練り物でもわかる
コンテナ技術

～ Dockerでどっかーん（笑）～

メタリックはんぺん

今日のながれ

- コンテナってなんですか
  - ↳ コンテナの用途、仮想マシンとの違い
- Dockerをつかってみる
  - ↳ docker pull, docker run, docker commit
- Dockerをべんりにつかう
  - ↳ docker build

自己紹介

- ぷよぐらま？

```
println!("making neighbor table...");
let mut result = Vec::new();
let h = data.len()    as i32;
let w = data[0].len() as i32;
let size = 3;
let mut exptable : Vec<f64> = Vec::new();
for i in 0..(size*size*2)+1 {
    exptable.push(std::f64::consts::E.powf(-i as f64));
}
let mut c =0;
for _y in 0..h {
    print!("{}/{}\r",_y.to_string(),h.to_string());
    let mut r = Vec::new();
    for _x in 0..w {
        let mut sum_weight = 0.0;
        let mut sumx = 0.0;
        let mut sumy = 0.0;
        for y in i32::max(_y-size,0)..i32::min(_y+size,h-1)+1 {
            for x in i32::max(_x-size,0)..i32::min(_x+size,w-1)+1 {
                let weight = exptable[((y-_y)*(y-_y) + (x-_x)*(x-_x)) as
                sum_weight += weight;
                if y != _y {
                    sumy += (data[y as usize][x as usize]-data[ y as usize
```

自己紹介

- ぷよぐらま？
- じてんしゃがすき

自己紹介

- ぷよぐらま？
- じてんしゃがすき
- とうげはこえるもの

自己紹介

- ぷよぐらま？
- じてんしゃがすき
- とうげはこえるもの
- ぼくはなんのひと…

おねがい

- ぼく「Docker完全に理解した」
- あんまり真に受けないで
- ツイッターでマサカリをなげてください
- マジで頼むぞお前ら

モチベーション

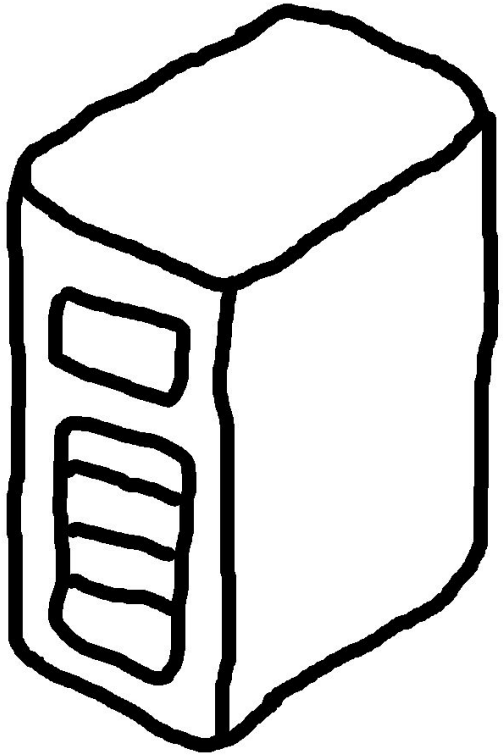
- ひと「このソフト（例：LaTeX）、すっごく便利なんだ」
- ぼく「へー、つかってみよ」
- ひと「でもこれインストールもすっごく面倒だよ」

モチベーション

- ひと「このソフト（例：LaTeX）、すっごく便利なんだ」
- ぼく「へー、つかってみよ」
- ひと「でもこれインストールもすっごく面倒だよ」
- ぼく「じゃあそのソフトが動いてるお前のPCをよこせ」

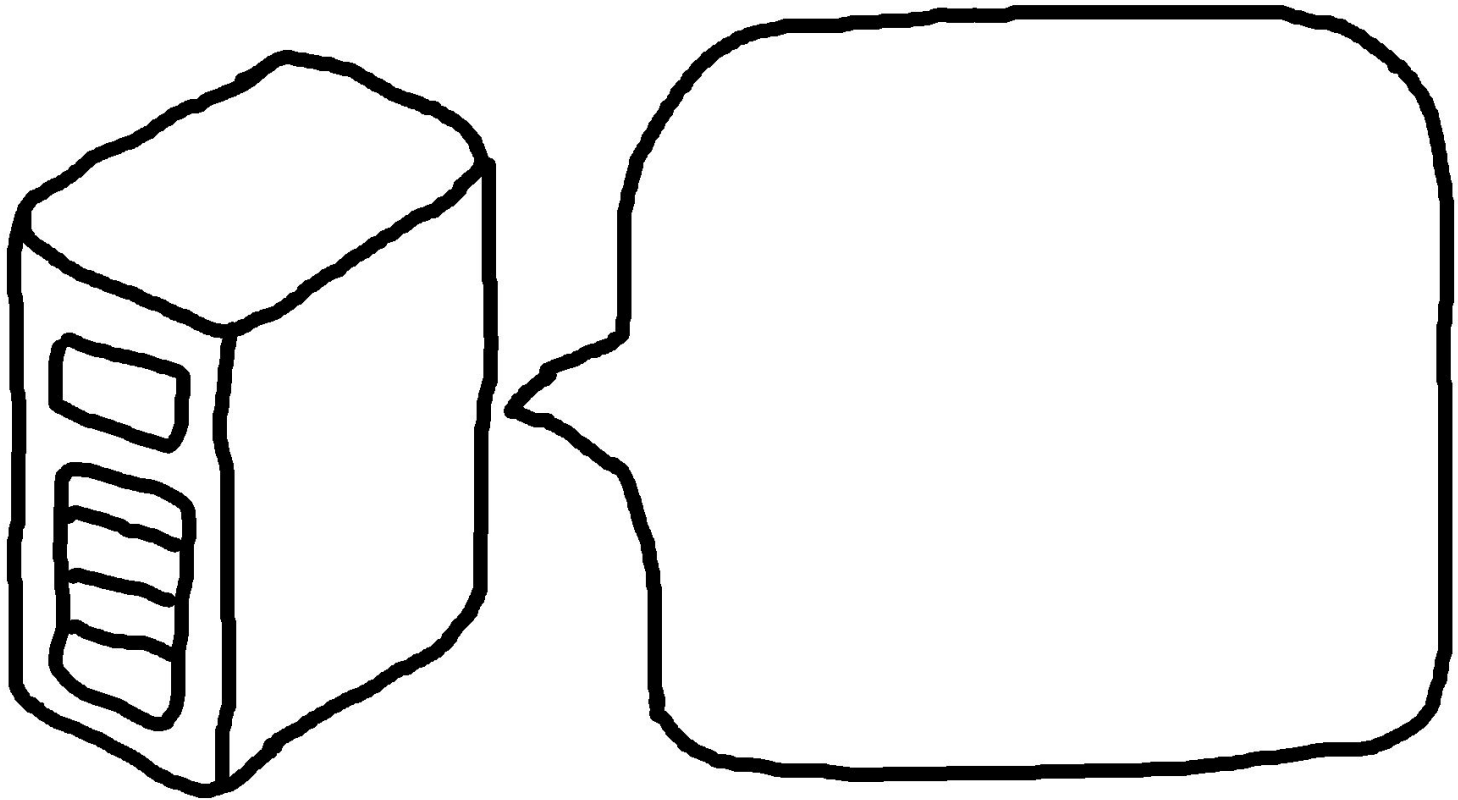
↑これができる

モチベーション

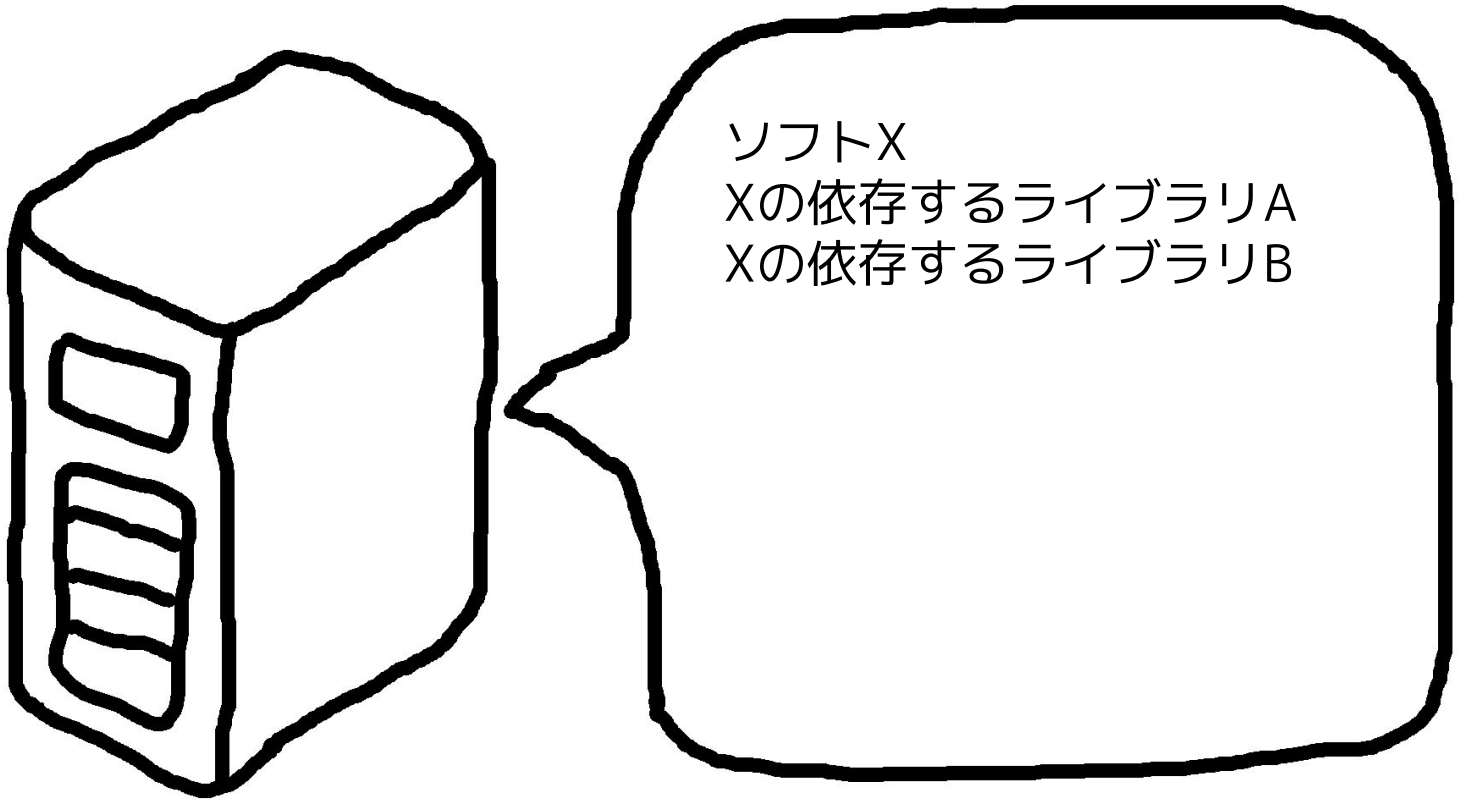


みなさんの、
パソコン

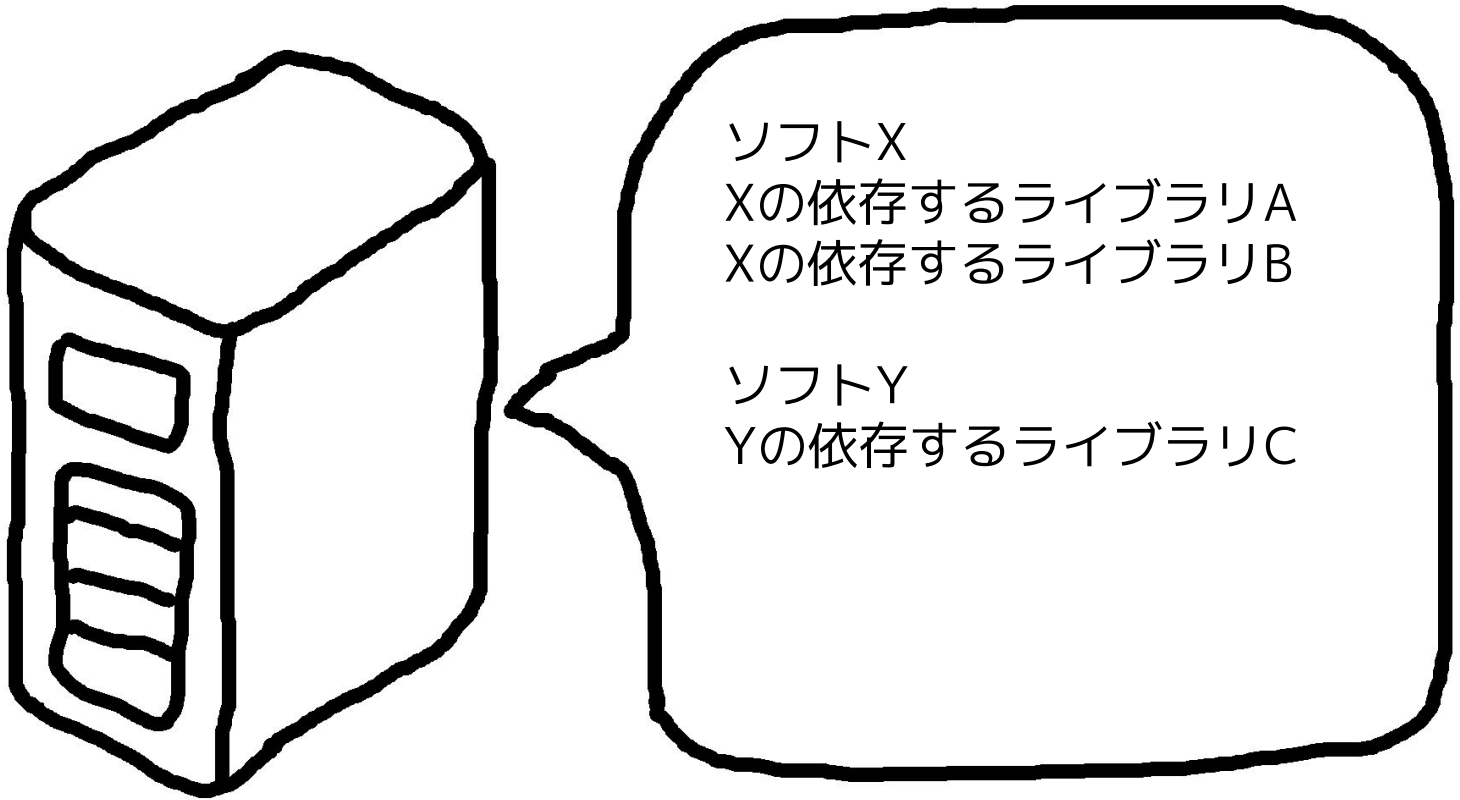
ふつうのインストール



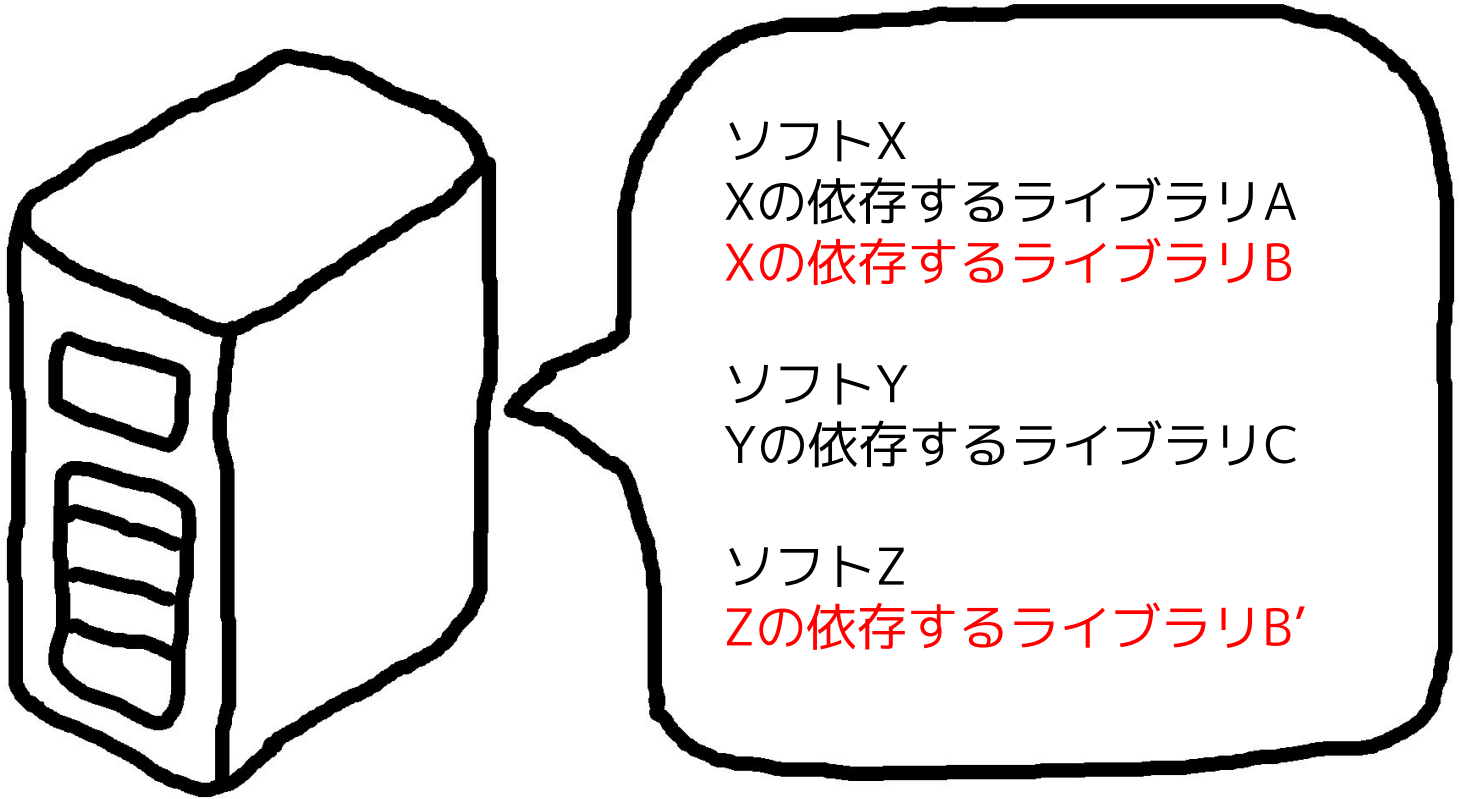
ふつうのインストール



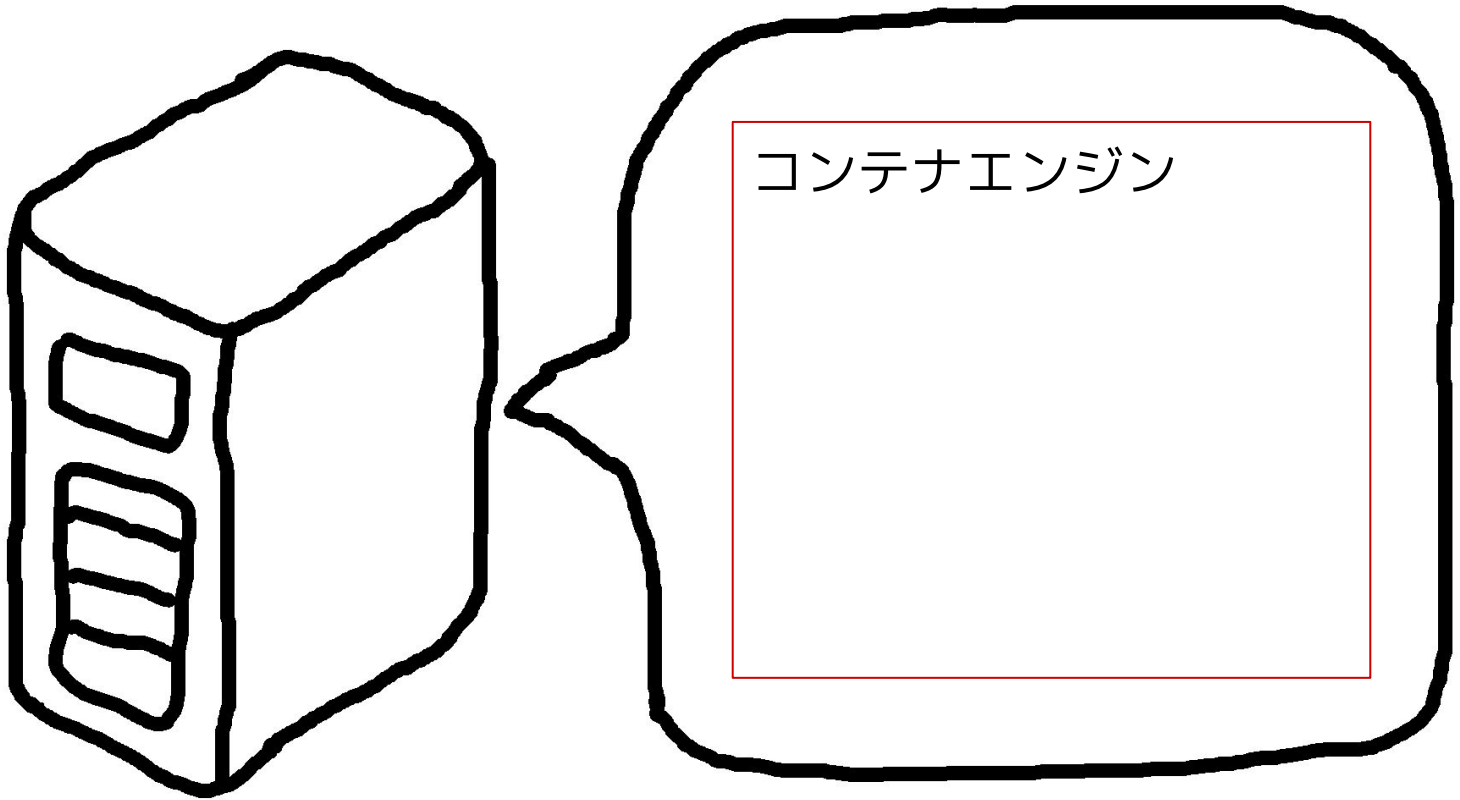
ふつうのインストール



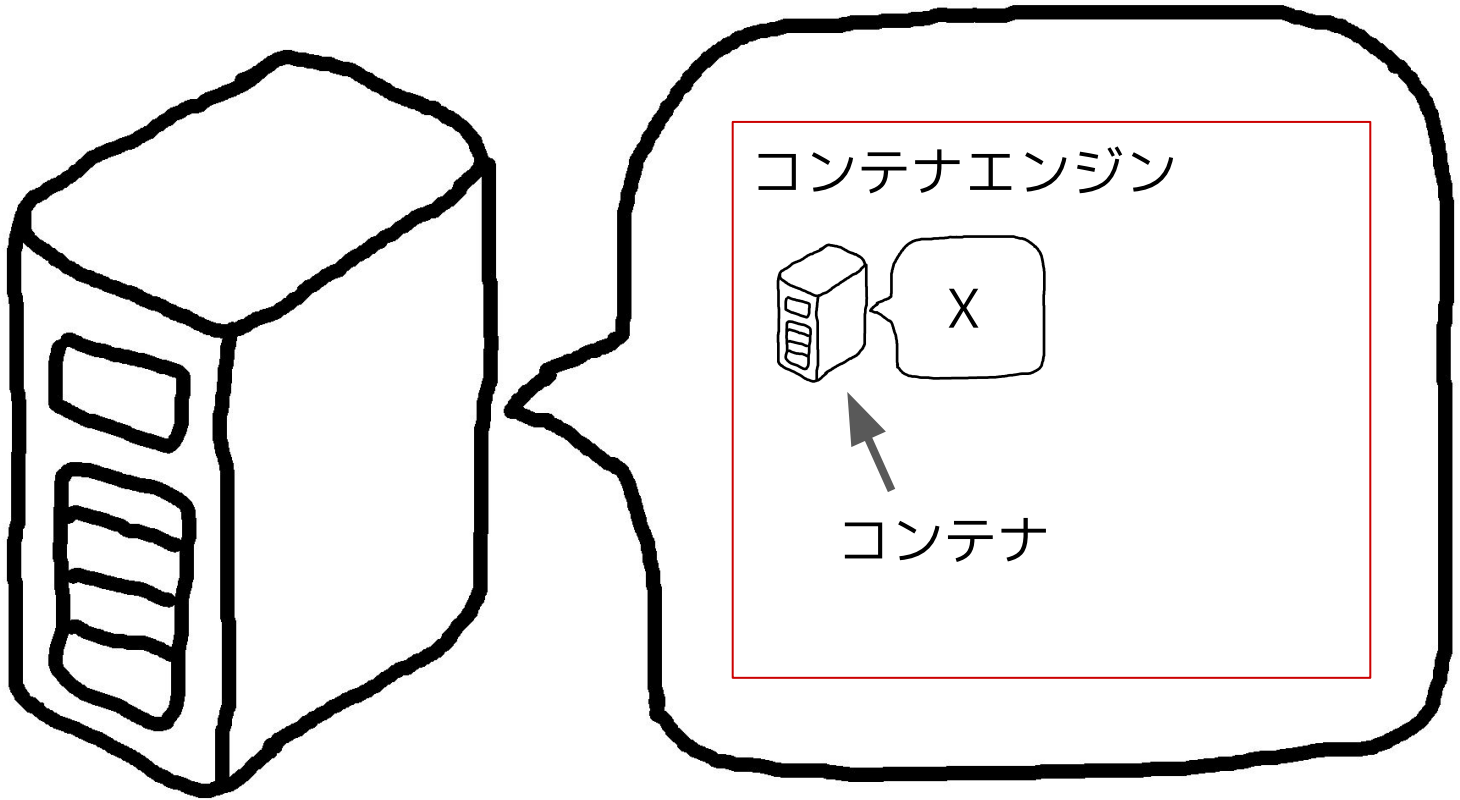
ふつうのインストール


ソフトX
Xの依存するライブラリA
Xの依存するライブラリB
ソフトY
Yの依存するライブラリC
ソフトZ
Zの依存するライブラリB’

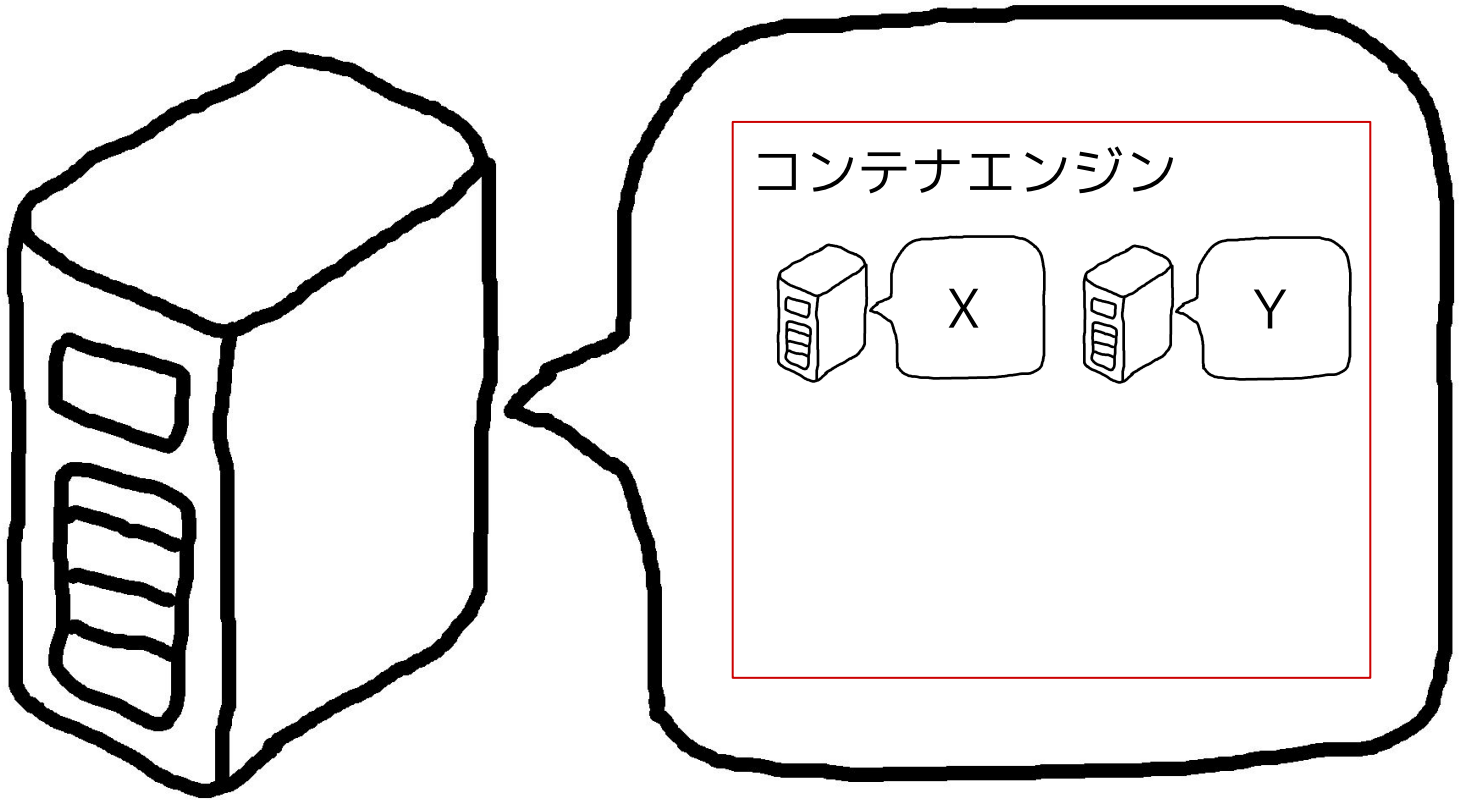
かわりにコンテナをつかう



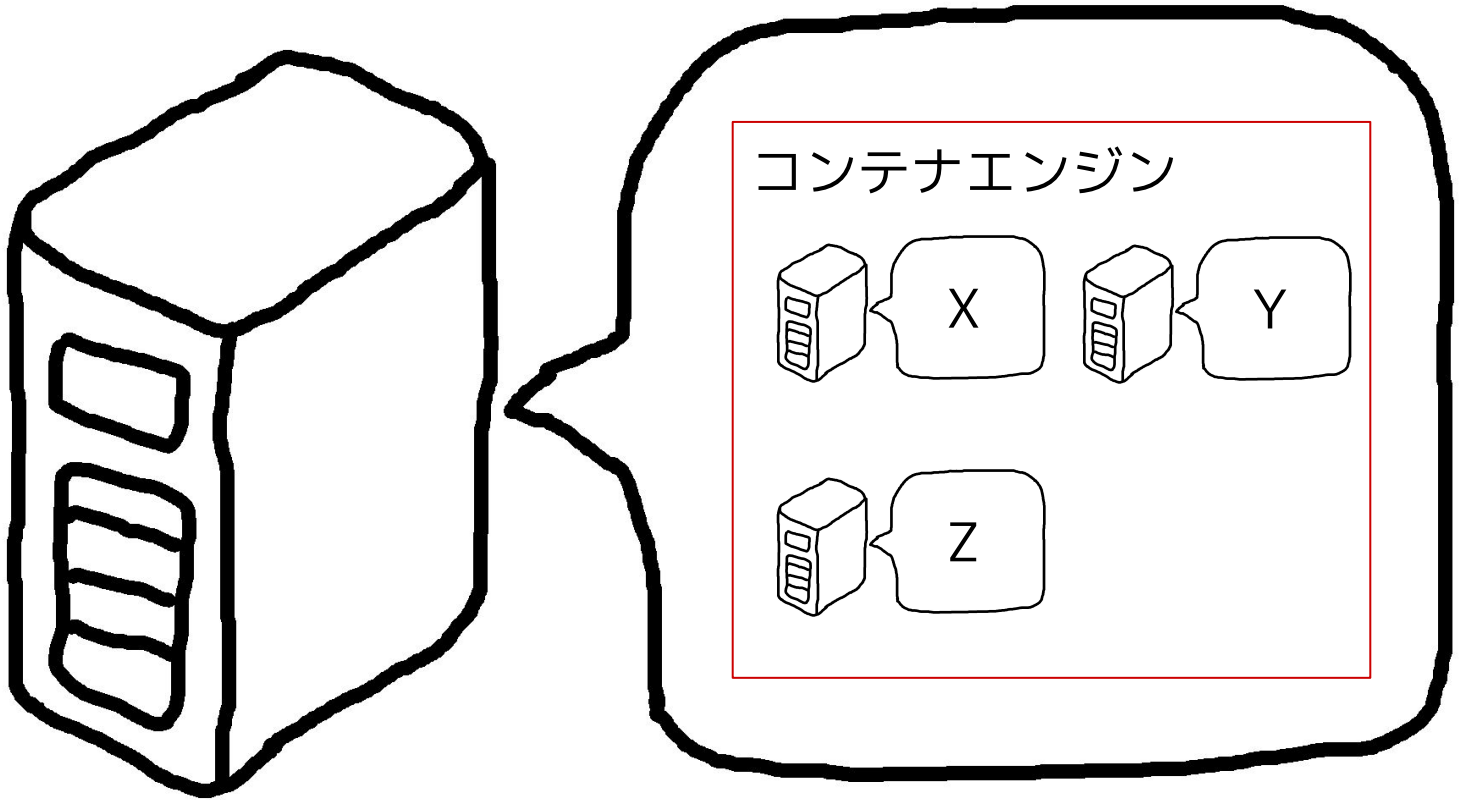
かわりにコンテナをつかう



かわりにコンテナをつかう



かわりにコンテナをつかう

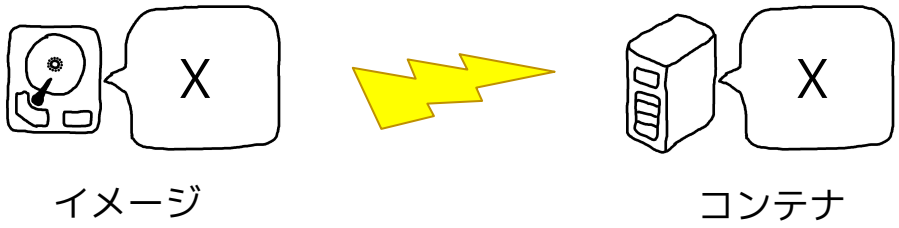


仮想マシンとの違い

- 仮想マシン：
  - OSの心臓部(カーネル)からシミュレーション
  - 重い
  - ただし、Linux上でWindows動かしたりできる
- コンテナ：
  - OSの心臓部を共有
  - 軽い
  - ホストOSと同種の環境で動くものしか使えない

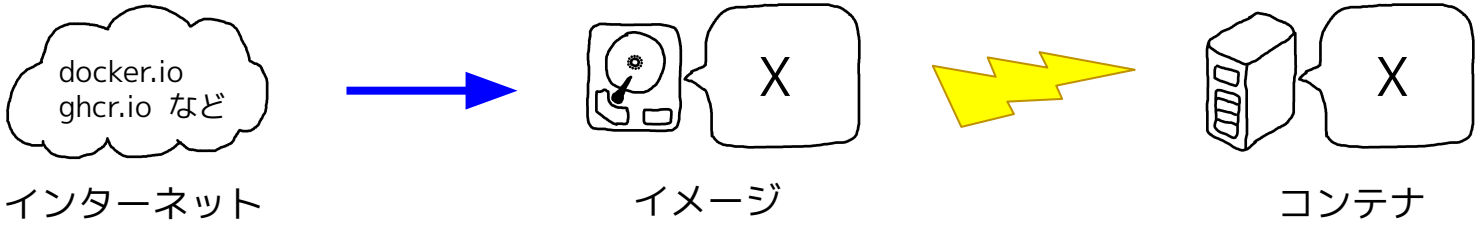
Dockerで、どっかーん(笑)

- Docker：人気のコンテナエンジン
- コンテナの雛形である「イメージ」を使う



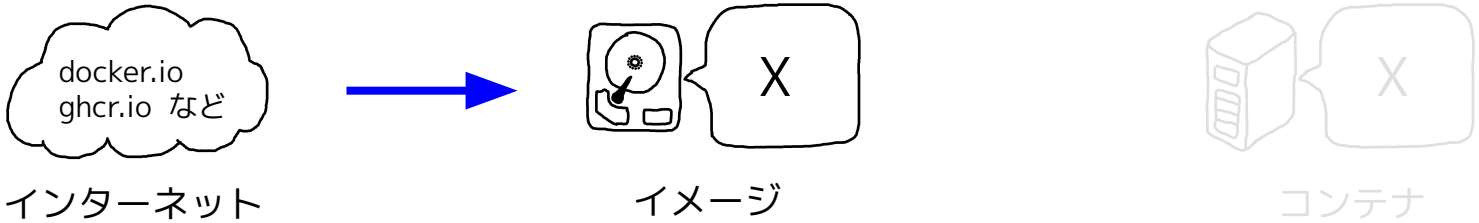
Dockerで、どっかーん(笑)

- Docker：人気のコンテナエンジン
- コンテナの雛形である「イメージ」を使う
- イメージの公開・受け渡しが容易



つかってみる

- docker pull [イメージ名]
  - Docker Hubで公開されているイメージを持ってくる

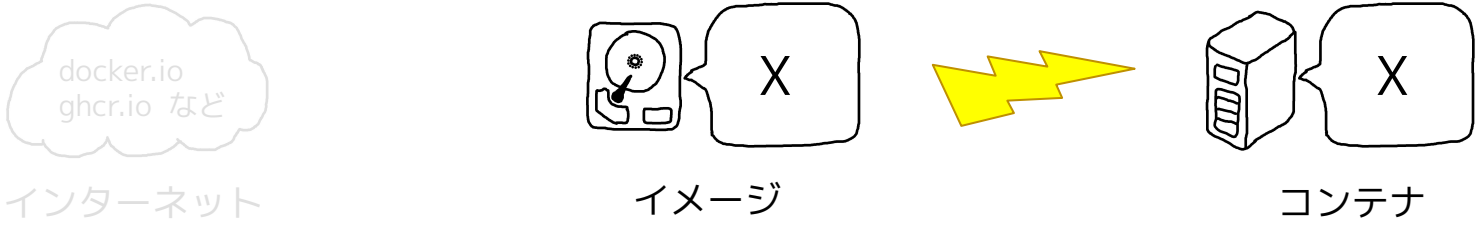


つかってみる

- docker pull [イメージ名]
  - Docker Hubで公開されているイメージを持ってくる
- イメージの例
  - debian:stable … debianもどき
  - debian:stable-slim … ↑の軽量版
  - node:slim … nodeが入ってる軽量イメージ

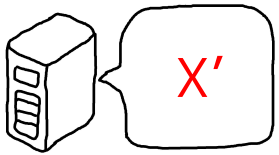
つかってみる

- docker run [イメージ名/イメージid]
  - 手元にあるイメージをコンテナにする
  - イメージ作成時に設定したコマンドが起動する



つかってみる

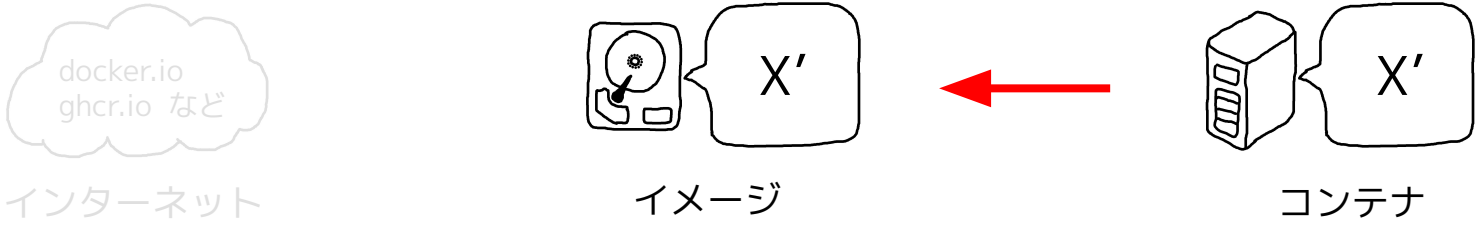
- docker run [イメージ名/イメージid]
  - 手元にあるイメージをコンテナにする
  - イメージ作成時に設定したコマンドが起動する
- docker run -it [イメージ名/イメージid] /bin/sh
  - コンテナ内のshを開く



コンテナ

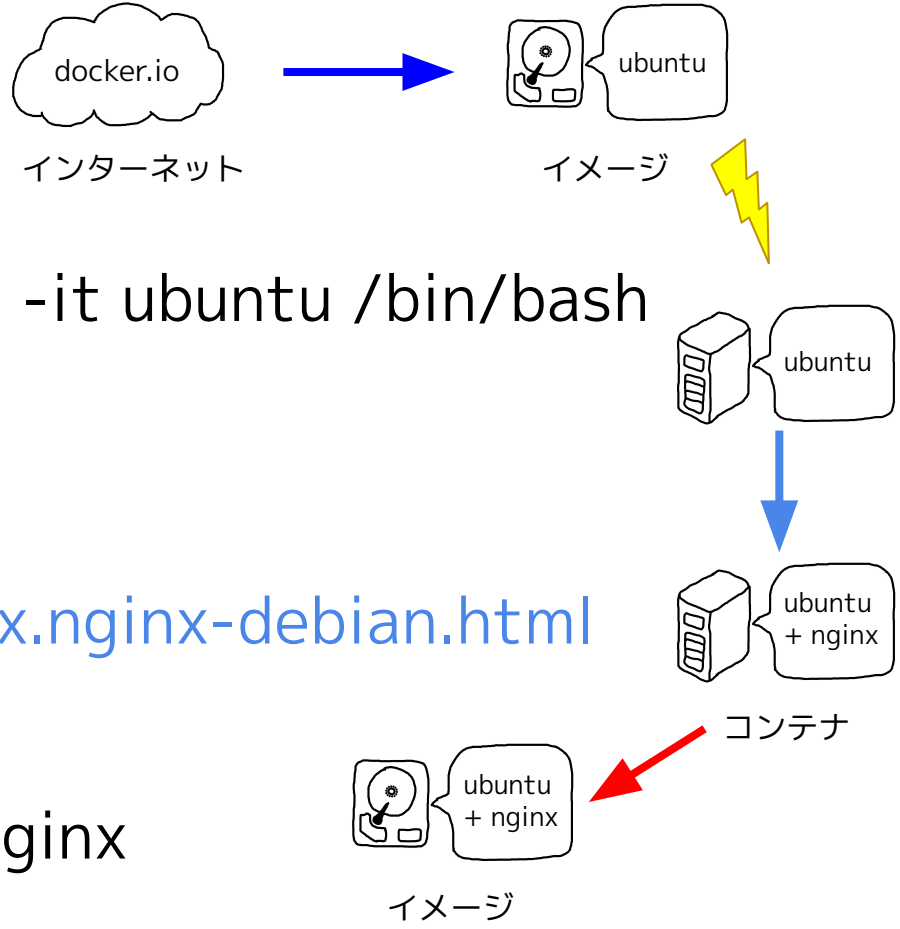
つかってみる

- docker commit [コンテナ名/コンテナid]
  - 指定したコンテナをイメージにする
  - pull → 中に入って作業 → commit で
    作業状態の保存ができる……！？

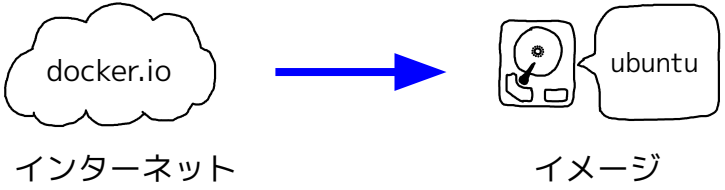


つかってみる

- $ docker pull ubuntu
- $ docker run —name c_test -it ubuntu /bin/bash
- # apt update
- # apt install nginx vim
- # vim /var/www/html/index.nginx-debian.html
- # exit
- $ docker commit c_test i_nginx

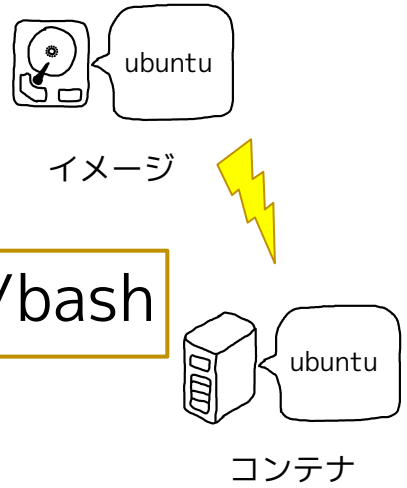


つかってみる



- $ docker pull ubuntu
- $ docker run —name c_test -it ubuntu /bin/bash
- # apt update
- # apt install nginx vim
- # vim /var/www/html/index.nginx-debian.html
- # exit
- $ docker commit c_test i_nginx

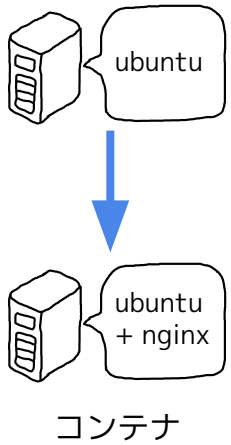
つかってみる



- $ docker pull ubuntu
- $ docker run —name c_test -it ubuntu /bin/bash
- # apt update
- # apt install nginx vim
- # vim /var/www/html/index.nginx-debian.html
- # exit
- $ docker commit c_test i_nginx

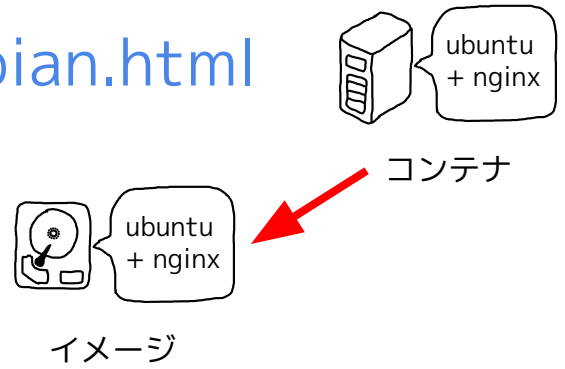
つかってみる

- $ docker pull ubuntu
- $ docker run —name c_test -it ubuntu /bin/bash
- # apt update
- # apt install nginx vim
- # vim /var/www/html/index.nginx-debian.html
- # exit
- $ docker commit c_test i_nginx



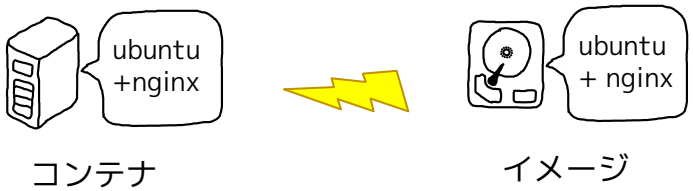
つかってみる

- $ docker pull ubuntu
- $ docker run —name c_test -it ubuntu /bin/bash
- # apt update
- # apt install nginx vim
- # vim /var/www/html/index.nginx-debian.html
- # exit
- $ docker commit c_test i_nginx



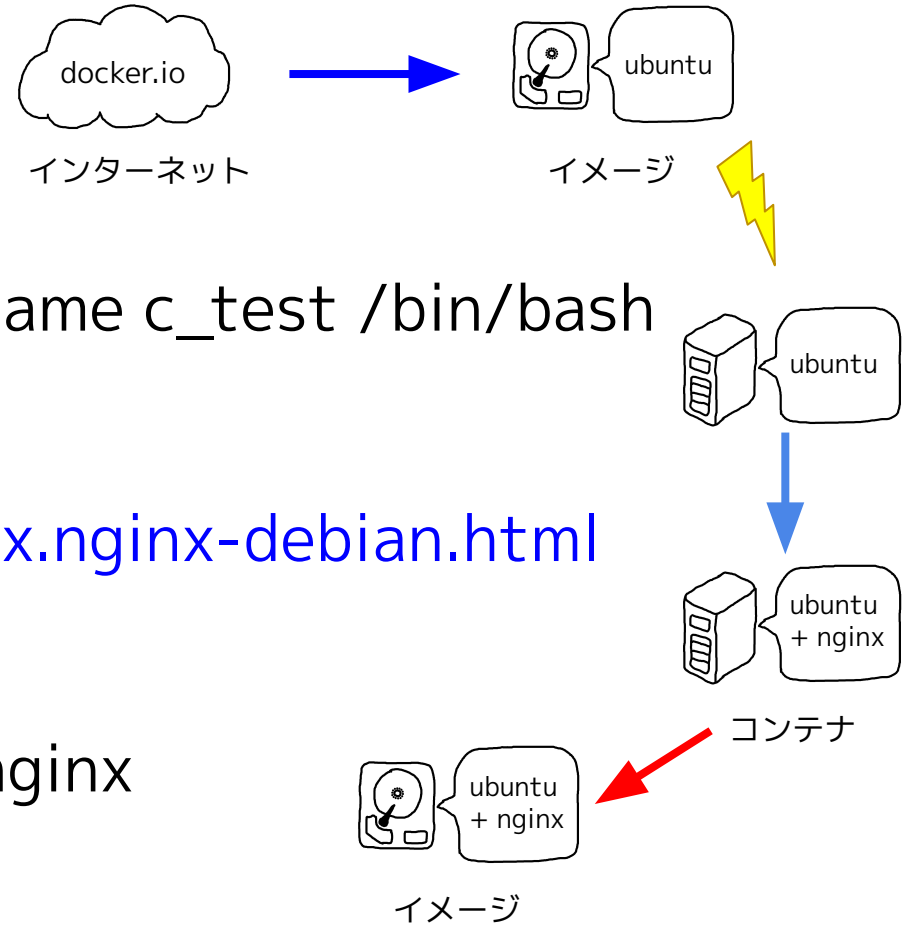
つかってみる

- $ docker run —name c_web -p 8081:80 -itd i_web bash -c “nginx && /bin/bash”
- $ curl http://127.0.0.1:8081
- $ docker stop c_web



さっきの

- $ docker pull ubuntu
- $ docker run -it ubuntu —name c_test /bin/bash
- # apt install nginx vim
- # vim /var/www/html/index.nginx-debian.html
- # exit
- $ docker commit c_test i_nginx



自動化する

```
FROM ubuntu
RUN apt update && apt install -y nginx
COPY html /var/www/html
CMD ["/usr/sbin/nginx", "-g", "daemon off;"]
```

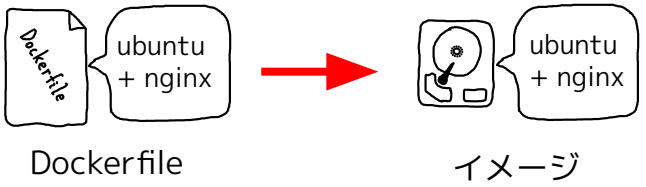
imageの生成の自動化

- Dockerfile を書いて、docker build . する
- 主な命令
  - FROM [イメージ名]
    - 公開されている/手元にあるイメージを元にする
  - RUN [コマンド]
    - コンテナでコマンドを実行する

imageの生成の自動化

- 主な命令（続）
  - COPY [ホスト側のパス] [コンテナ側のパス]
    - ホストのファイルをコンテナにコピーする
  - WORKDIR [コンテナ側のパス]
    - cd（コマンド）みたいなやつ
  - CMD [コマンド･引数(JSON記法)]
    - docker runしたときデフォルトで実行される

イメージをビルドする



- $ docker build -t i_web2 .
- $ docker run —name c_web2 -p 8081:80 -d i_web2
- $ curl http://127.0.0.1:8081
- $ docker stop c_web2

便利なイメージを使う

```
FROM nginx
COPY html /var/www/html
CMD ["/usr/sbin/nginx", "-g", "daemon off;"]
```

実話

- 某氏「Discord用のbotを作ったので置いてください」
- ぼく「はーい♪　……どうするんだっけ、ぼくtsのビルドのしかた知らないし、npm buildでいいのか？？？　プロセスの管理は？？？？？」

実話

- 某氏「Discord用のbotを作ったので置いてください」
- ぼく「はーい♪　……どうするんだっけ、ぼくtsのビルドのしかた知らないし、npm buildでいいのか？？？　プロセスの管理は？？？？？」
- 某氏「Dockerfile作ってあるのでそれ使ってください」

結論：

結論：

コンテナは、いい文明。

実用事例

- Discordのbot
  - ぼく「環境構築も起動の仕方もわからん！」
  - 某氏「Dockerfileがあるので、使ってください」
- VSCodeのリモートコンテナ
  - Dockerfileで開発環境を構築できる

続・よくある悲劇

- ぼく「……入れてきたぞ、全部」
- ソフト「それは別のソフトが使ってる*[ライブラリB']* だ、私が要求している*[ライブラリB]* とはバージョンが違うし互いに競合する」
- 死「ぼく」

どうすればいいのか

- 必要なファイルだけ全部詰めたらたぶんうごく：

```
├ /usr/lib
│     ├ [ライブラリA]
│     ├ [ライブラリB]
│     └ ……
├ /usr/bin
│     └[なんかのソフト]
└……
```

どうすればいいのか

- 必要なファイルだけ全部詰めたらたぶんうごく：

```
├ /usr/lib
│      ├ [ライブラリA]
│      ├ [ライブラリB]
│      └ ……
├ /usr/bin
│      └[なんかのソフト]
└……
```

- ソフト1個動かすごとにPC1個用意しろってこと……？

どうすればいいのか

- 必要なファイルだけ全部詰めたらたぶんうごく：

```
├ /usr/lib
│     ├ [ライブラリA]
│     ├ [ライブラリB]
│     └ ……
├ /usr/bin
│     └[なんかのソフト]
└……
```

↓できるわけねえじゃん！！！

- ソフト1個動かすごとにPC1個用意しろってこと……？

コンテナとはなにか

- ソフトを動かすためのPCのはりぼて
- コンテナの内側からは独立したPCに見える
- 独立したファイルシステムを持つ