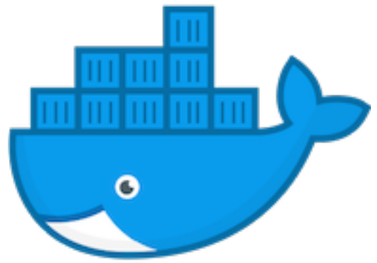


Docker 101

Plan

- Why Docker?
- Terminology
- Ecosystem
- Dockerfile
- CLI
- Tips & Tricks
- Docker-compose
- Tutorial

Why docker?

Data Scientist

SuperComputer

Why docker?

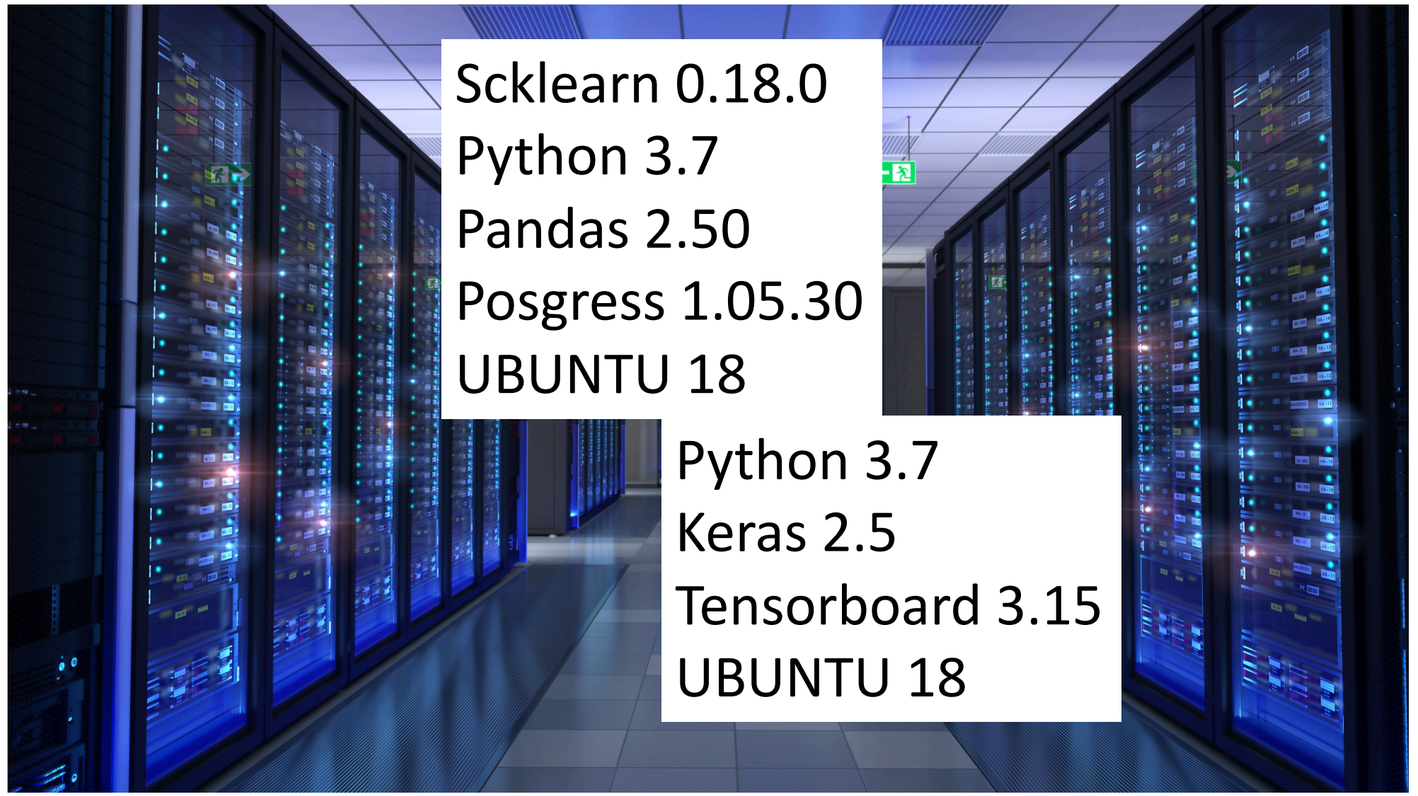
Data Scientist

SuperComputer

Why docker?

Data Scientist

Data Scientist



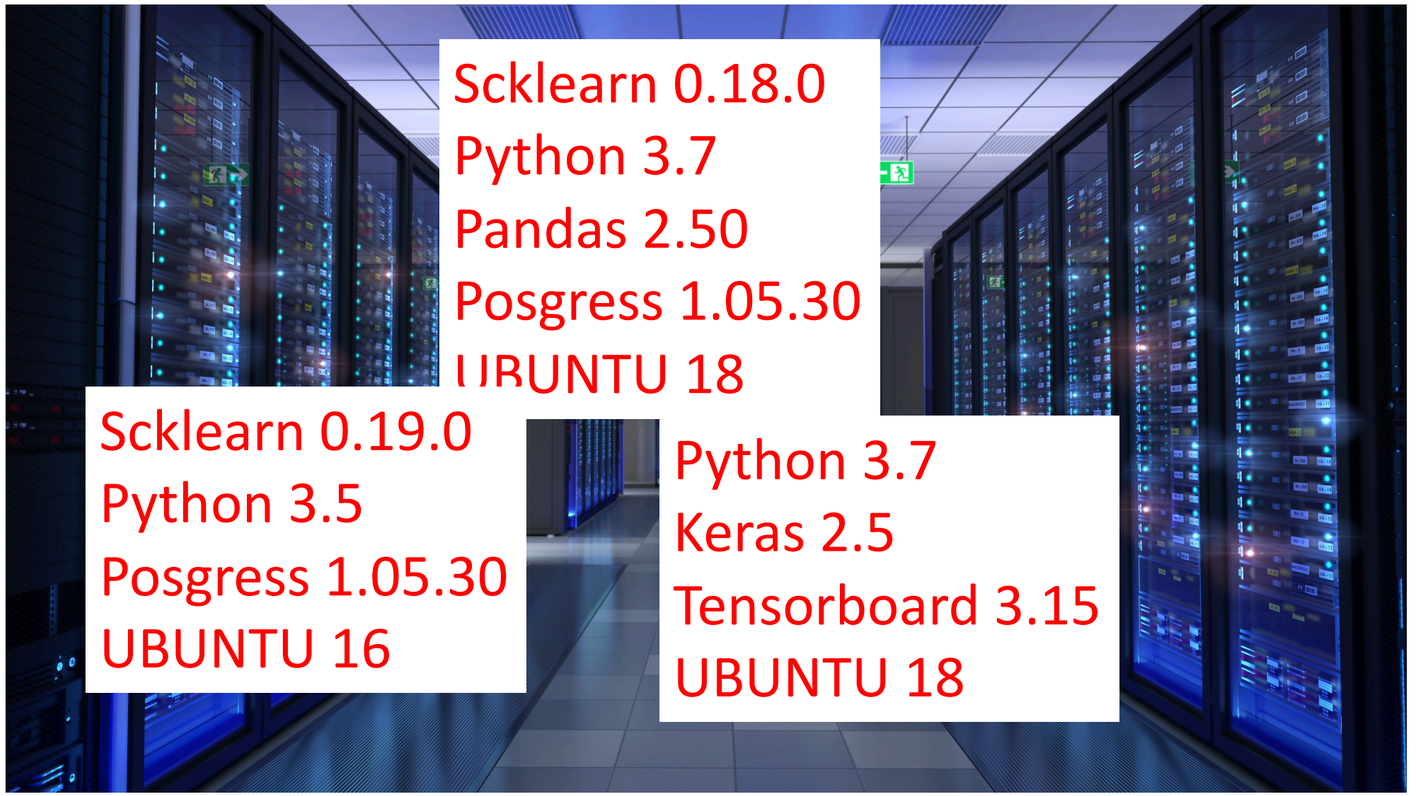
SuperComputer

Why docker?

Data Scientist

Data Scientist

Data Scientist



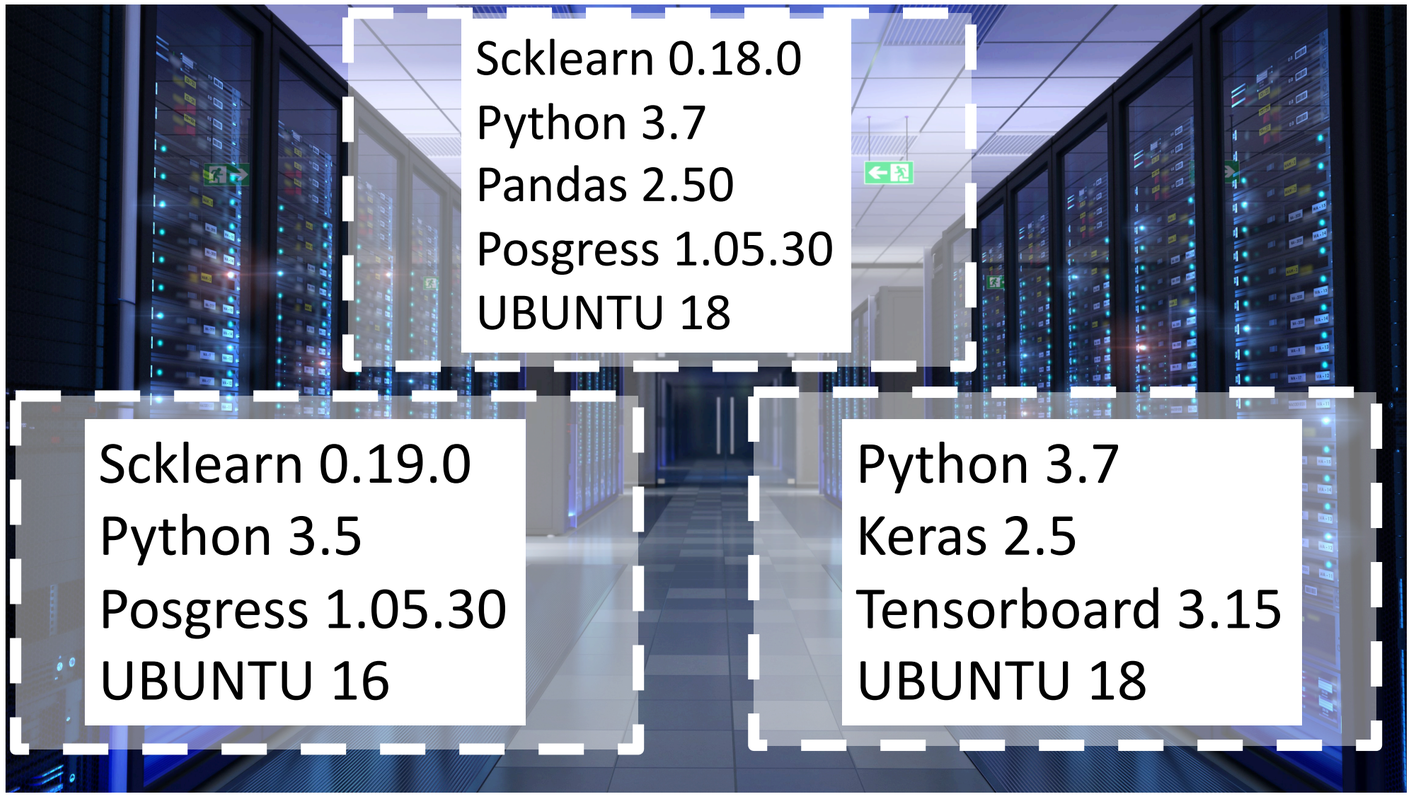
SuperComputer

Why docker?

Data Scientist

Data Scientist

Data Scientist

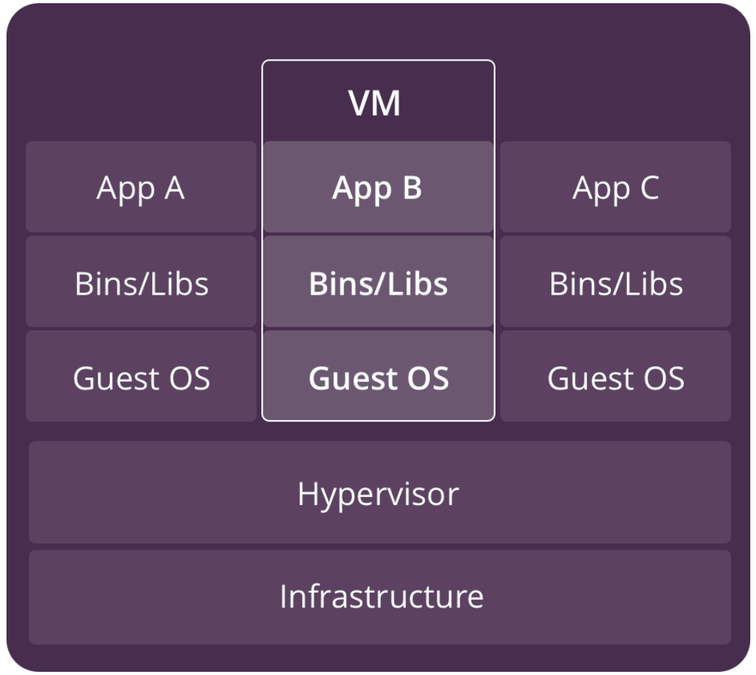


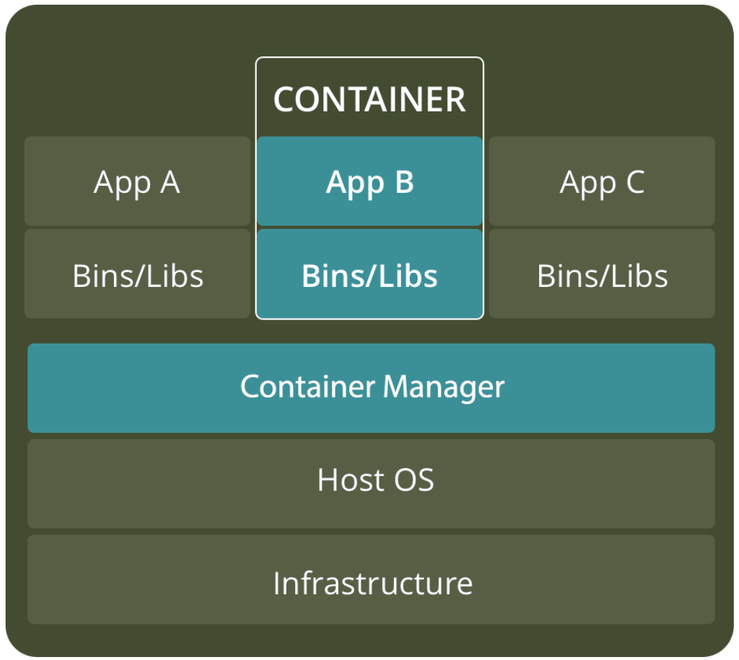
SuperComputer

Why docker?

SuperComputer

Why containers?





Docker terminology

Dockerfile

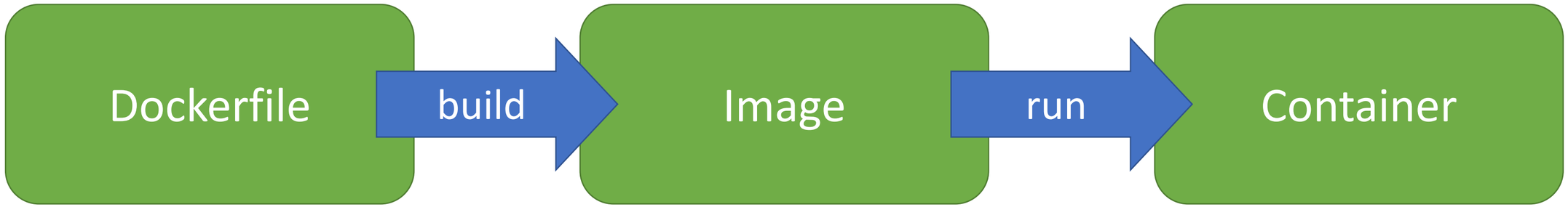
Image

Container

Docker terminology

Dockerfile	Image	Container
Formal class description	Class	Class instance (object)

Docker terminology



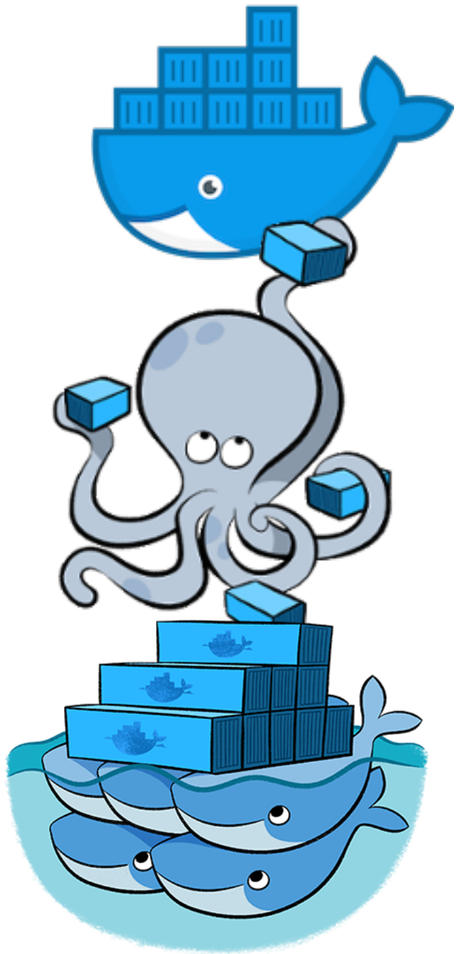
Formal class description

Class

Class instance (object)

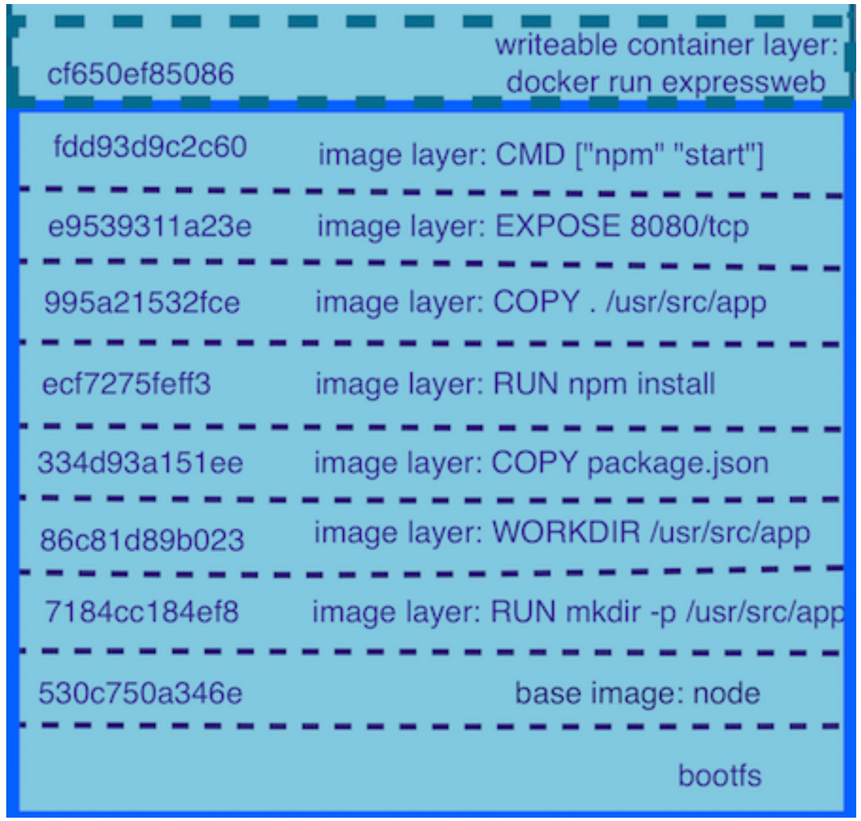
What is docker?

- Docker Containers
- Docker Hub
- Docker Compose
- Docker Swarm
- Docker Cloud



Dockerfile reference

- FROM
- RUN
- ENTRYPOINT / CMD
- COPY / ADD
- WORKDIR



Docker cli basic

```
docker build -t alikhil/my-image:v1 -f Dockerfile .



docker run -p 8080:80 -d alikhil/my-image:v1
```

Logs

```
docker logs [-f] [--tail=N] cntr_id_or_name



docker logs -f nginx
```

Exec

```
docker exec [--it] cntr_name_or_id command



docker exec --it myapp bash
```

Volumes

```
docker run -v host_dir:cntr_dir image-name
```

Port forwarding

```
docker run -p <host-port>:<container-port> image-name
```

Example stateful postgres in docker:

```
docker run -d --rm
    -v $(PWD)/data/pg:/var/lib/postgresql/data
    -e POSTGRES_PASSWORD=1234
    -e POSTGRES_USER=postgres
    -p 5432:5432
    --name pg-app postgres
```

Real example

```
docker run --runtime=nvidia \
  -e NVIDIA_VISIBLE_DEVICES=<Parameter> \
  -e LOGIN=$(whoami) \
  --cap-add SYS_ADMIN \
  --cap-add DAC_READ_SEARCH \
  -e LOGIN_UID=$(id -u) \
  -e LOGIN_GID=$(id -g) \
  -v $HOME/.ssh/authorized_keys:/mnt/authorized_keys:ro \
  -v $HOME/work:/work/work \
  --name $(whoami) \
  -it \
  --memory 64g \
  -p 8888 \
  -p 6006 \
  -p 22 \
  registry.skbkontur.ru/research/anaconda3-cuda10.0
```

Trick #1

Speed up builds by:

- caching layers
- use .dockerignore

```
1   FROM node:8.12
2
3   RUN mkdir app
4   WORKDIR /app
5
6   ADD package.json
7
8   RUN npm install
9
10  ADD . .
11
12  CMD ["node", 'index.js']
```

https://blog.playmoweb.com/speed-up-your-builds-with-docker-cache-bfed14c051bf

Trick #2

Decrease image size by:

- using alpine images:
- combining several RUN layers into one

https://hackernoon.com/tips-to-reduce-docker-image-sizes-876095da3b34

Trick #3

Save disk space in your host machine by:

- running containers with `--rm` argument
- or remove them later with:

```
docker rm $(docker ps  -q -f 'status=exited')
```

- prune old volumes `docker volume prune`
- delete unused images - `docker rmi $(docker images -q -f “dangling=true")`

Trick #4

Docker run can become very complicated. Like this:

```
docker run -d --rm
    -v $(PWD)/data/pg:/var/lib/postgresql/data
    -e POSTGRES_PASSWORD=1234
    -e POSTGRES_USER=postgres
    -p 5432:5432
    --name pg-app postgres
```

Solution: use `docker-compose`

docker-compose

```yaml
version: '3'

services:
  redis:
    image: redis:5.0
    volumes:
      - $PWD/redis_data:/data
    ports:
      - 6379:6379

  postgres:
    image: postgres:9.6
    environment:
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=1234
      - POSTGRES_DB=postgres
    ports:
      - 5433:5432
    volumes:
      - $PWD/postgres_data:/var/lib/postgresql/data
```

Resources

- https://www.backblaze.com/blog/vm-vs-containers/
- https://github.com/alikhil/docker-101-tutorial