Ubuntuを追加

```
anai@DESKTOP-CT2JHOV MINGW64 /c/Program Files/Docker Toolbox
$ docker run -it ubuntu bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
5c939e3a4d10: Pull complete
c63719cdbe7a: Pull complete
19a861ea6baf: Pull complete
651c9d2d6c4f: Pull complete
Digest: sha256:8d31dad0c58f552e890d68bbfb735588b6b820a46e459672d96e585871acc110
Status: Downloaded newer image for ubuntu:latest
root@9107f2aa8d87:/# pwd
/
root@9107f2aa8d87:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
```

docker run -it ubuntu bash	
でubuntuのイメージをダウンロードしてコンテナ作成、実行までやってくれる	
run	イメージをダウンロードしてコンテナ作成、実行までやってくれる
-it	bashでubutuを開く　という意味？

```
root@9107f2aa8d87:/# exit
exit

anai@DESKTOP-CT2JHOV MINGW64 /c/Program Files/Docker Toolbox
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS

anai@DESKTOP-CT2JHOV MINGW64 /c/Program Files/Docker Toolbox
$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                          PORTS
9107f2aa8d87        ubuntu              "bash"              7 minutes ago       Exited (0) 10 seconds ago
a2cb9f6ef58a        python:3.6          "/bin/bash"         3 months ago        Exited (0) 3 months ago
924f74ebd6fe        hello-world         "/hello"            3 months ago        Exited (0) 3 months ago

anai@DESKTOP-CT2JHOV MINGW64 /c/Program Files/Docker Toolbox
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              latest              ccc6e87d482b        3 weeks ago         64.2MB
anai2019/ya         latest              7f6888766378        3 months ago        1.51GB
ya                  latest              7f6888766378        3 months ago        1.51GB
python              3.6                 a2e9f0fba405        3 months ago        913MB
hello-world         latest              fce289e99eb9        13 months ago       1.84kB
```

実行していたのはexitで出れる。
この場合、実行中の作業内容はすべて消えるらしい

コンテナとイメージは残っている

MYSQLを追加①

```
anai@DESKTOP-CT2JHOV MINGW64 /c/Program Files/Docker Toolbox
$ docker pull mysql
Using default tag: latest
latest: Pulling from library/mysql
619014d83c02: Pull complete
9ced578c3a5f: Pull complete
731f6e13d8ea: Pull complete
3c183de42679: Pull complete
6de69b5c2f3c: Pull complete
00f0a4086406: Pull complete
84d93aea836d: Pull complete
f18efbfd8d76: Pull complete
012b302865d1: Pull complete
fe16fd240f59: Pull complete
ca3e793e545e: Pull complete
51d0f2cb2610: Pull complete
Digest: sha256:6d0741319b6a2ae22c384a97f4bbee411b01e75f6284af0cce339fee83d7e314
Status: Downloaded newer image for mysql:latest
docker.io/library/mysql:latest

anai@DESKTOP-CT2JHOV MINGW64 /c/Program Files/Docker Toolbox
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
mysql               latest              791b6e40940c        8 days ago          465MB
ubuntu              latest              ccc6e87d482b        3 weeks ago         64.2MB
anai2019/ya         latest              7f6888766378        3 months ago        1.51GB
ya                  latest              7f6888766378        3 months ago        1.51GB
python              3.6                 a2e9f0fba405        3 months ago        913MB
hello-world         latest              fce289e99eb9        13 months ago       1.84kB

anai@DESKTOP-CT2JHOV MINGW64 /c/Program Files/Docker Toolbox
$ docker run --name mysql -e MYSQL_ROOT_PASSWORD=mysql -d -p 3306:3306 mysql
1979c273465320bb7b08f34b9a905b2399484229af6456aee34c5600acccab60

anai@DESKTOP-CT2JHOV MINGW64 /c/Program Files/Docker Toolbox
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                               NAMES
1979c2734653        mysql               "docker-entrypoint.s…"   28 seconds ago      Up 27 seconds       0.0.0.0:3306->3306/tcp, 33060/tcp   mysql
```

```
anai@DESKTOP-CT2JHOV MINGW64 /c/Program Files/Docker Toolbox
$ docker exec -it 1979c2734653 bash
root@1979c2734653:/# maysql -u root -p
bash: maysql: command not found
root@1979c2734653:/# _
```

```
root@1979c2734653:/# exit
exit

anai@DESKTOP-CT2JHOV MINGW64 /c/Program Files/Docker Toolbox
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                               NAMES
1979c2734653        mysql               "docker-entrypoint.s…"   5 minutes ago       Up 5 minutes        0.0.0.0:3306->3306/tcp, 33060/tcp   mysql

anai@DESKTOP-CT2JHOV MINGW64 /c/Program Files/Docker Toolbox
```

MYSQLを追加②

```
anai@DESKTOP-CT2JHOV MINGW64 /c/Program Files/Docker Toolbox
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                               NAMES
1979c2734653        mysql               "docker-entrypoint.s…"   11 minutes ago      Up 11 minutes       0.0.0.0:3306->3306/tcp, 33060/tcp   mysql

anai@DESKTOP-CT2JHOV MINGW64 /c/Program Files/Docker Toolbox
$ docker rm 1979c2734653
Error response from daemon: You cannot remove a running container 1979c273465320bb7b08f34b9a905b2399484229af6456aee34c5600acccab60. Stop the container before
 attempting removal or force remove

anai@DESKTOP-CT2JHOV MINGW64 /c/Program Files/Docker Toolbox
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                               NAMES
1979c2734653        mysql               "docker-entrypoint.s…"   11 minutes ago      Up 11 minutes       0.0.0.0:3306->3306/tcp, 33060/tcp   mysql

anai@DESKTOP-CT2JHOV MINGW64 /c/Program Files/Docker Toolbox
```

```
anai@DESKTOP-CT2JHOV MINGW64 /c/Program Files/Docker Toolbox
$ docker stop 1979c2734653
1979c2734653

anai@DESKTOP-CT2JHOV MINGW64 /c/Program Files/Docker Toolbox
$ docker rm 1979c2734653
1979c2734653

anai@DESKTOP-CT2JHOV MINGW64 /c/Program Files/Docker Toolbox
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES

anai@DESKTOP-CT2JHOV MINGW64 /c/Program Files/Docker Toolbox
$
```

公式のmysqlイメージを使う

https://github.com/docker-library/docs/tree/master/mysql

```
docker pull mysql
docker run --name mysql -e MYSQL_ROOT_PASSWORD=mysql -d -p 3306:3306 mysql

# 接続確認 passwordはmysqlになります。
mysql -h $(boot2docker ip) -uroot -p
```

上記でいきなり使える。

EXPOSE （露出用のポート）

`run` コマンドには、コンテナのネットワークに対応する以下のオプションがあります。

```
--expose=[]: コンテナ内のポートまたはポート範囲を露出する
             これらは「EXPOSE」命令の露出ポートに追加する
-P=false   : 全ての露出ポートをホスト側インターフェースに公開する
-p=[]      : コンテナのポートまたはポート範囲をホスト側に公開する
               書式: ip:ホスト側ポート:コンテナ側ポート | ip::コンテナ側ポート | ホスト側ポート:コン
               ホスト側ポートとコンテナ側のポートは、どちらもポート範囲を指定可能です。
               両方で範囲を指定した時は、コンテナ側のポート範囲とホスト側のポート範囲が
               一致する必要があります。例：
                   -p 1234-1236:1234-1236/tcp

               ホスト側のポート範囲しか指定しない時は、コンテナ側ポートが範囲になるとは限りません。
               このような場合、コンテナ側で公開されるポートはホスト側のポート範囲のいずれかです。
                (例 -p 1234-1236:1234/tcp )

               (実際の割り当てを確認するには ``docker port`` を使う)
--link=""  : 他のコンテナに対するリンクを追加 (<名前 or id>:エイリアス or <名前 or id>)
```

イメージの開発者は、`EXPOSE` 命令以外のネットワーク機能に関する管理は行えません。`EXPOSE` 命令が定義するのは、サービスが初期化時に提供する受信用ポートです。このポートはコンテナの中のプロセスが利用可能にします。作業者は `--expose` オプションを使い、公開用ポートを追加できます。

コンテナの内部ポートを露出（expose）するために、オペレータはコンテナ実行時に `-P` か `-p` フラグを使えます。公開用のポートはホスト上でアクセス可能であり、そのポートはホストに到達可能なクライアントであれば誰でも利用できます。

`-P` オプションはホスト・インターフェース上に全てのポートを公開します。Docker は公開されたポートを、ホスト側のポートに対してランダムに拘束（bind）します。このポートの範囲をエフェメラル・ポート範囲（*ephemeral port range*） と呼び、`/proc/sys/net/ipv4/ip_local_port_range` によって定義されています。 `-p` フラグを使うと、特定のポートやポートの範囲を割り当てます。

コンテナ内のポート番号（サービスがリッスンしているポート番号）は、コンテナの外に露出するポート番号（クライアントが接続する番号）と一致させる必要がありません。例えば、コンテナ内の HTTP サービスがポート 80 をリッスンしているとします（そして、イメージ開発者は Dockerfile で `EXPOSE 80` を指定しているでしょう ）。実行する時に、ホスト側のポート 42800 以上が使われます。公開用ポートがホスト側のどのポートに割り当てられたかを確認するには、 `docker port` コマンドを使います。

デフォルトのブリッジ・ネットワークにおいて、新しいクライアント・コンテナの起動時にオペレータが `--link` を指定したら、クライアント・コンテナはプライベートなネットワーク・インターフェースを経由して公開ポートにアクセスできます。 Docker ネットワーク概要 にあるユーザ定義ネットワーク上で `--link` を指定したら、コンテナをリンクするためのエイリアス名を作成します。

ENV（環境変数）

新しいコンテナの作成時、Docker は以下の環境変数を自動的に設定します。

変数	値
`HOME`	`USER` の値を元にして指定
`HOSTNAME`	コンテナに関連づけるホスト名
`PATH`	`/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin` のよう
`TERM`	コンテナが疑似ターミナル（pseudo-TTY）を割り当てるときは `xterm`

のような一般的なディレクトリを含む

更に、オペレータはコンテナに対して **環境変数の組み合わせ** を `-e` フラグで追加できます。先ほど言及した環境変数や、開発者が Dockerfile の中で `ENV` で定義済みの環境変数を上書きできます。

```
$ docker run -e "deep=purple" --rm ubuntu /bin/bash -c export
declare -x HOME="/"
declare -x HOSTNAME="85bc26a0e200"
declare -x OLDPWD
declare -x PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
declare -x PWD="/"
declare -x SHLVL="1"
declare -x container="lxc"
declare -x deep="purple"
```

デタッチドまたはフォアグラウンド

Docker コンテナの起動時に、まず、コンテナをバックグラウンドで「デタッチド」モード（detached mode）で実行するか、デフォルトのフォアグラウンド・モード（foreground mode）で実行するかを決める必要があります。

```
-d=false: Detached mode: Run container in the background, print new container id
```

デタッチド (-d)

コンテナをデタッチド・モードで起動するには、 `-d=true` か `-d` オプションを使います。設計上、コンテナが実行するルート・プロセスが終了したら、デタッチド・モードで起動したコンテナも終了します。デタッチド・モードのコンテナは停止しても自動的に削除できません。つまり `-d` オプションでは `--rm` を指定できません。

デタッチドのコンテナでは `service x start` コマンドを受け付けません。例えば、次のコマンドは `nginx` サービスの起動を試みます。

```
$ docker run -d -p 80:80 my_image service nginx start
```

コンテナ内で `nginx` サービスの起動は成功します。しかしながら、デタッチド・コンテナの枠組み内では処理に失敗します。これはルート・プロセス（ `service nginx start` ）が終了するため、デタッチド・コンテナは停止されます。その結果、 `nginx` サービスは実行しますが、実行を継続できません。この方法を使わず `nginx` ウェブ・サーバのプロセスを実行するには、次のようにします。

```
$ docker run -d -p 80:80 my_image nginx -g 'daemon off;'
```

コンテナの入出力はネットワーク接続や共有ボリュームも扱えます。コマンドラインで `docker run` を実行し終わった後でも、必要になる場合があるでしょう。

デタッチド・コンテナに再度アタッチ（接続）するには、 `docker` attach コマンドを使います。

Docker run

`docker run`
以下3つのコマンドを順に実行するのと等しい

`docker pull`
イメージ取得

`docker create`
コンテナ作成

`docker start`
コンテナ起動

例：

```
docker run hello-world
```

※ hello-worldイメージをrunする

Docker イメージ

`docker images`
ダウンロード済みのDockerイメージ一覧表示

`docker rmi イメージ名:タグ`
イメージ削除

例：
`docker rmi hello-world:latest`

Docker コンテナ1

`docker ps -a`
停止中のものも含めてコンテナー一覧表示

`docker ps`
起動中のコンテナー一覧表示

`docker start コンテナID`
コンテナ起動

`docker stop コンテナID`
コンテナ停止

`docker rm コンテナID`
コンテナ削除

`docker attach コンテナID`
コンテナログイン（attach）

Ctrl P + Q（コンテナは停止しない）
exit　（コンテナ停止）
attachからのコンテナログアウト

`docker exec -it コンテナID /bin/sh`
コンテナログイン（exec）

Ctrl P + Q（コンテナは停止しない）
exit　（コンテナ停止しない）
execからのコンテナログアウト

コンテナとローカル環境をつなぐ

```
docker run --name ys -it -v /Users/yoshiko/Desktop/
docker:/home/docker python:3.6 /bin/bash
```

Python3.6のイメージでコンテナを立て、ローカル環境とつなぐ

オプション

--rm

コンテナ終了時に自動的に削除

-v

バインドマウント

-it

i:アタッチされていない状態でも入力を保持できる

t: 擬似ターミナルを割り当てる

DockerHub へイメージを pushする

```
docker login
```

```
Authenticating with existing credentials...
Login Succeeded
```

Docker Hubへログイン

```
docker commit コンテナID 任意の名前:任意のtag
```

停止したコンテナからイメージを作る

```
docker tag イメージID dockerUser名/リポジトリ名
```

イメージにタグ付け

```
docker push dockerUser名/リポジトリ名
```

Docker Hubにpushする

```
docker pull dockerUser名/リポジトリ名
```

Docker Hubからイメージをpullする