CONTAINERIZATION w/ DOCKER

By Jonas M. Hansen

WHO AM I? - Jonas M. Hansen

Intern and student helper at Systematic working on SitaWare Headquarters.
Aug. 2016 - May 2017

Master of Science in Computer Science from California State University Channel Islands
2018 - Present

Bachelors of Engineering in Information Technology with a specialization in Software from Aarhus University
2014 - 2018

Higher Technical Examination Programme with focus on Project Management and Game Development from VidenDjurs
2011 - 2014

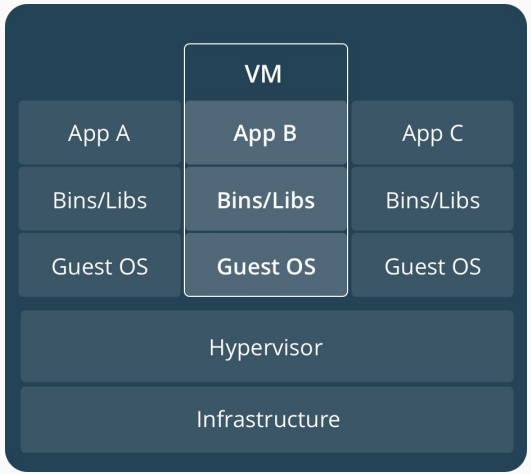
OUTLINE

- The Problems
- Usual Solution
- The Docker Way
- Container Management
- Maybe a demo?

THE PROBLEM

- Software is complicated
  - Multiple dependencies
  - Different OS’s to support
  - Artifacts even after uninstall
- Production environment vs development env.
- Scalability

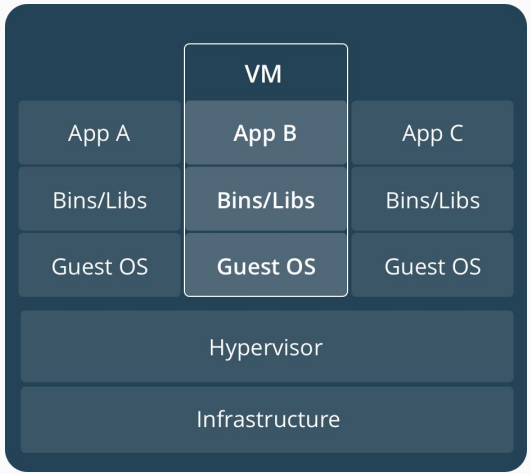
OLD SOLUTION - VIRTUALIZATION



- Sandboxes application
- Resource heavy

Image source: https://docs.docker.com/

WHAT IS A CONTAINER?



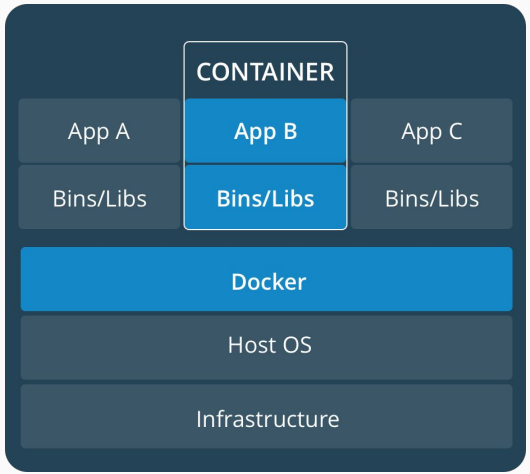


Image source: https://docs.docker.com/

WHAT IS A CONTAINER?

- Created from a Image
  - Base Image / OS
  - Dependencies
  - Application
- Think it’s running as its own server

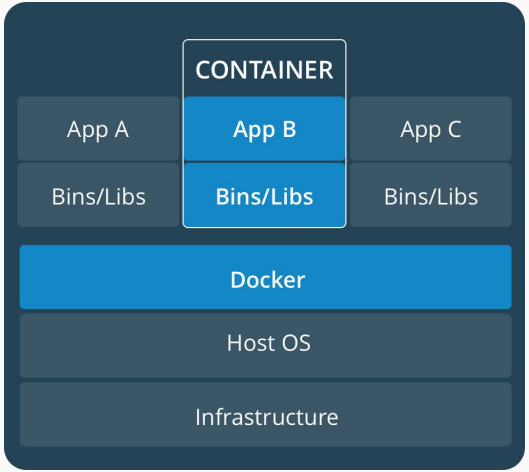


Image source: https://docs.docker.com/

DOCKER ARCHITECTURE

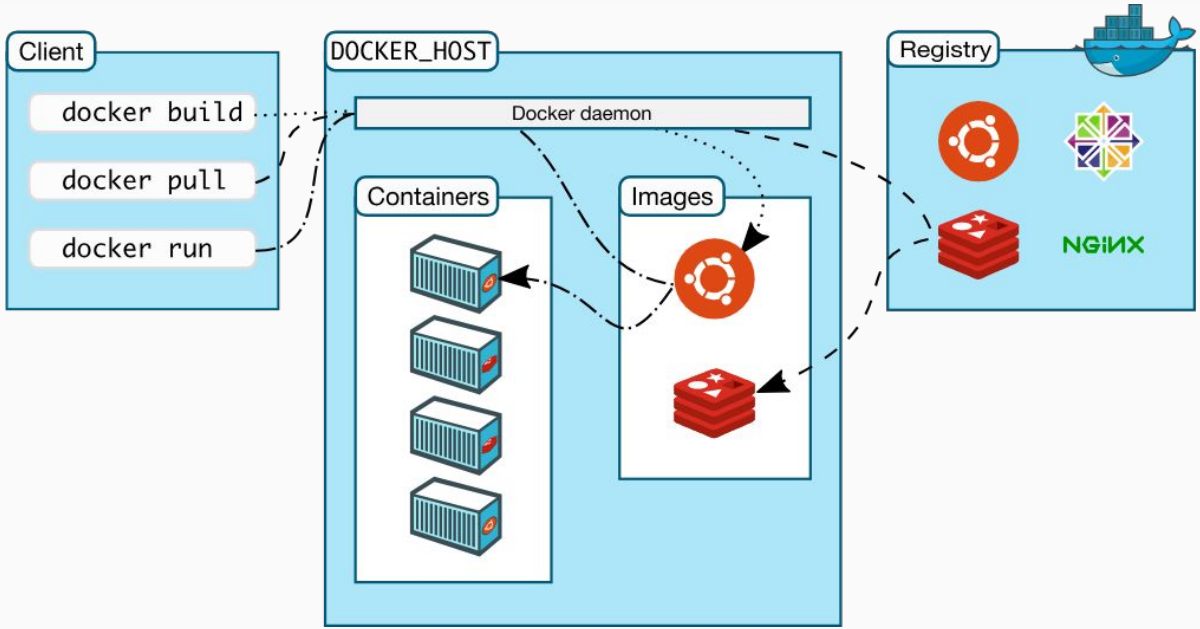


Image source: https://docs.docker.com/

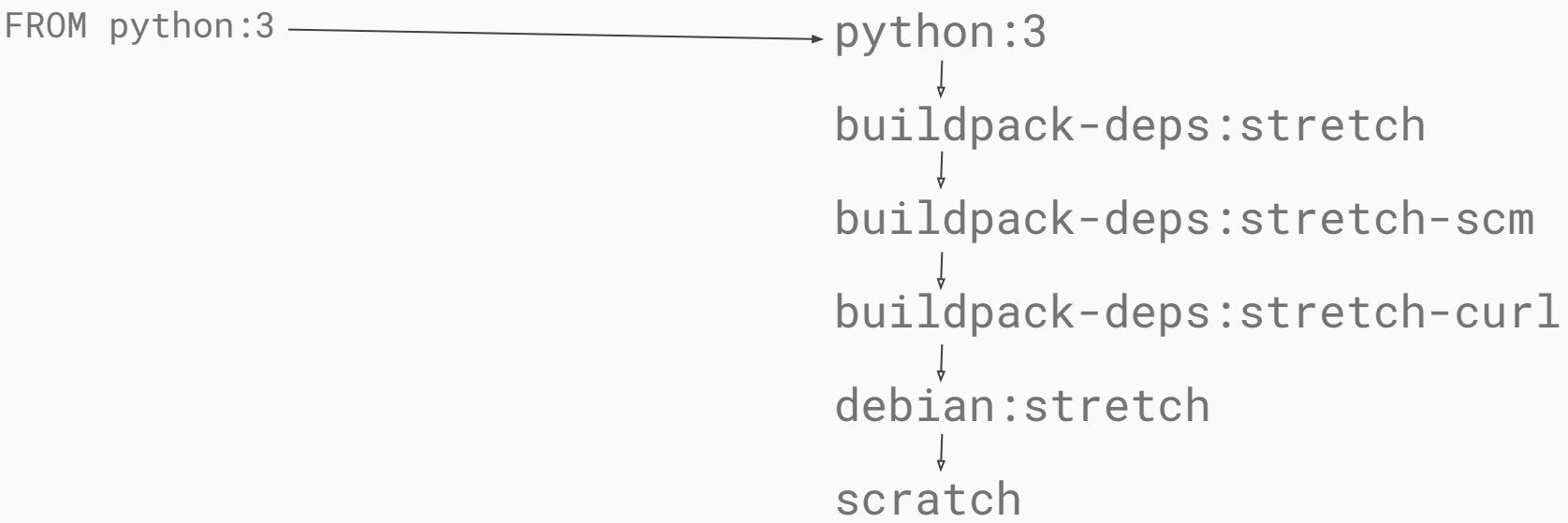
CREATING A IMAGE

Dockerfile:

```
FROM python:3
WORKDIR /usr/src/app
COPY . .
EXPOSE 8000
CMD [ "python3", "./hello_world_web.py" ]
```

```
docker build -t [owner/]<image>[:tag] .
```

LAYERING


FROM python:3
python:3
buildpack-deps:stretch
buildpack-deps:stretch-scm
buildpack-deps:stretch-curl
debian:stretch
scratch

CREATING A CONTAINER

```
docker run args [owner/]<image-name>[:tag]
```

Publish port: `-p <hostport>:<containerport>`
Mount volume: `-v <hostvolume>:<containervolume>`

Example:

```
docker run cs599:latest -p 8000:8000
```

MANAGING MULTIPLE CONTAINERS

Docker-compose!

```
docker-compose up

docker-compose down
```

docker-compose.yml:

```
version: '3'
services:
  web:
    build: ./example_app
    ports:
    - 9000:8000
  phpmyadmin:
    image: phpmyadmin/phpmyadmin
    ports:
     - 9001:80
```

QUICK DEMO?

WHAT DID WE COVER?

- Basic docker architecture
- Images
- Containers
- Management

QUESTIONS?

Slides and example:
https://github.com/JonasMH/cs599_presentation

Want to start using Docker?
https://docs.docker.com/get-started/