Dockerについて

自己紹介

- 名前：苗村 智行（なむら ともゆき）
- ハンドルネーム：shika
- 所属：NTTアドバンステクノロジ株式会社
  ビジネスインテリジェンスAIセンタ
- 入社３年目
- 主な業務内容：機械学習モデル構築案件
- よく使うツール：Python, R言語, Docker

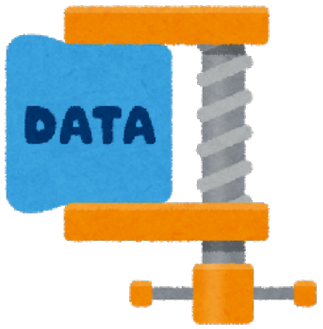
機械学習モデル構築案件

- データベース等にある生データを加工し、機械学習アルゴリズムに学習させることでモデルを作成

入力

機械学習モデル構築

データ加工



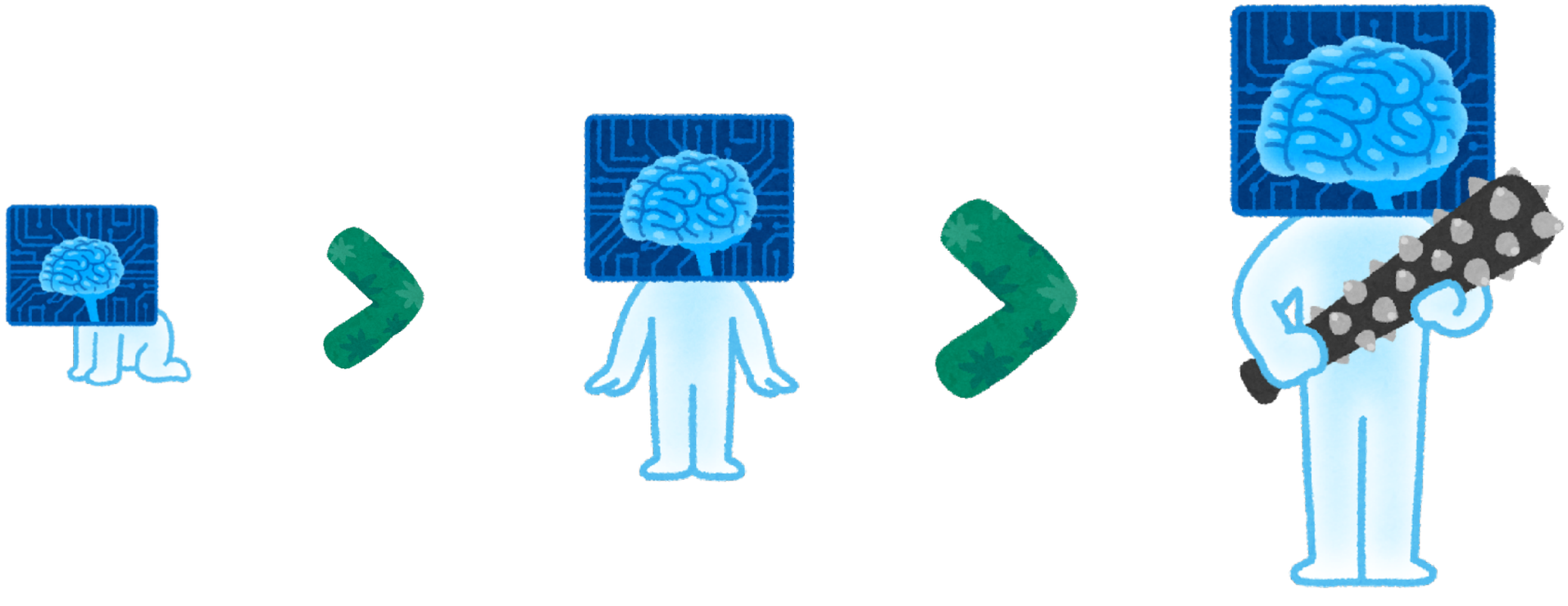
学習

精度評価

出力

機械学習モデル構築案件

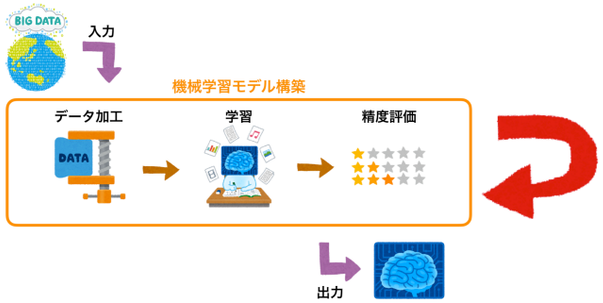
- データベース等にある生データを加工し、機械学習アルゴリズムに学習させることでモデルを作成
- **try&errorを繰り返すことでより強いAIを作ることを目指す**



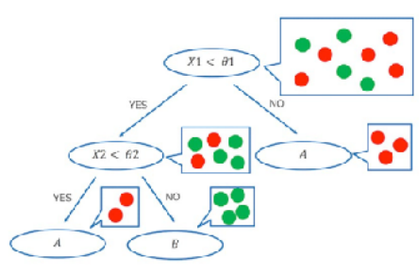
よく困る点

- **案件毎にデータの形式や適した機械学習アルゴリズムが違う**
  - データ形式：表形式、json、画像、自然言語、etc
  - 機械学習アルゴリズム：線形回帰、決定木系、NN、etc

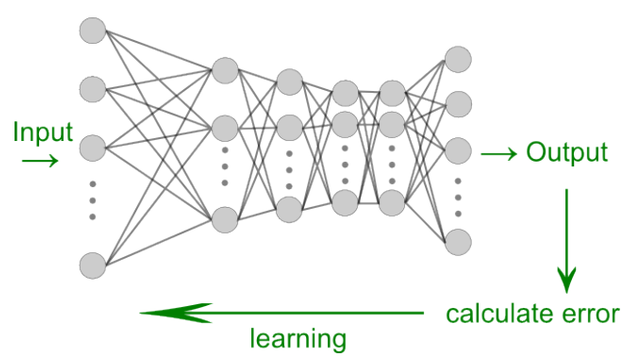
	データ形式	**機械学習アルゴリズム**
案件A	表形式	決定木系
案件B	自然言語	NN
・ ・ ・		



id	x1	x2	x3
1	12	11	6
2	22	65	5
3	3	55	8
4	56	6	1



多層ニューラルネットの考え方は、
ニューラルネットの元始となるパーセプトロンが
出て来た当初から存在していた。
多層ニューラルネットは理論上どんな関数でも
表現でき、今回のテーマのCNNも、
日本人研究者である福島邦彦先生が1982年に
ネオコグニトロン*1として発表されたものが元に
なっている。



よく困る点

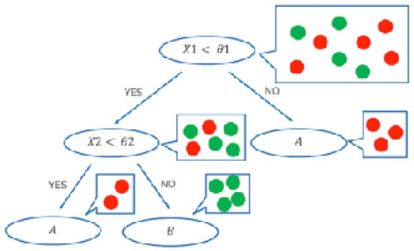
- 案件毎にデータの形式や適した機械学習アルゴリズムが違う
  → **これらに応じたDBMSやライブラリを使う必要がある**

表形式

id	x1	x2	x3
1	12	11	6
2	22	65	5
3	3	55	8
4	56	6	1

PostgreSQL　MySQL　cassandra

決定木系



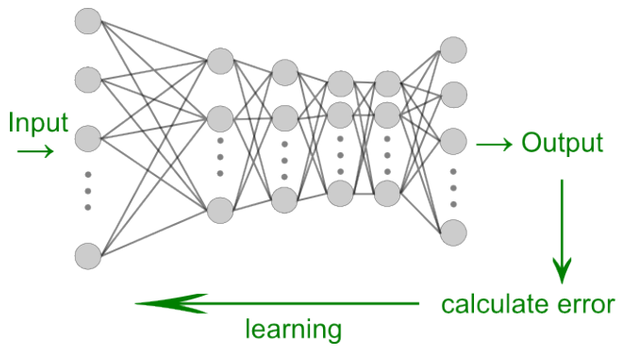
python　scikit learn　R

自然言語

多層ニューラルネットの考え方は、
ニューラルネットの元始となるパーセプトロンが
出て来た当初から存在していた。
多層ニューラルネットは理論上どんな関数でも
表現でき、今回のテーマのCNNも、
日本人研究者である福島邦彦先生が1982年に
ネオコグニトロン*1として発表されたものが元に
なっている。

Amazon DynamoDB　redis

NN



python　PyTorch　Chainer　TensorFlow

よく困る点

- 案件毎にデータの形式や適した機械学習アルゴリズムが違う
  → これらに応じたDBMSやライブラリを使う必要がある
    → **全部を一つのホストマシン上に環境構築したくない！**

よく困る点

- 案件毎にデータの形式や適した機械学習アルゴリズムが違う
  → これらに応じたDBMSやライブラリを使う必要がある
    → **全部を一つのホストマシン上に環境構築したくない！**

一つ一つの環境構築が面倒で時間かかる…

依存ライブラリの違いで動作しないかも…

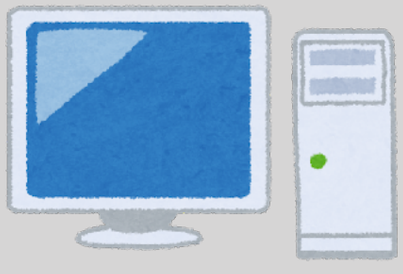
案件毎にサラピンの環境が使いたいけど全部消して入れ直すの大変…

バージョン管理が面倒…

便利そうなツールがあっても お試しで使いにくい…

チームメンバー間で環境を合わせるの大変…

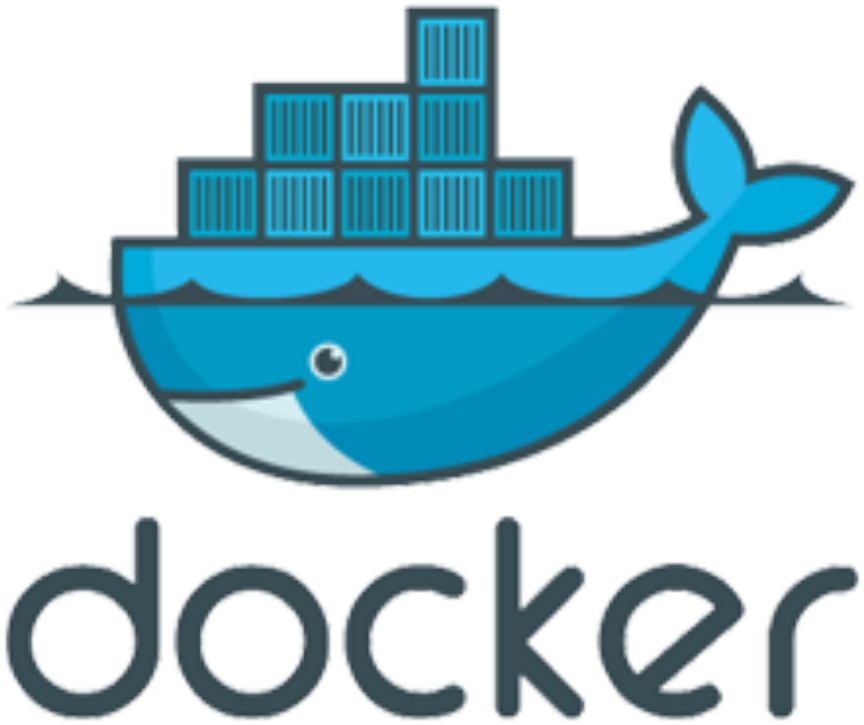
そもそもローカルで使えないやろ…



ホストマシン

Dockerを用いれば…

- 各ツールを別々の仮想環境に建てホストからアクセスして利用可！
- 仮想環境の構築・削除が恐ろしく簡単かつ素早く可能！
- 自分の仮想環境を他者へ簡単に共有可能！

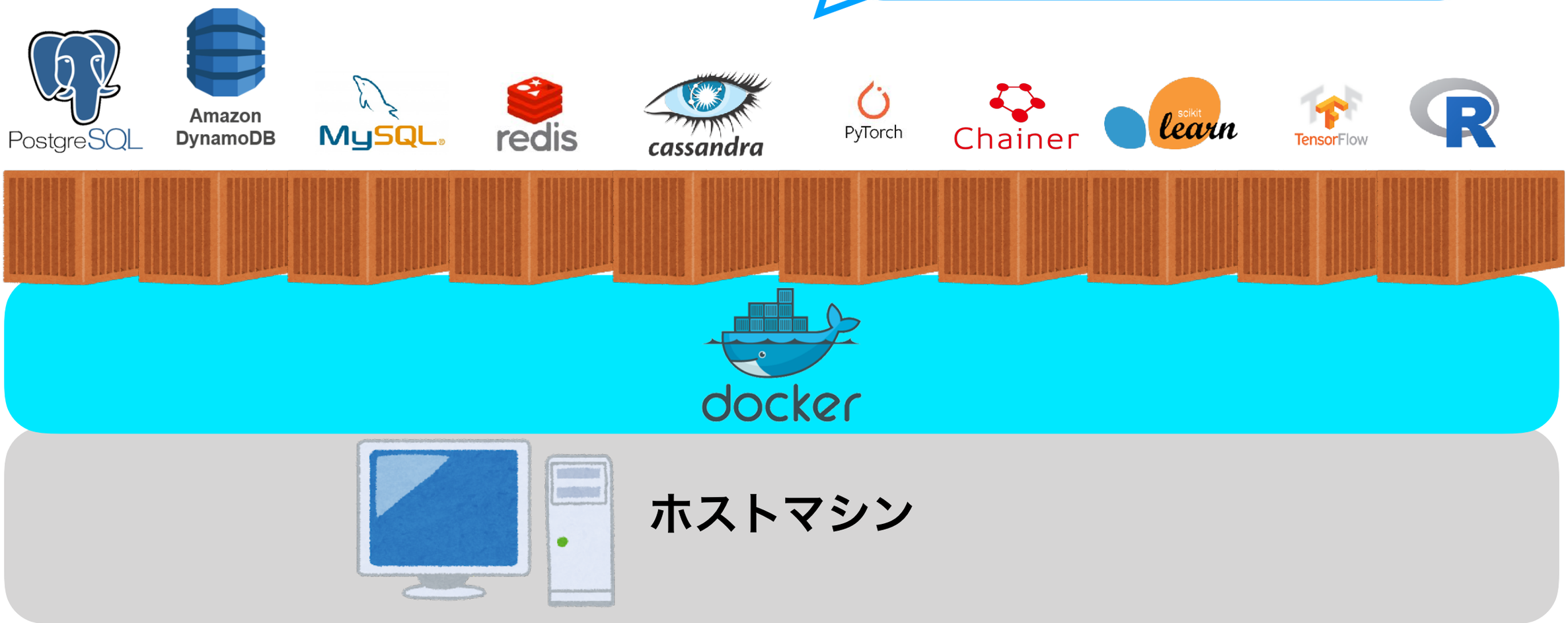


コンテナ型の仮想環境を
作成、配布、実行
するためのプラットフォーム

Dockerを用いれば…

- 各ツールを別々の仮想環境に建てホストからアクセスして利用可！
- 仮想環境の構築・削除が恐ろしく簡単かつ素早く可能！
- 自分の仮想環境を他者へ簡単に共有可能！

ツール間の依存ライブラリ
のバージョンとか気にする必要なし



Dockerを用いれば…

- 各ツールを別々の仮想環境に建てホストからアクセスして利用可！
- 仮想環境の構築・削除が恐ろしく簡単かつ素早く可能！
- 自分の仮想環境を他者へ簡単に共有可能！

仮想環境の新規構築方法は…

```
$ docker run ~
```

仮想環境の停止・削除方法は…

```
$ docker rm ~
```

Dockerを用いれば…

- 各ツールを別々の仮想環境に建てホストからアクセスして利用可！
- 仮想環境の構築・削除が恐ろしく簡単かつ素早く可能！
- 自分の仮想環境を他者へ簡単に共有可能！

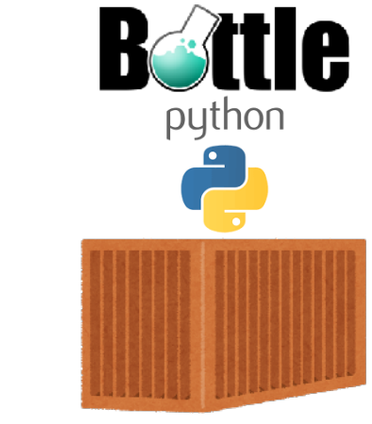
例：PostgreSQL9.5の環境を作成

```
仮想環境（コンテナ名：postgres95）作成
$ docker run -d --name postgres95 -e POSTGRES_PASSWORD=test -p
5433:5432 postgres:9.5

仮想環境へアクセス
$ docker exec -it postgres95 /bin/bash

PostgreSQLにログイン & ログアウト & 仮想環境から出る
# psql -U postgres
postgres=# \q
# exit

仮想環境を停止&削除
$ docker stop postgres95
$ docker rm postgres95
```

Dockerを用いれば…

- 各ツールを別々の仮想環境に建てホストからアクセスして利用可！
- 仮想環境の構築・削除が恐ろしく簡単かつ素早く可能！
- 自分の仮想環境を他者へ簡単に共有可能！

既存の仮想環境をイメージ化して
共有するだけ！



仮想環境(コンテナ)

イメージ化

Docker
イメージ

共有

Dockerを用いれば…

- 各ツールを別々の仮想環境に建てホストからアクセスして利用可！
- 仮想環境の構築・削除が恐ろしく簡単かつ素早く可能！
- 自分の仮想環境を他者へ簡単に共有可能！

例：pythonのbottle環境をイメージ化

```
仮想環境（コンテナ名：python37）作成 & bottle0.12.7インストール
$ docker run -it -d --name python37 python:3.7.5
$ docker exec -it python37 /bin/bash
# pip install bottle==0.12.7
# pip list

仮想環境を停止・イメージ化
$ docker stop python37
$ docker commit python37 bottle_image

イメージをtar.gz形式で保存
$ docker save bottle_image > bottle_image.tar.gz
```

Dockerを用いれば…

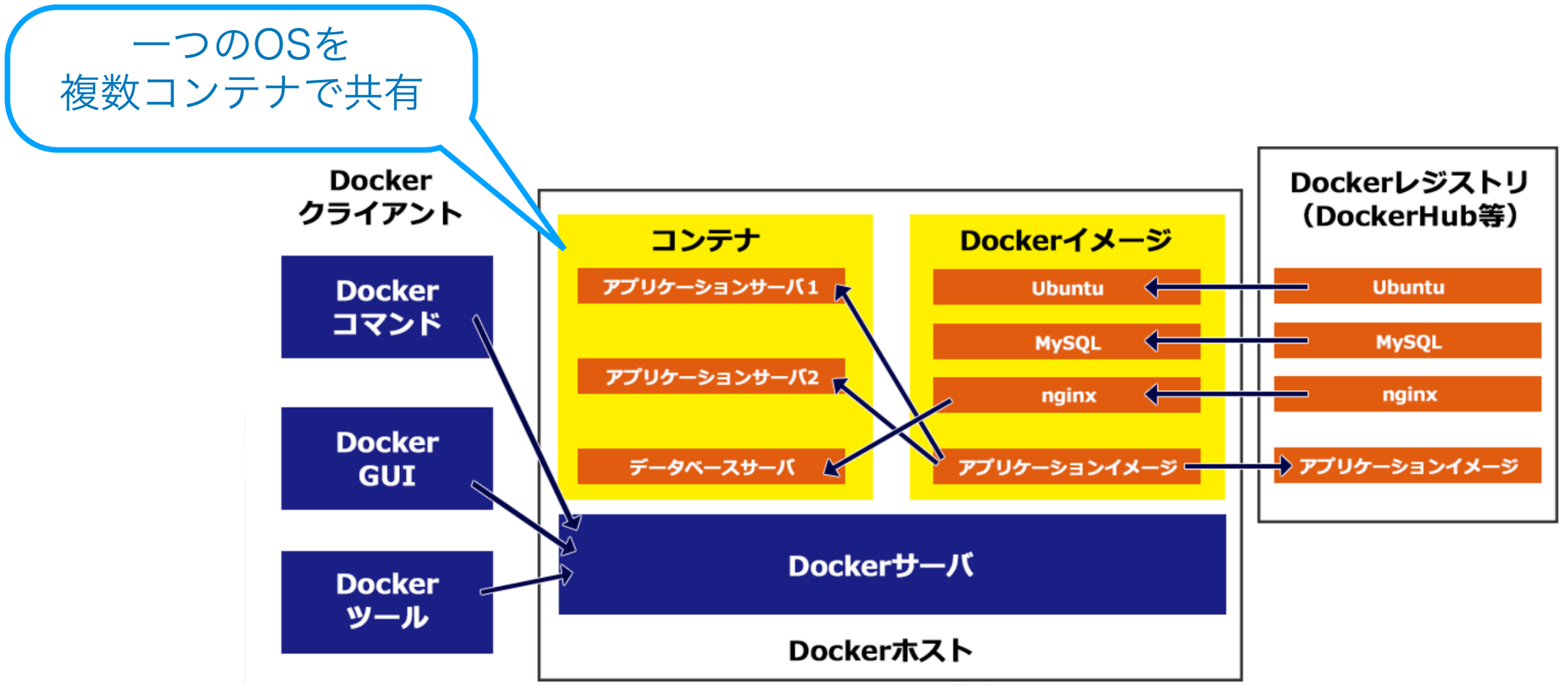
- 各ツールを別々の仮想環境に建てホストからアクセスして利用可！
- 仮想環境の構築・削除が恐ろしく簡単かつ素早く可能！
- 自分の仮想環境を他者へ簡単に共有可能！

例：pythonのbottle環境のイメージを読み込み

```
tar.gz形式のイメージを読み込み
$ docker load < bottle_image.tar.gz

読み込んだイメージから仮想環境（コンテナ名：bottle12）作成
$ docker run -it -d --name bottle12 bottle_image

イメージ化前と同じ環境であることを確認
$ docker exec -it bottle12 /bin/bash
# pip list

コンテナ停止 & 削除 & イメージを削除
$ docker stop bottle12
$ docker rm bottle12
$ docker rmi bottle_image
```

Dockerについて

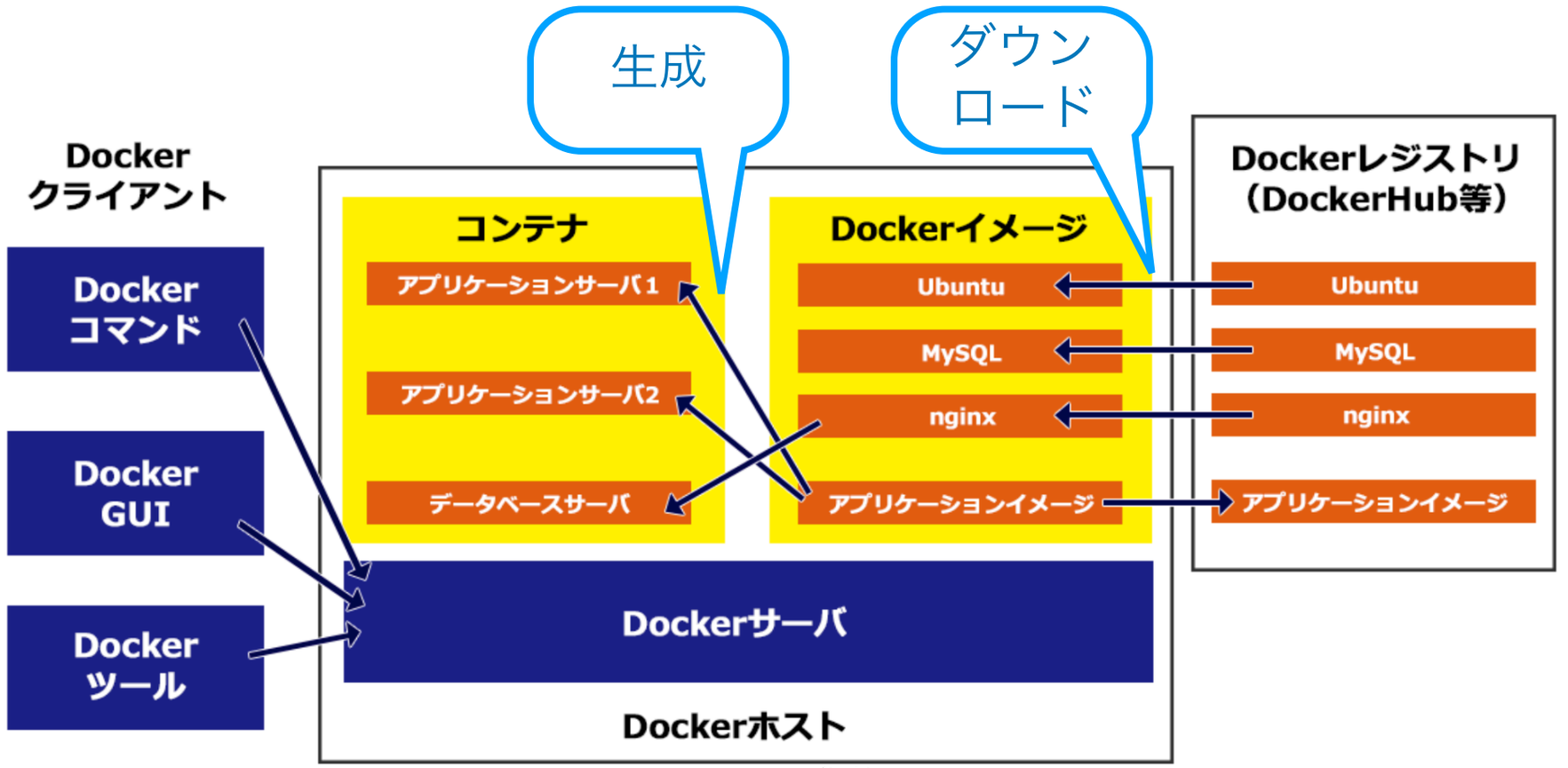
- コンテナ型仮想環境を作成、配布、実行するためのプラットフォーム
  一つのOSを複数コンテナで共有する仕組みで軽量・高速化（詳細は知りません…）
- Dockerイメージをダウンロードしてコンテナを生成
- DockerHubに公式イメージが多数公開されており、環境構築済みのイメージをすぐに利用可能



参考：https://udemy.benesse.co.jp/development/web/docker.html

Dockerについて

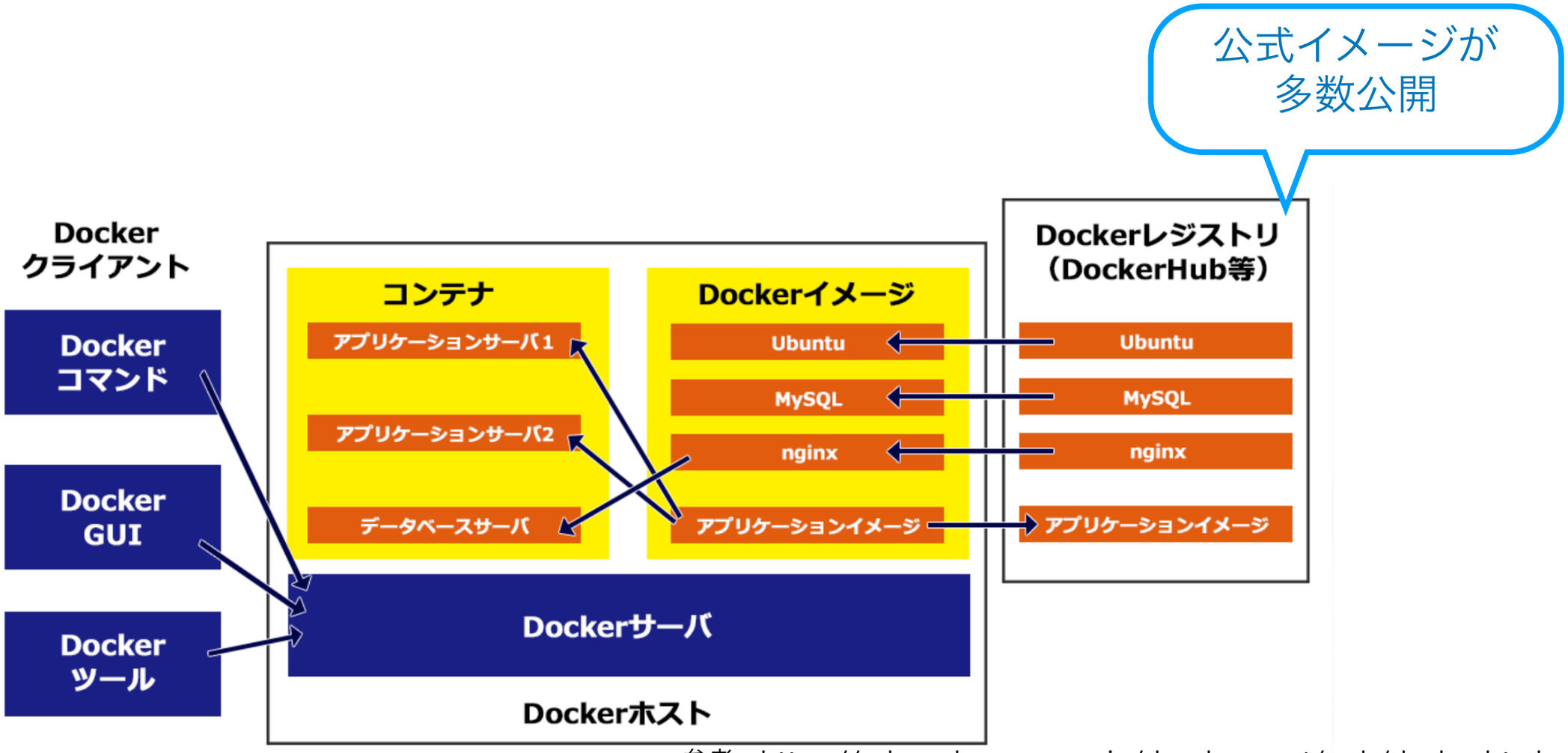
- コンテナ型仮想環境を作成、配布、実行するためのプラットフォーム
  一つのOSを複数コンテナで共有する仕組みで軽量・高速化（詳細は知りません…）
- Dockerイメージをダウンロードしてコンテナを生成
- DockerHubに公式イメージが多数公開されており、環境構築済みのイメージをすぐに利用可能



参考：https://udemy.benesse.co.jp/development/web/docker.html

Dockerについて

- コンテナ型仮想環境を作成、配布、実行するためのプラットフォーム
  一つのOSを複数コンテナで共有する仕組みで軽量・高速化（詳細は知りません…）
- Dockerイメージをダウンロードしてコンテナを生成
- DockerHubに公式イメージが多数公開されており、環境構築済みのイメージをすぐに利用可能



参考：https://udemy.benesse.co.jp/development/web/docker.html

よく使うDockerコマンド、オプション

イメージダウンロード(なければ)　＆
新しいコンテナの生成/起動

```
$ docker run [オプション] イメージ
```

オプション名	詳細
-i, --interactive	コンテナのSTDIN(標準入力)を取得する。 要はコンテナに対する操作ができるようにする。
-t, --tty	tty（擬似端末）を割り当てる。 (一般に上の-iとセットで-itと入力する。)
-d, --detach	コンテナをバックグラウンドで実行する。
--name=""	コンテナ名を割り当てる。
-e, --env=[]	コンテナの環境変数を指定する。
-p, --publish=[ホスト側ポート番号: コンテナ側ポート番号]	コンテナのポートをホスト側に公開する。 ポートフォワーディング（みたいなの）。
-v, --volume=[ホスト側マウント元: コンテナ側マウント先]	ボリューム(データを永続化できる場所、コンテナでの HDDのようなもの)のマウントをする。

詳細：http://docs.docker.jp/engine/reference/commandline/run.html

よく使うDockerコマンド、オプション

実行中のコンテナにログイン

```
$ docker exec [オプション] コンテナ /bin/bash
```

オプション名	詳細
-i, --interactive	コンテナのSTDIN(標準入力)を取得する。 要はコンテナに対する操作ができるようにする。
-t, --tty	tty（擬似端末）を割り当てる。 (一般に上の-iとセットで-itと入力する。)

詳細：http://docs.docker.jp/engine/reference/commandline/exec.html

よく使うDockerコマンド、オプション

コンテナを起動/停止/再起動

```
$ docker start/stop/restart コンテナ
```

コンテナを削除

```
$ docker rm コンテナ
```

コンテナをイメージ化

```
$ docker commit コンテナ 任意のイメージ名
```

イメージをtar.gz形式で保存

```
$ docker save イメージ > 任意のファイル名.tar,gz
```

tar.gz形式のイメージを読み込み

```
$ docker load < イメージのファイル名.tar,gz
```

イメージの削除

```
$ docker rmi イメージ
```

中級者向けDocker(紹介のみ)

Dockerfile

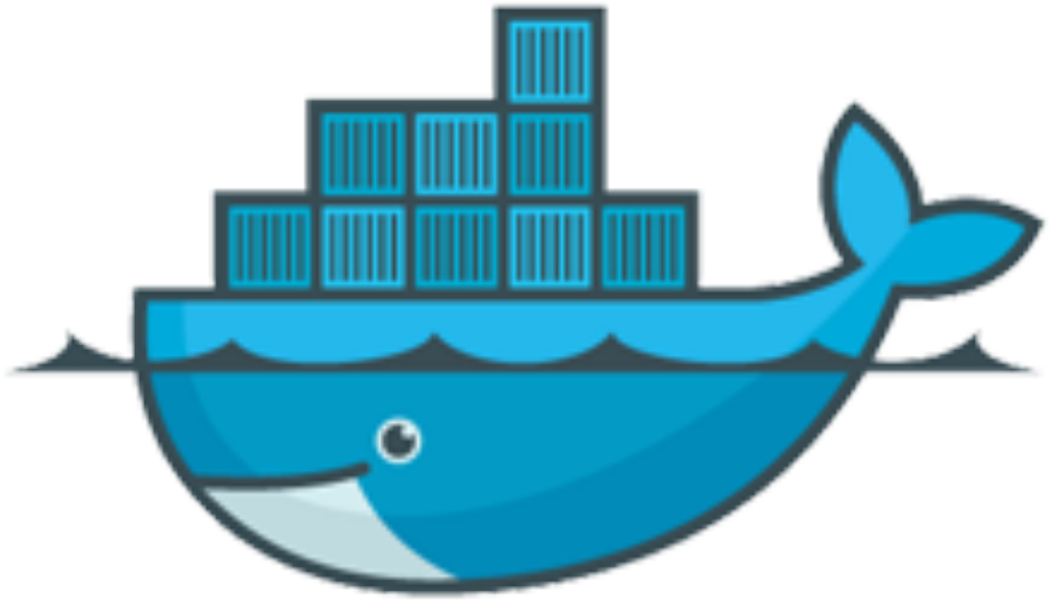
- オリジナルイメージを構築するための命令を書き込んだテキストファイル
- コマンド１つでテキスト中の複数の命令を順次実行し、新規イメージを構築可能。

Docker Compose

- 複数のコンテナを使うDockerアプリケーションを、定義・実行するツール
- コマンド１つで設定した複数のコンテナを作成・起動が可能。

以上！ご静聴ありがとうございました

便利なDockerをお試しあれ…



docker