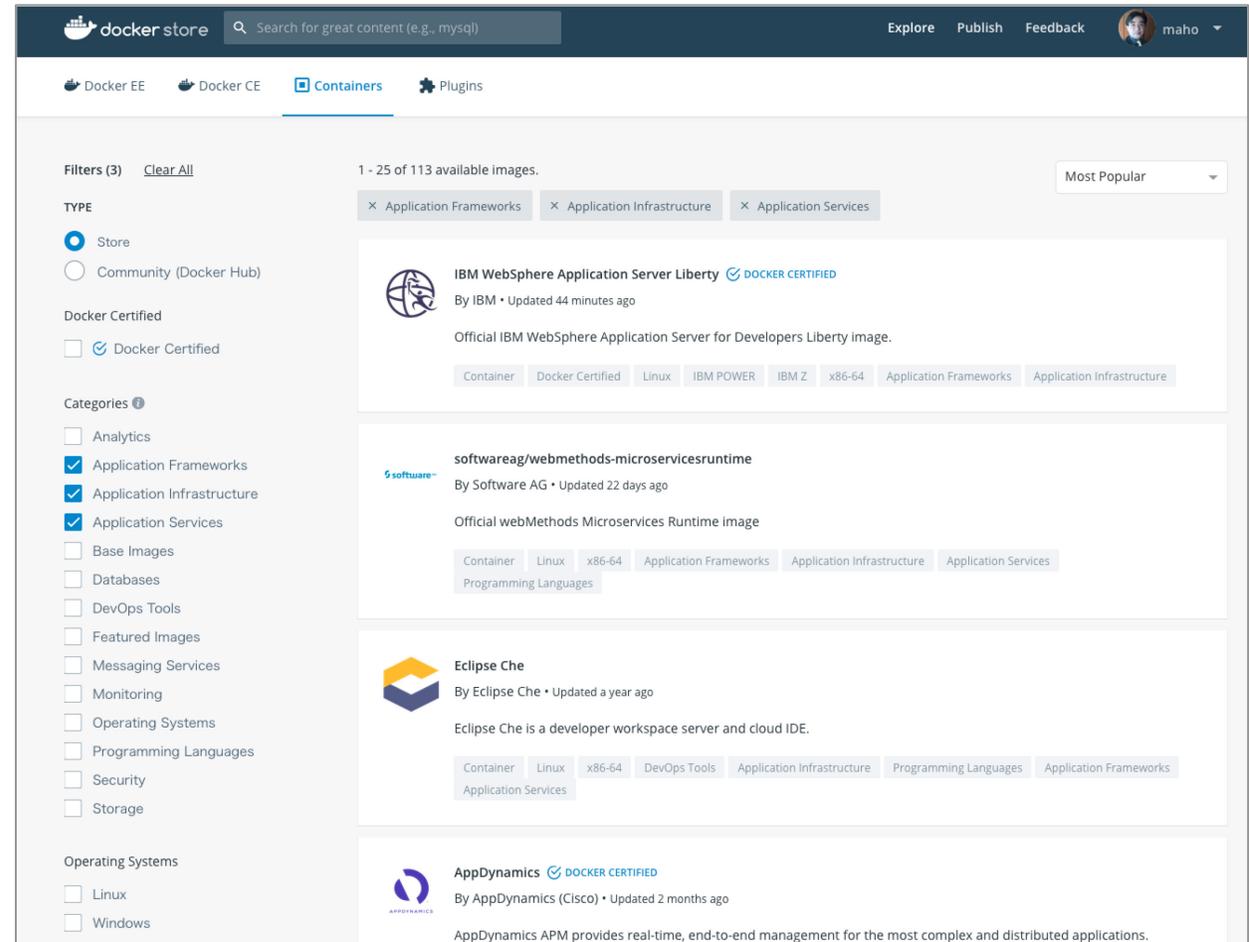
<https://hub.docker.com/explore/>



Explore Official Repositories

 nginx official	10.1K STARS	10M+ PULLS	> DETAILS
 alpine official	4.6K STARS	10M+ PULLS	> DETAILS
 busybox official	1.4K STARS	10M+ PULLS	> DETAILS
 redis official	6.0K STARS	10M+ PULLS	> DETAILS
 httpd official	2.1K STARS	10M+ PULLS	> DETAILS
 mongo official	5.2K STARS	10M+ PULLS	> DETAILS
 ubuntu official	8.7K STARS	10M+ PULLS	> DETAILS
 postgres official	5.8K STARS	10M+ PULLS	> DETAILS
 node official	6.5K STARS	10M+ PULLS	> DETAILS

<https://store.docker.com/>



docker store Search for great content (e.g., mysql)

Filters (3) Clear All

1 - 25 of 113 available images. Most Popular

TYPE

- Store
- Community (Docker Hub)

Docker Certified

- Docker Certified

Categories

- Analytics
- Application Frameworks
- Application Infrastructure
- Application Services
- Base Images
- Databases
- DevOps Tools
- Featured Images
- Messaging Services
- Monitoring
- Operating Systems
- Programming Languages
- Security
- Storage

Operating Systems

- Linux
- Windows

Application Frameworks Application Infrastructure Application Services

IBM WebSphere Application Server Liberty **DOCKER CERTIFIED**
By IBM • Updated 44 minutes ago
Official IBM WebSphere Application Server for Developers Liberty image.
Container Docker Certified Linux IBM POWER IBM Z x86-64 Application Frameworks Application Infrastructure

softwareag/webmethods-microservicesruntime
By Software AG • Updated 22 days ago
Official webMethods Microservices Runtime image
Container Linux x86-64 Application Frameworks Application Infrastructure Application Services Programming Languages

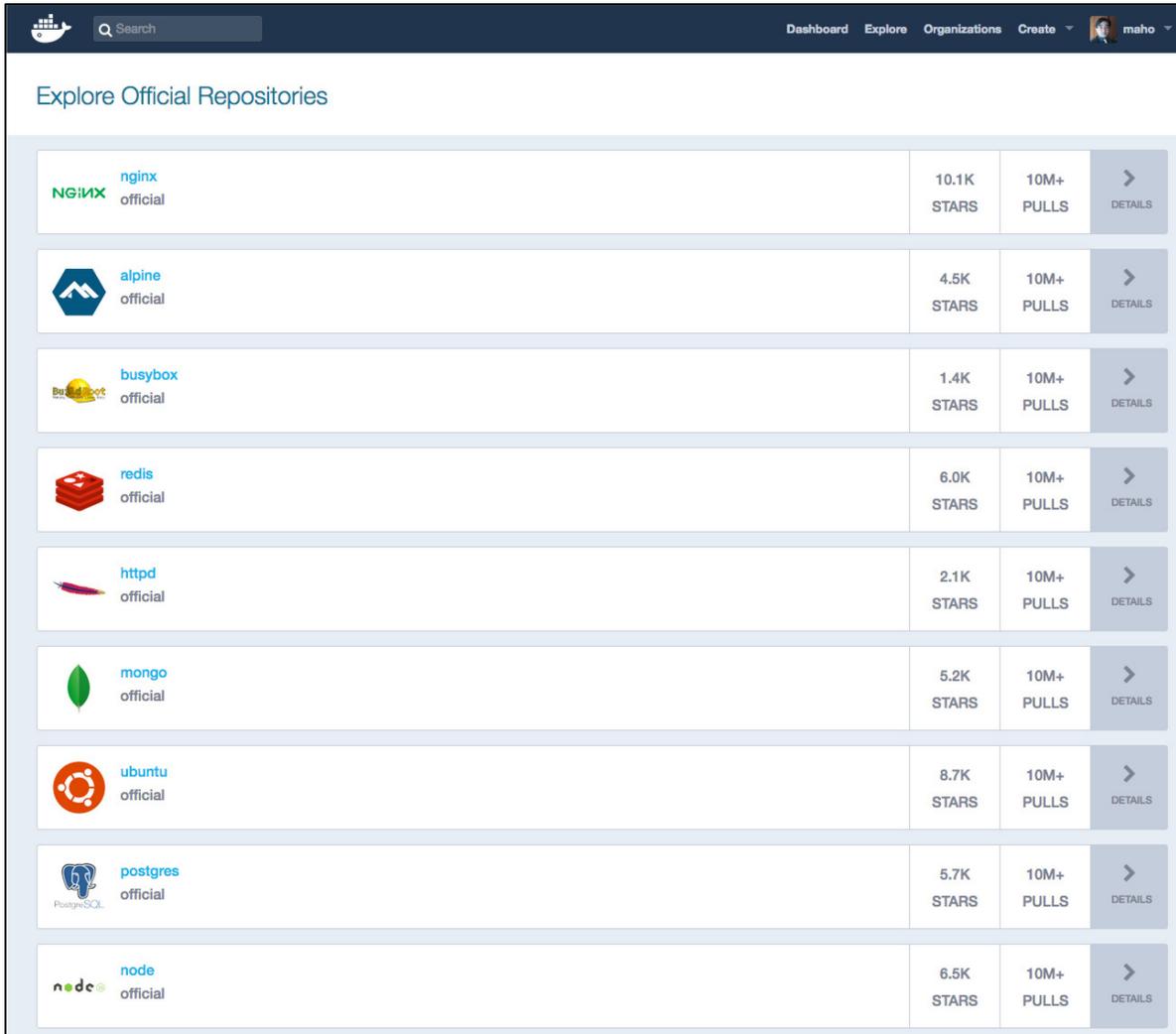
Eclipse Che
By Eclipse Che • Updated a year ago
Eclipse Che is a developer workspace server and cloud IDE.
Container Linux x86-64 DevOps Tools Application Infrastructure Programming Languages Application Frameworks Application Services

AppDynamics **DOCKER CERTIFIED**
By AppDynamics (Cisco) • Updated 2 months ago
AppDynamics APM provides real-time, end-to-end management for the most complex and distributed applications.

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います。



The screenshot shows the DockerHub 'Explore Official Repositories' page. The page lists several official Docker images, sorted by popularity. Each entry includes the repository name, its icon, the number of stars, the number of pulls, and a 'DETAILS' link.

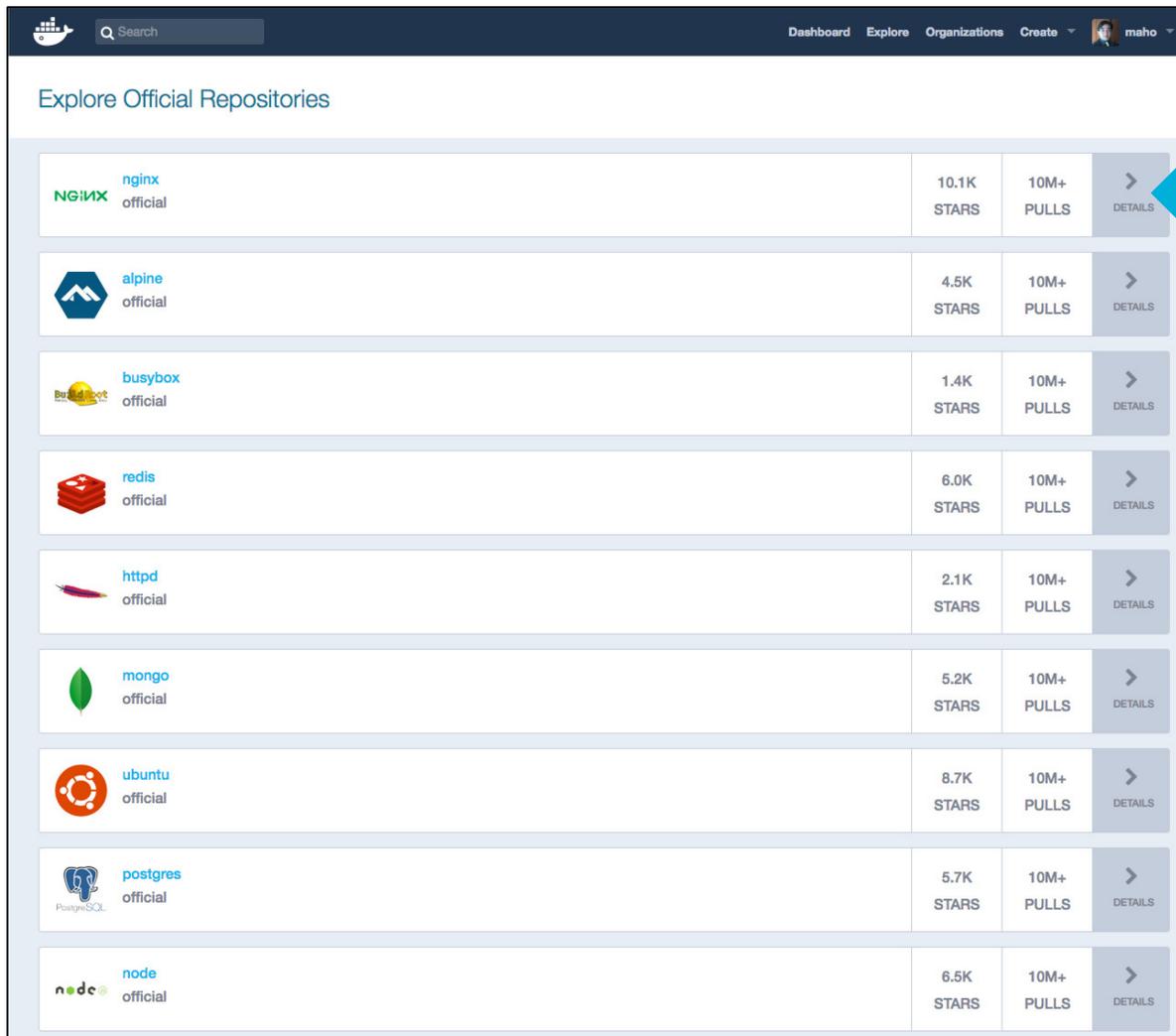
Repository Name	Stars	Pulls	Details
nginx official	10.1K	10M+	> DETAILS
alpine official	4.5K	10M+	> DETAILS
busybox official	1.4K	10M+	> DETAILS
redis official	6.0K	10M+	> DETAILS
httpd official	2.1K	10M+	> DETAILS
mongo official	5.2K	10M+	> DETAILS
ubuntu official	8.7K	10M+	> DETAILS
postgres official	5.7K	10M+	> DETAILS
node official	6.5K	10M+	> DETAILS

どのよう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。



Repository Name	Stars	Pulls	Action
nginx official	10.1K STARS	10M+ PULLS	DETAILS
alpine official	4.5K STARS	10M+ PULLS	DETAILS
busybox official	1.4K STARS	10M+ PULLS	DETAILS
redis official	6.0K STARS	10M+ PULLS	DETAILS
httpd official	2.1K STARS	10M+ PULLS	DETAILS
mongo official	5.2K STARS	10M+ PULLS	DETAILS
ubuntu official	8.7K STARS	10M+ PULLS	DETAILS
postgres official	5.7K STARS	10M+ PULLS	DETAILS
node official	6.5K STARS	10M+ PULLS	DETAILS

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

タグのリスト表示

OFFICIAL REPOSITORY
nginx ☆
Last pushed: 3 days ago

Repo Info Tags

Short Description
Official build of Nginx.

Docker Pull Command
docker pull nginx

Full Description

Supported tags and respective Dockerfile links

- 1.15.5, mainline, 1, 1.15, latest ([mainline/stretch/Dockerfile](#))
- 1.15.5-perl, mainline-perl, 1-perl, 1.15-perl, perl ([mainline/stretch-perl/Dockerfile](#))
- 1.15.5-alpine, mainline-alpine, 1-alpine, 1.15-alpine, alpine ([mainline/alpine/Dockerfile](#))
- 1.15.5-alpine-perl, mainline-alpine-perl, 1-alpine-perl, 1.15-alpine-perl, alpine-perl ([mainline/alpine-perl/Dockerfile](#))
- 1.14.0, stable, 1.14 ([stable/stretch/Dockerfile](#))
- 1.14.0-perl, stable-perl, 1.14-perl ([stable/stretch-perl/Dockerfile](#))
- 1.14.0-alpine, stable-alpine, 1.14-alpine ([stable/alpine/Dockerfile](#))
- 1.14.0-alpine-perl, stable-alpine-perl, 1.14-alpine-perl ([stable/alpine-perl/Dockerfile](#))

Quick reference

- **Where to get help:**
the Docker Community Forums, the Docker Community Slack, or Stack Overflow
- **Where to file issues:**
<https://github.com/nginxinc/docker-nginx/issues>
- **Maintained by:**
the NGINX Docker Maintainers
- **Supported architectures:** ([more info](#))
amd64, arm32v6, arm32v7, arm64v8, i386, ppc64le, s390x
- **Published image artifact details:**
[repo-info](#) [repo's](#) [repos/nginx/](#) [directory](#) ([history](#))
(image metadata, transfer size, etc)

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.conf
```

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ています。

詳細なレポート
を参照できます。

OFFICIAL REPOSITORY
nginx ☆
Last pushed: 3 days ago

Repo Info Tags

Scanned Images ?

1.14-perl	Compressed size: 55 MB Scanned 21 days ago	ⓘ This image has vulnerabilities
stable-perl	Compressed size: 55 MB Scanned 21 days ago	ⓘ This image has vulnerabilities
1.14.0-perl	Compressed size: 55 MB Scanned 21 days ago	ⓘ This image has vulnerabilities
1.14	Compressed size: 45 MB Scanned 21 days ago	ⓘ This image has vulnerabilities
stable	Compressed size: 45 MB Scanned 21 days ago	ⓘ This image has vulnerabilities
1.14.0	Compressed size: 45 MB Scanned 21 days ago	ⓘ This image has vulnerabilities
1.15-perl	Compressed size: 55 MB Scanned 3 days ago	ⓘ This image has vulnerabilities
1.15.5-perl	Compressed size: 55 MB Scanned 3 days ago	ⓘ This image has vulnerabilities
1.15	Compressed size: 45 MB Scanned 21 days ago	ⓘ This image has vulnerabilities
1.15.5	Compressed size: 45 MB Scanned 21 days ago	ⓘ This image has vulnerabilities
1.15-alpine-perl	Compressed size: 17 MB Scanned 4 days ago	ⓘ This image has vulnerabilities

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/>

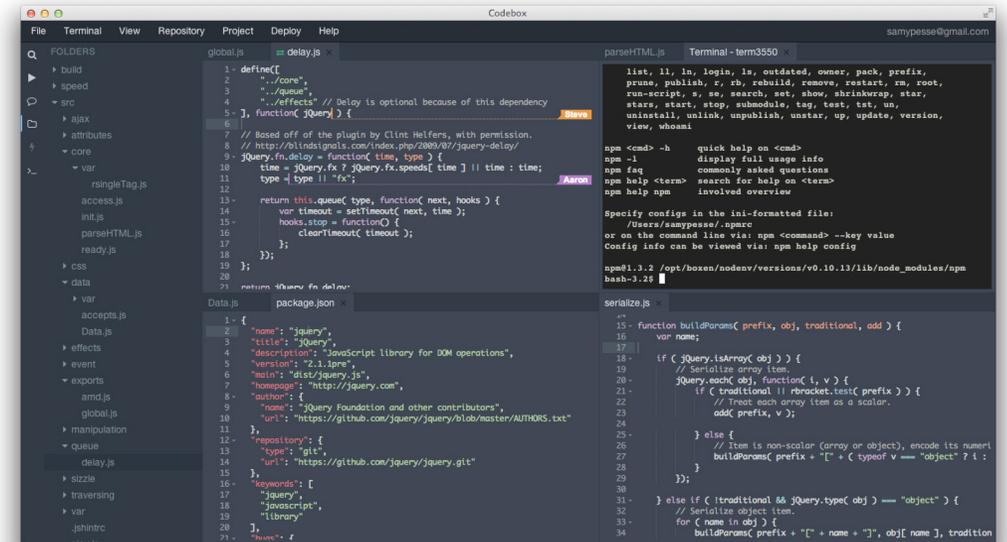


https://hub.docker.com/_/owncloud/



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。

```
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。

バックグラウ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。

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。

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

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

今回は、コンテンツを入れてNginxを起動して見ます。

ディレクトリ html を作成して、コピーで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>


</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。

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
3e4d002fdb02e720f4409a440de9bf9fb834b78e85210d5e121231da5ddb1b2
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 HipChat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/>

The image shows a composite of the Rocket.Chat website and its mobile application. The website header includes the Rocket.Chat logo and navigation links: Install, Pricing, Community, Docs, Blog, Contact, and a 'Try now' button. A banner below the header reads 'Moving from Atlassian? Welcome to Rocket.Chat!'. The main content area features the heading 'Open Source Team Communication' and a sub-headline: 'Rocket.Chat is free, unlimited and open source. Replace email, HipChat & Slack with the ultimate team chat software solution.' Below this are two buttons: 'Start cloud trial' and 'Install own server'. The mobile app interface is shown in the foreground, displaying a list of channels on the left, a list of people in the middle, and a chat conversation on the right. The chat conversation includes messages from @marie.rowe and @louisa.fields, with a link to a photo and a typing indicator at the bottom.

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.



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
```

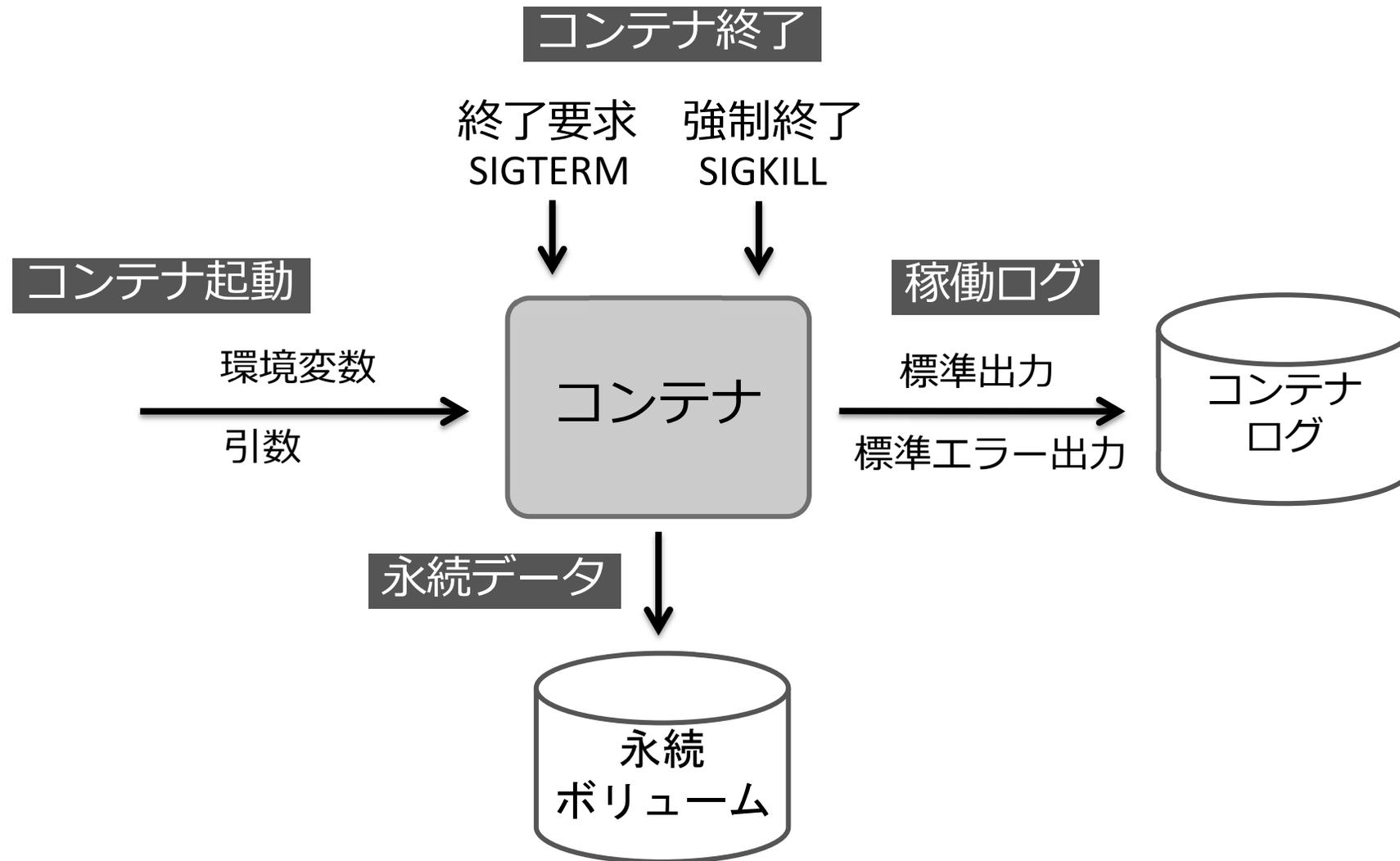
Then, access it via `http://localhost` in a browser. Replace `localhost` in `ROOT_URL` with your own domain name if you are hosting at your own domain.

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。

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

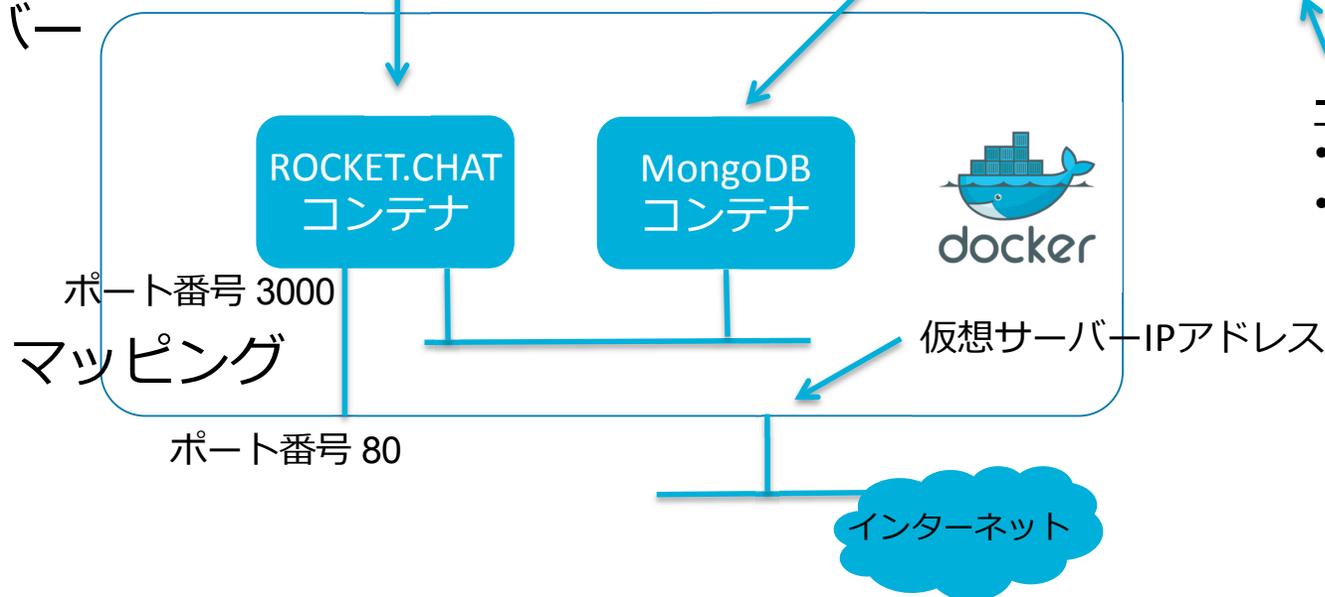
```
docker run --name rocketchat -p 80:3000 --env ROOT_URL=http://仮想サーバーIPアドレス --link db -d rocket.chat
```

コンテナのプロセスへ環境変数をセットします。

① MongoDBの起動

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

仮想サーバー



--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

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
1b8701e71d2b1053badbf923b6fc8caca444f90e14a86283213603a5dd8bedf
minion@iw01:~$
```

<http://仮想サーバーIPアドレス/> でアクセス

The screenshot shows the Rocket.Chat Setup Wizard interface. On the left, there is a sidebar with the Rocket.Chat logo and 'SETUP WIZARD' text. Below this, the title 'Setup Wizard' is displayed, followed by a descriptive paragraph: 'We'll guide you through setting up your first admin user, configuring your organisation and registering your server to receive free push notifications and more.' A vertical progress indicator shows four steps: 1 Admin Info (highlighted), 2 Organization Info, 3 Server Info, and 4 Register Server.

The main content area is titled 'STEP 1 Admin Info' and contains four input fields:

- NAME:** A text input field with a person icon and the placeholder text 'Type your name'.
- USERNAME:** A text input field with an '@' icon and the placeholder text 'Type your username'.
- ORGANIZATION EMAIL:** A text input field with an envelope icon and the placeholder text 'Type your email'.
- PASSWORD:** A text input field with a key icon and the placeholder text 'Type your password'.

At the bottom of the form is a 'Continue' button.

実行例

 **ROCKET.CHAT** **SETUP WIZARD**

Setup Wizard

We'll guide you through setting up your first admin user, configuring your organisation and registering your server to receive free push notifications and more.

- 1 Admin Info**
- 2 Organization Info
- 3 Server Info
- 4 Register Server

STEP 1
Admin Info

NAME

USERNAME

ORGANIZATION EMAIL

PASSWORD

実行例

 **ROCKET.CHAT** SETUP WIZARD

Setup Wizard

We'll guide you through setting up your first admin user, configuring your organisation and registering your server to receive free push notifications and more.

- Admin Info
- Organization Info**
- Server Info
- Register Server

STEP 2
Organization Info

ORGANIZATION TYPE

ORGANIZATION NAME

INDUSTRY

SIZE

COUNTRY

WEBSITE

実行例

 **ROCKET.CHAT** SETUP WIZARD

Setup Wizard

We'll guide you through setting up your first admin user, configuring your organisation and registering your server to receive free push notifications and more.

- 1 Admin Info
- 2 Organization Info
- 3 **Server Info**
- 4 Register Server

STEP 3

Server Info

SITE NAME

LANGUAGE

SERVER TYPE

実行例

 **ROCKET.CHAT** SETUP WIZARD

Setup Wizard

We'll guide you through setting up your first admin user, configuring your organisation and registering your server to receive free push notifications and more.

- 1 Admin Info
- 2 Organization Info
- 3 Server Info
- 4 Register Server

STEP 4

Register Server

Use the preconfigured gateways and proxies provided by Rocket.Chat Technologies Corp.

Register to access

- ✓ Mobile push notifications gateway
- ✓ Livechat omnichannel proxy
- ✓ OAuth proxy for social network
- ✓ Apps Marketplace

Newsletter, offers and product updates

Keep standalone, you'll need to

- Create accounts with service providers
- Update the preconfigured settings
- Recompile the mobile apps with your own certificates

Back Continue

実行例



実行例

The screenshot displays the Rocket.Chat interface for a channel named "# general". The left sidebar shows the channel list with "# general" selected. The main area shows a message history starting with a system message "会話を開始しました" (Started conversation) dated "2018年11月8日". Three messages follow: "takara" (Admin) at 9:14 AM saying "チャンネルへ参加しました。" (Joined channel.), "takara" (Admin) at 9:19 AM saying "こんにちは" (Hello), and "syota" at 9:23 AM saying "チャンネルへ参加しました。" (Joined channel.). A "ユーザーを追加" (Add user) dialog is open on the right, showing a search box with the placeholder "ユーザー名を入力してください..." (Please enter the user name...). The dialog has "キャンセル" (Cancel) and "+ ユーザーを追加" (+ Add user) buttons. The bottom of the interface shows a message input field with a microphone icon and a plus sign for more options.

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

