FROM VIRTUALISATION TO CONTAINERISATION

SDD - Data Engineering
https://supaerodatascience.github.io/deep-learning/

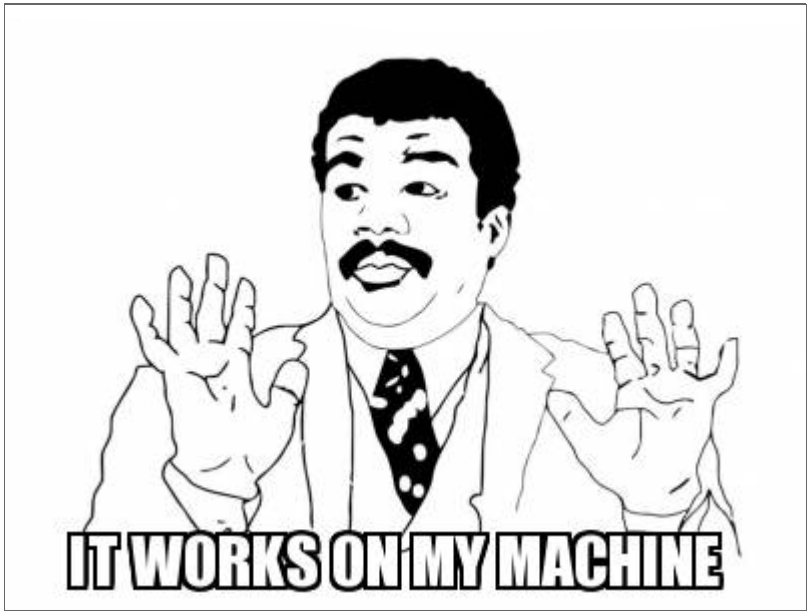
OUTLINE

- **Presentation** (45m)
- **Self-paced Workshop** (2h15)
- BE Cloud & Docker (23/11)
- To be continued with Orchestration & Deployment (08/01)

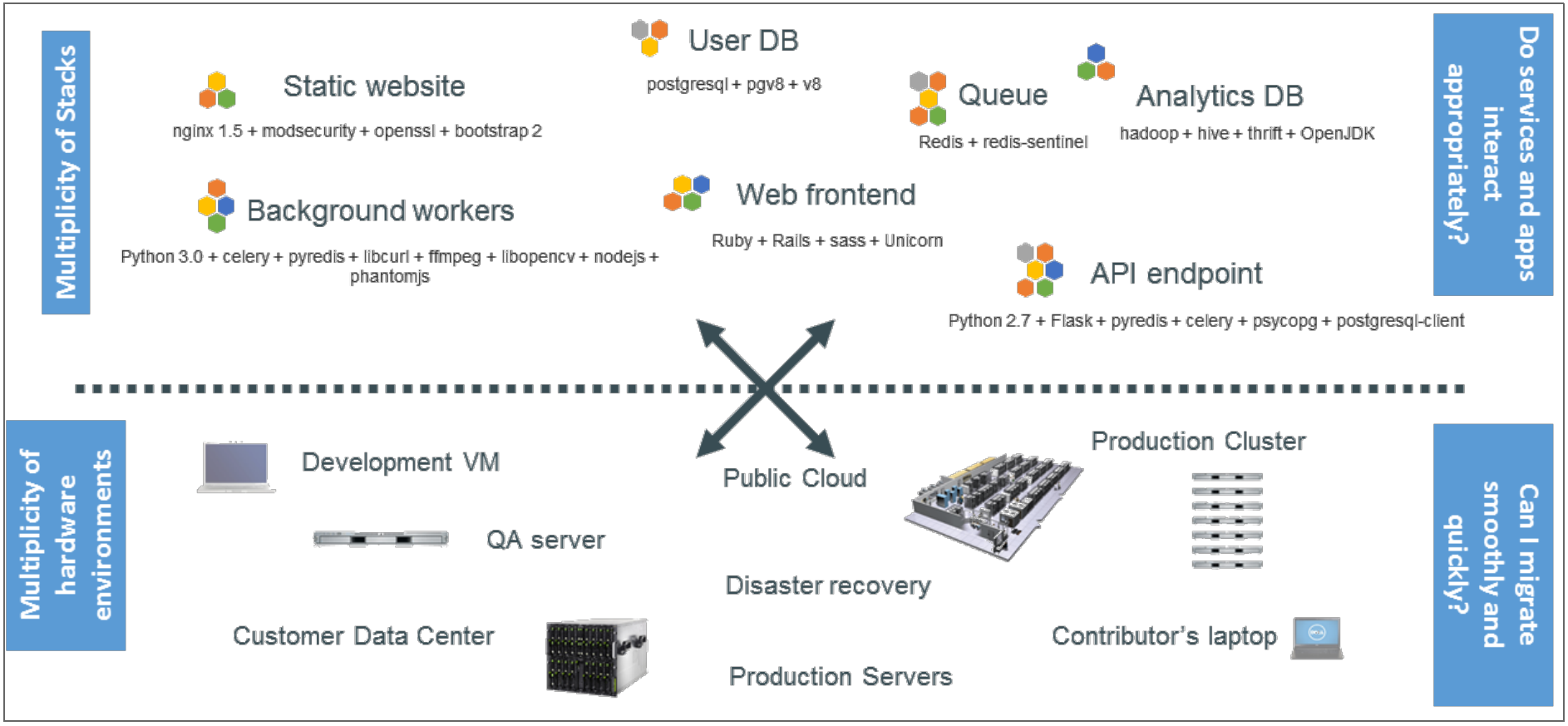
THIS CLASS WILL BE SUCCESSFUL IF YOU UNDERSTAND

- why we need a tool like docker
- the basics of docker (containers, images)
- the basics of a container registry
- how to pull an image and run a container
- what a Dockerfile looks like

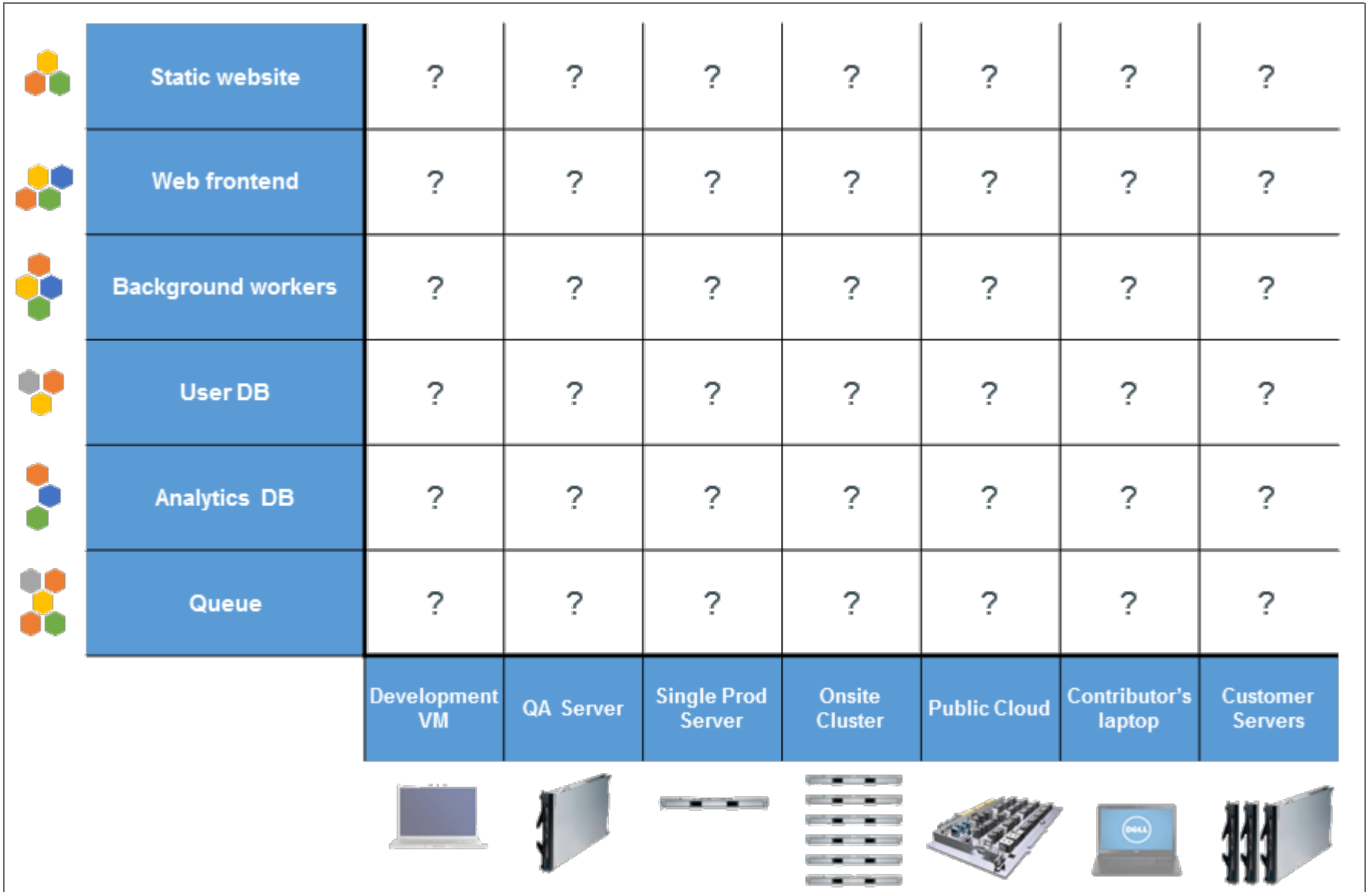
THE NEED FOR CONTAINERS IN SOFTWARE



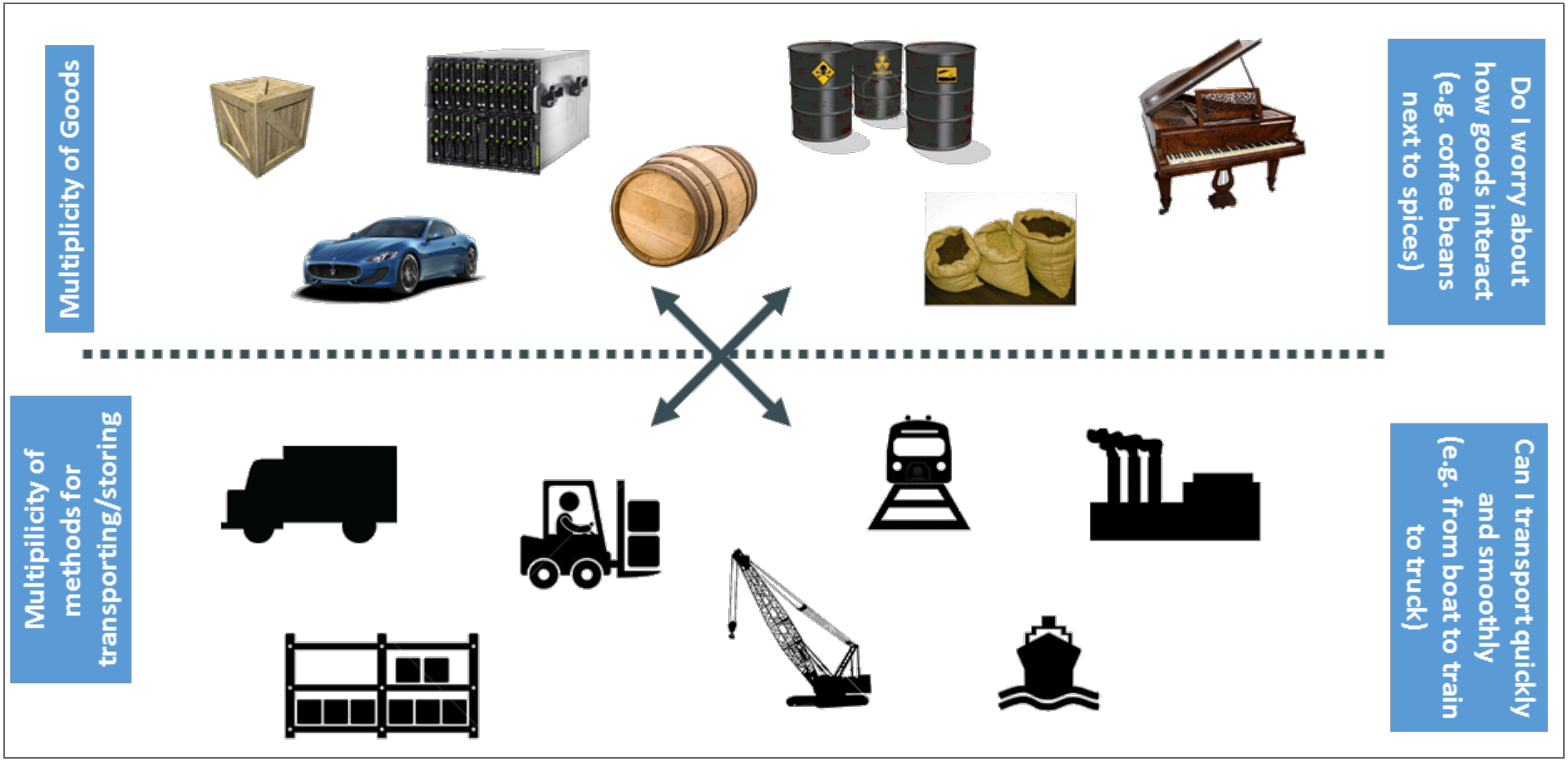
IT MULTIMODALITY



THE MATRIX FROM HELL



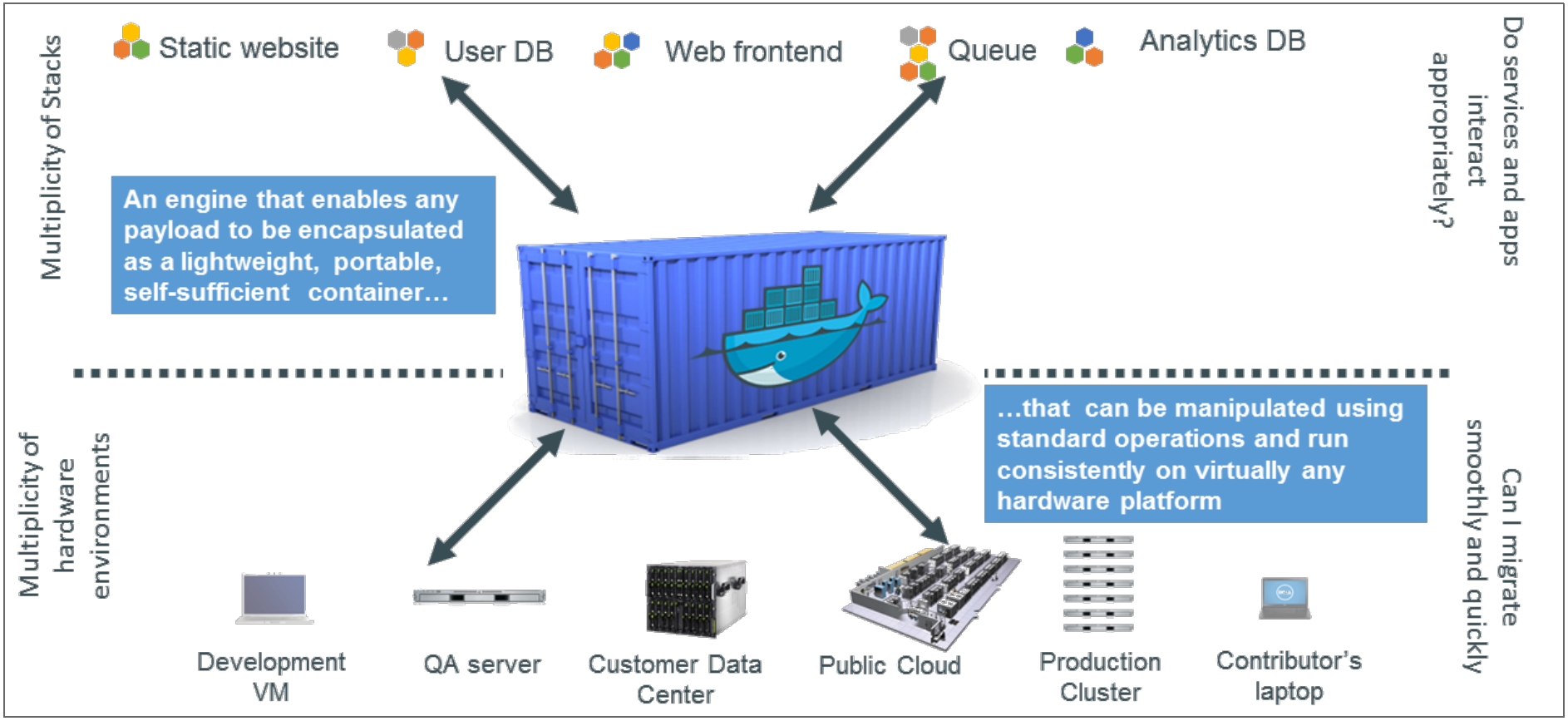
ANALOGY


Multiplicity of Goods
Do I worry about how goods interact (e.g. coffee beans next to spices)
Multipillicity of methods for transporting/storing
Can I transport quickly and smoothly (e.g. from boat to train to truck)

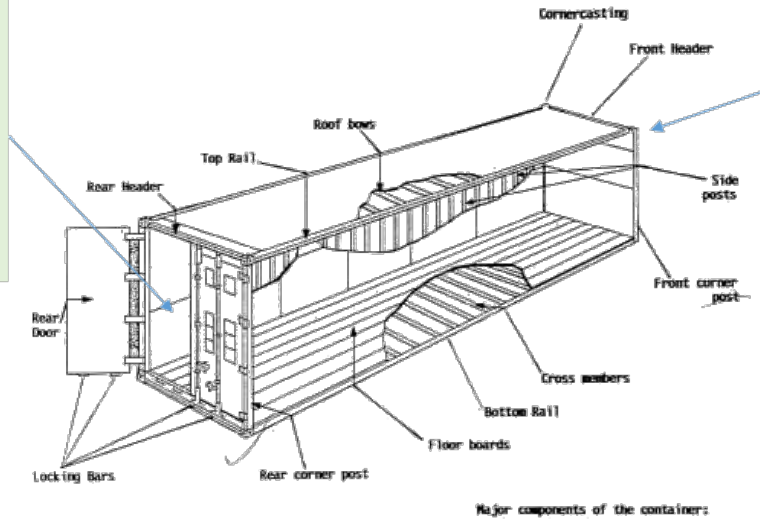
SOLUTION ?


Multiplicity of Goods
A standard container that is loaded with virtually any goods, and stays sealed until it reaches final delivery.
Do I worry about how goods interact (e.g. coffee beans next to spices)
…in between, can be loaded and unloaded, stacked, transported efficiently over long distances, and transferred from one mode of transport to another
Multiplicity of methods for transporting/storing
Can I transport quickly and smoothly (e.g. from boat to train to truck)

SOLUTION !

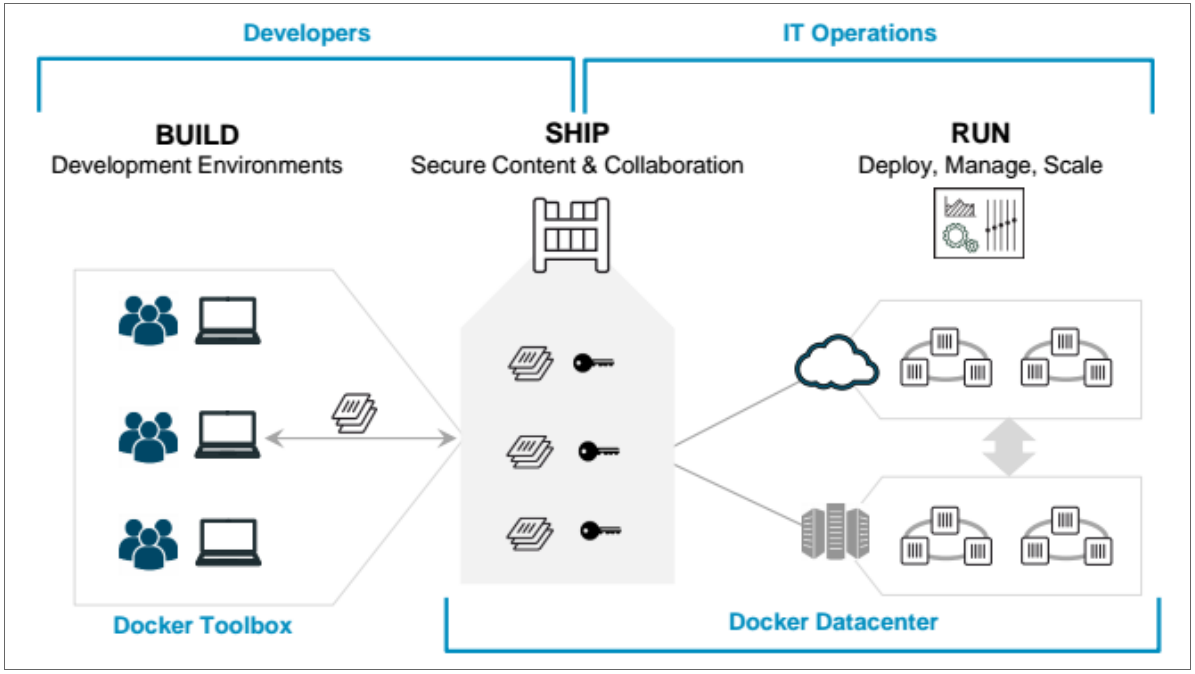

Multiplicity of Stacks
Static website
User DB
Web frontend
Queue
Analytics DB
Do services and apps interact appropriately?
An engine that enables any payload to be encapsulated as a lightweight, portable, self-sufficient container…
…that can be manipulated using standard operations and run consistently on virtually any hardware platform
Multiplicity of hardware environments
Can I migrate smoothly and quickly
Development VM
QA server
Customer Data Center
Public Cloud
Production Cluster
Contributor's laptop

- **Dan the Developer**
  - Worries about what's "inside" the container
    - His code
    - His Libraries
    - His Package Manager
    - His Apps
    - His Data
  - All Linux servers look the same



Major components of the container:

- **Oscar the Ops Guy**
  - Worries about what's "outside" the container
    - Logging
    - Remote access
    - Monitoring
    - Network config
  - All containers start, stop, copy, attach, migrate, etc. the same way


Developers
IT Operations
BUILD
Development Environments
SHIP
Secure Content & Collaboration
RUN
Deploy, Manage, Scale
Docker Toolbox
Docker Datacenter

DOCKER

Docker is **a** solution that **standardizes** packaging and execution of software in isolated environments (**containers**) that share resources and can communicate between themselves

> *Build, Share, and Run Any App, Anywhere*

Docker

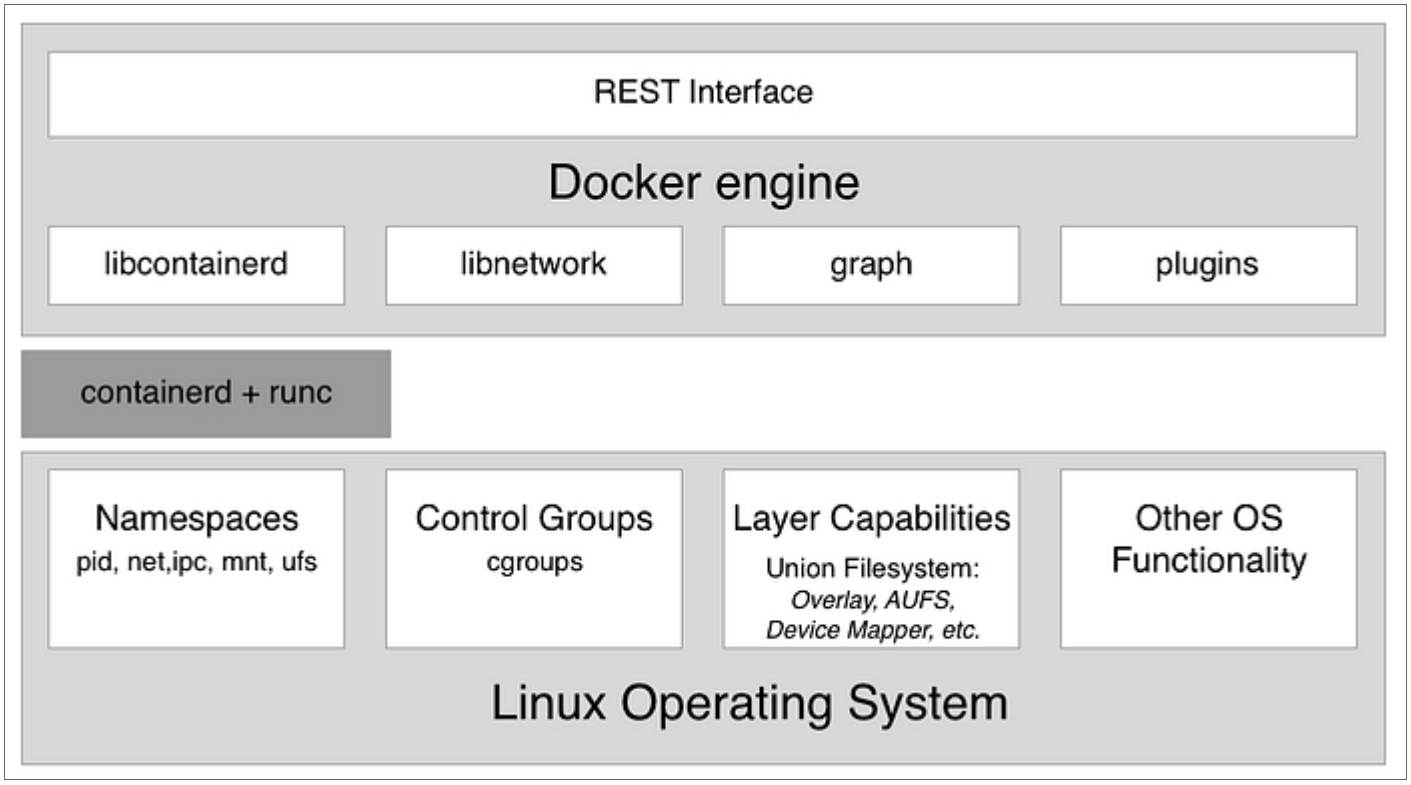
- Created in 2013
- Open Source
- Not a new idea but set a new standard
- Docker is a company built around its main product (Docker Engine)
- in charge of dev of everything docker + additional paid services (Docker hub...)

Docker is not the only solution for containers

https://chimeracoder.github.io/docker-without-docker/#30

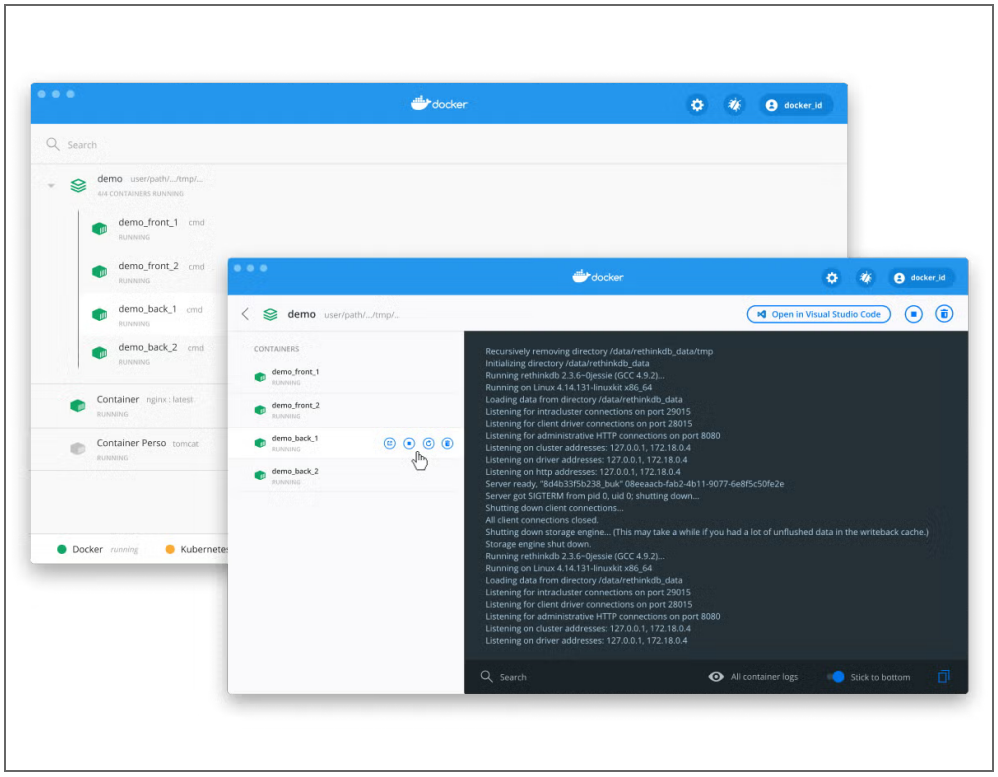
https://podman.io/

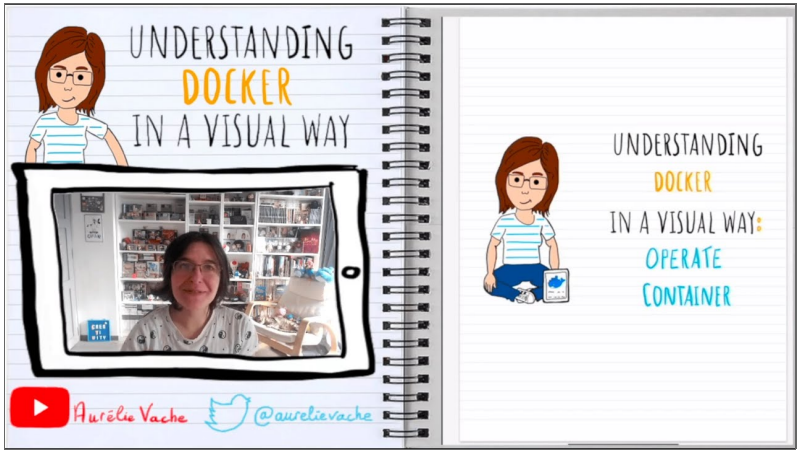
Docker is some fancy tech over linux kernel capabilities (containers)



more info

But Docker is available on **Windows and MacOS** !





https://www.youtube.com/c/AurelieVache/videos

CONTAINERS OR VIRTUAL MACHINES

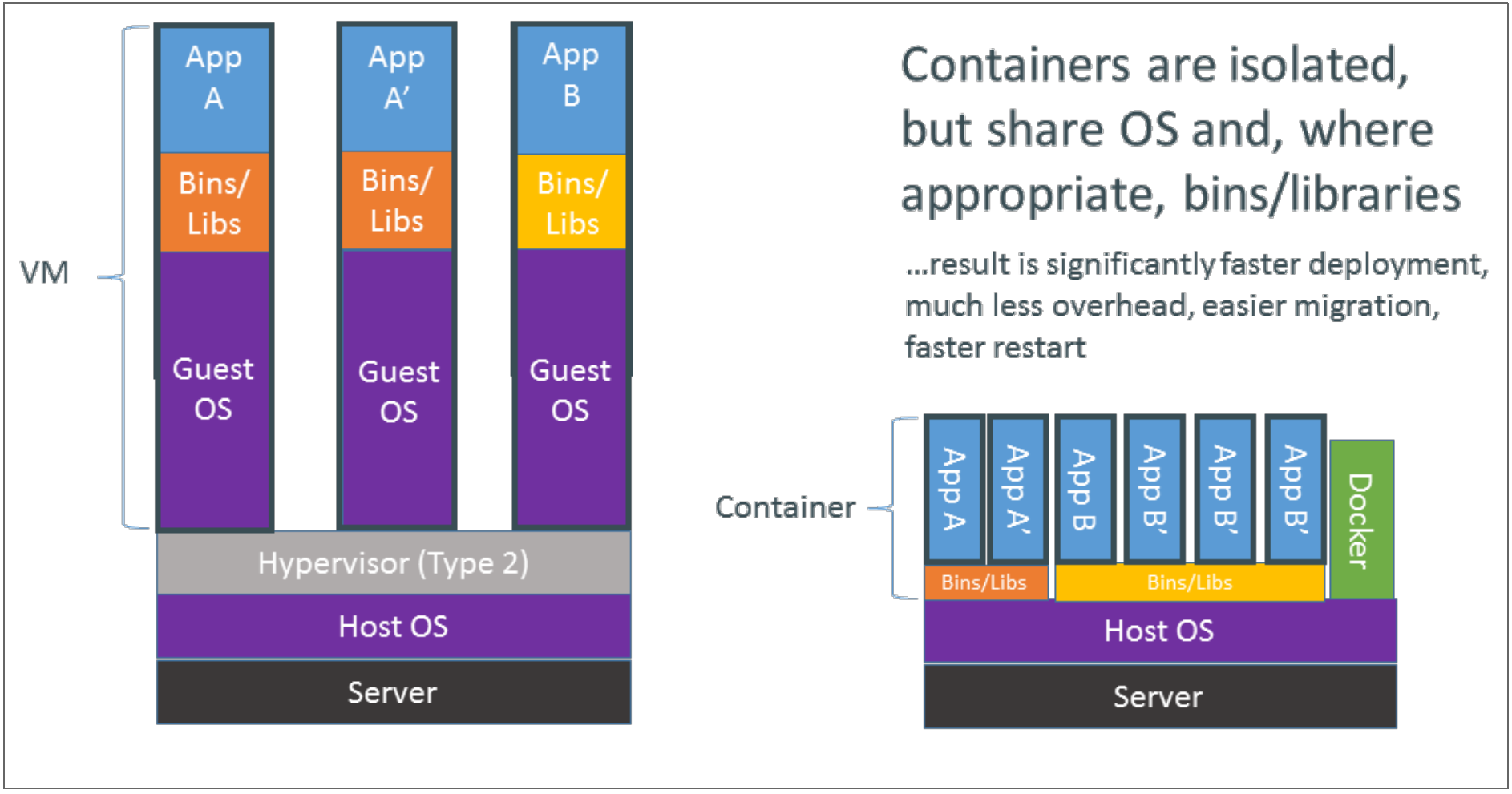
SIMILARITIES

- Isolated environments for applications
- Movable between hosts

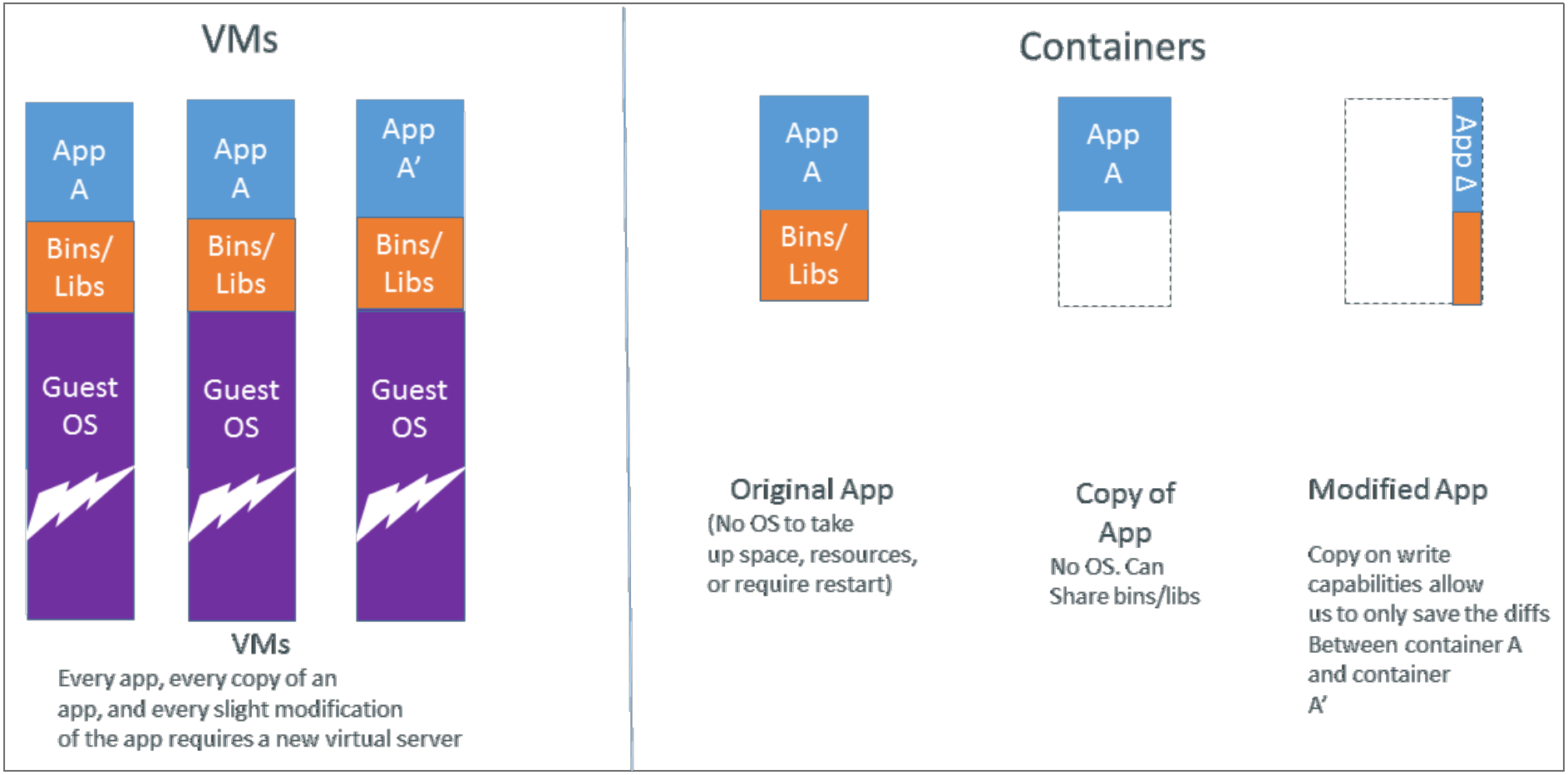
DRAWBACKS OF VMS

- VM Contains full OS at each install => Install + Resource overhead
- VM needs pre-allocation of resource for each VM (=> Waste if not used)
- Communication between VM <=> Communication between computers

CONTAINER VS VIRTUAL MACHINE



WHY ARE DOCKER CONTAINERS LIGHTWEIGHT ?



CONTAINER VS VIRTUAL MACHINE, AN ANALOGY

A Colony of Bungalows

On Premise Deployment:

1) Every app is hosted on a separate sever (bungalow)
2) No optimization of resources at all.
3) No shared services used.
4) High maintenance

An Apartment Complex

Virtualization:

1) Some resources or amenities shared (hyper-visor, host -OS) by apps
2) Some amenities are still app specific (guest OS)
3) Moderate maintenance needed.

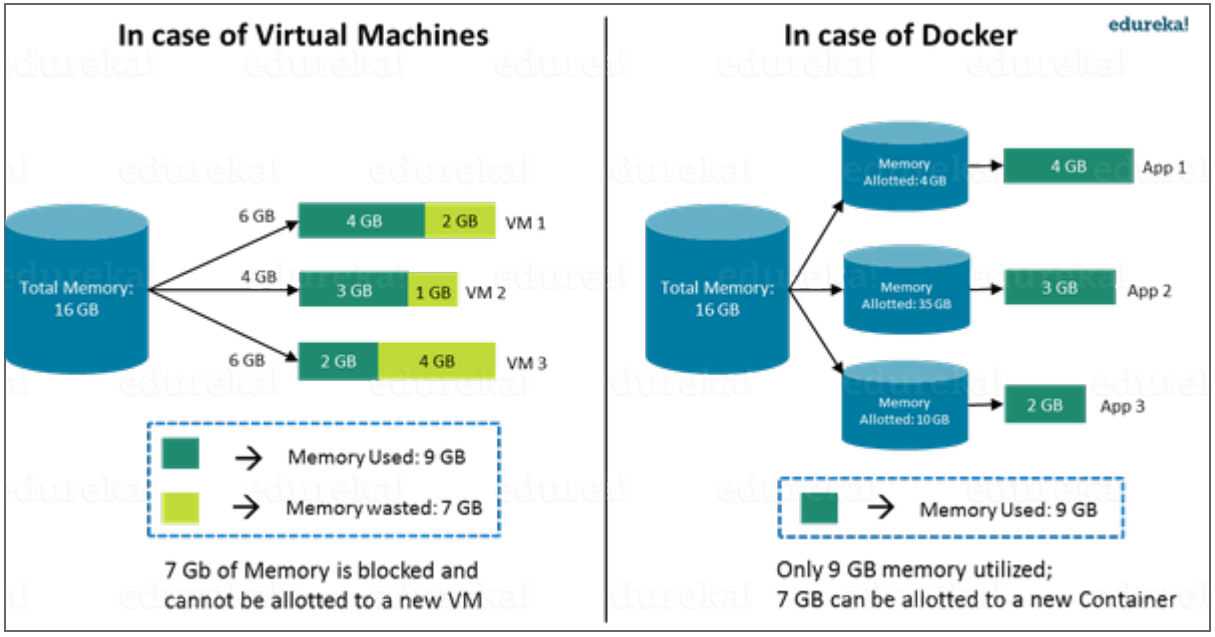
A Hotel

Containerization:

1) Most amenities are shared
2) Guests have only a subset of amenities to use
3) Hotel has better control over managing resources

RESOURCES ALLOCATION IN CONTAINERS

- Containers share underlying OS / Kernels
- The container engine can allocate resources (CPU, Storage, RAM) on the fly (!= VM)
- GPU is way easier to manage / share with containers



SOME DRAWBACKS OF CONTAINERS

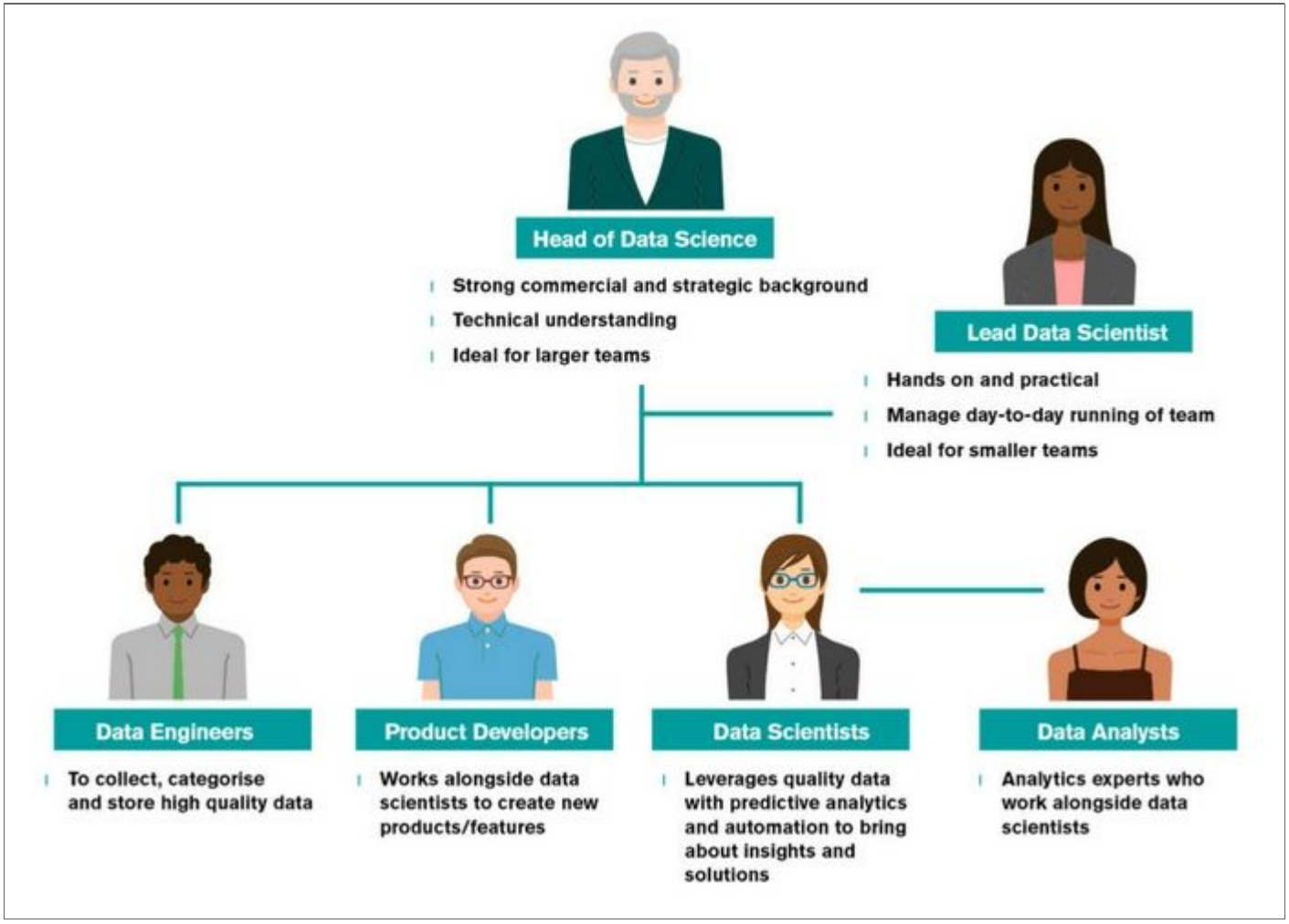
- Containers are based on linux tech

(Docker makes Windows container possible though)

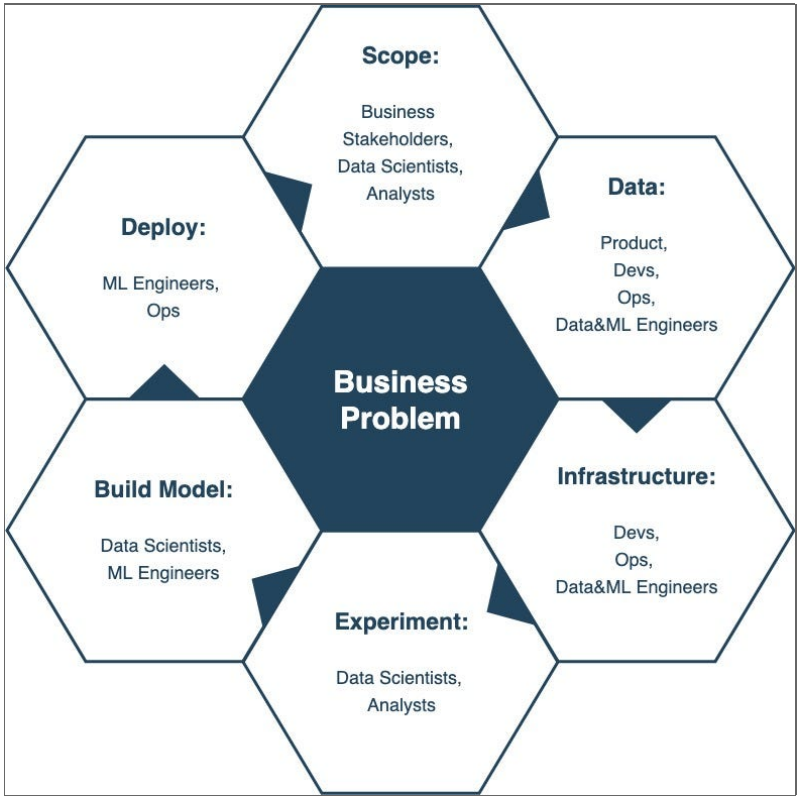
- Isolation is not perfect since containers share underlying kernels (security and stability)

CONTAINERS FOR DATA SCIENCE

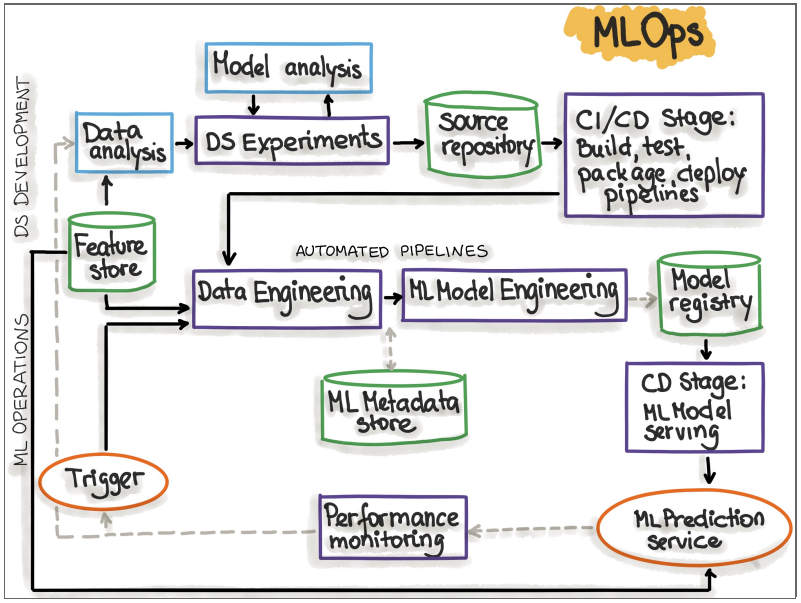
MULTIPLE PEOPLE



COMPLEX WORKFLOWS



MULTIPLE COMPONENTS



DATA SCIENCE IS ABOUT REPRODUCIBILITY

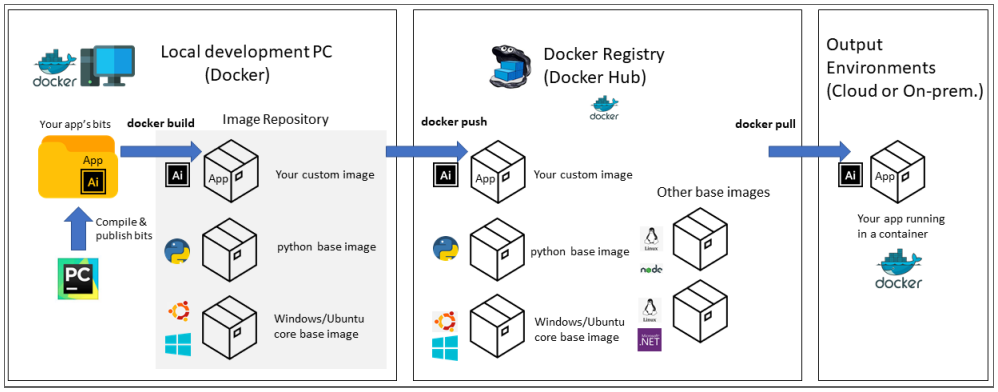
- Experimental science
- Communicating results
- Hands-out to other teams
- Deployment and versioning of models

SO... CONTAINERS ?

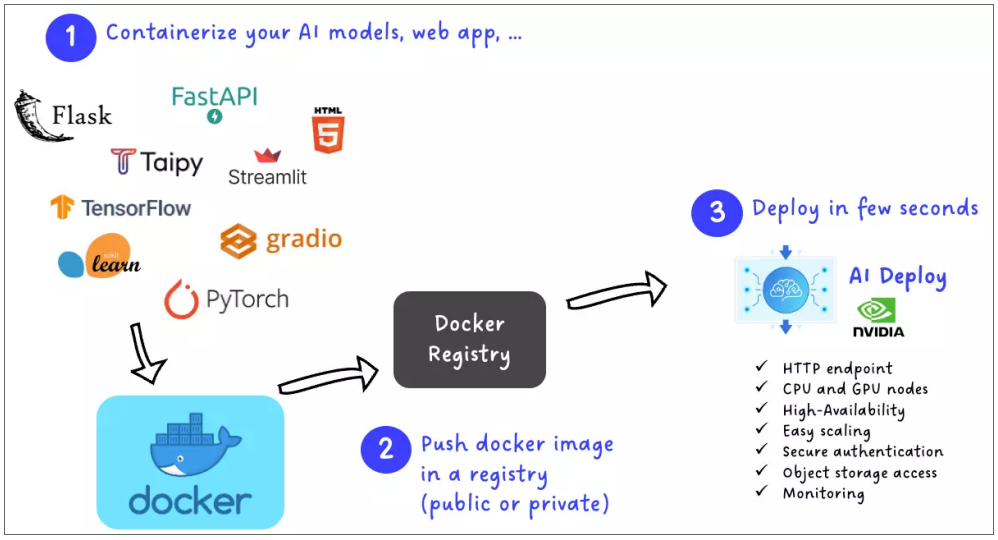
- ... for deployment
- ... for standardized development environments
- ... dependency management
- ... for complex / large scale workflows

~~it works on my notebook !~~ *here's the model ready to run !*

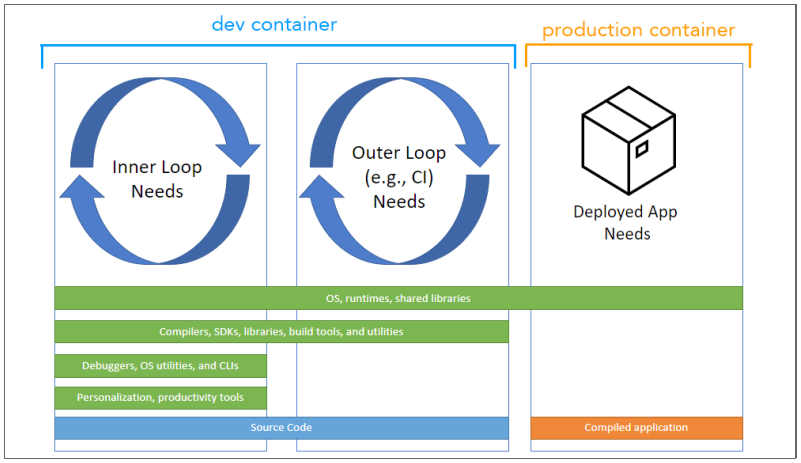
Build, Ship, Run in Data Science



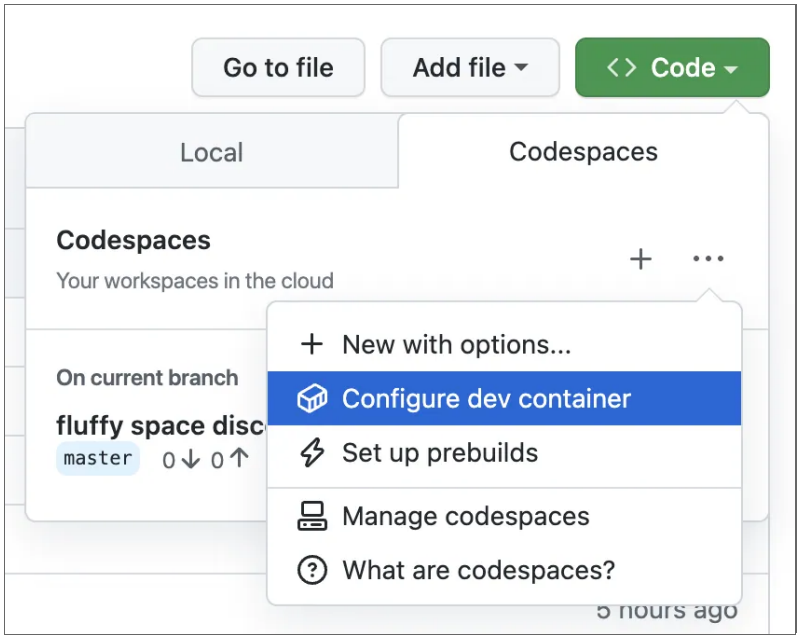
Deployment


1 Containerize your AI models, web app, ...
Flask
FastAPI
HTML
Taipy
Streamlit
TensorFlow
gradio
learn
PyTorch
docker
Docker Registry
2 Push docker image in a registry (public or private)
3 Deploy in few seconds
AI Deploy
NVIDIA
✓ HTTP endpoint
✓ CPU and GPU nodes
✓ High-Availability
✓ Easy scaling
✓ Secure authentication
✓ Object storage access
✓ Monitoring

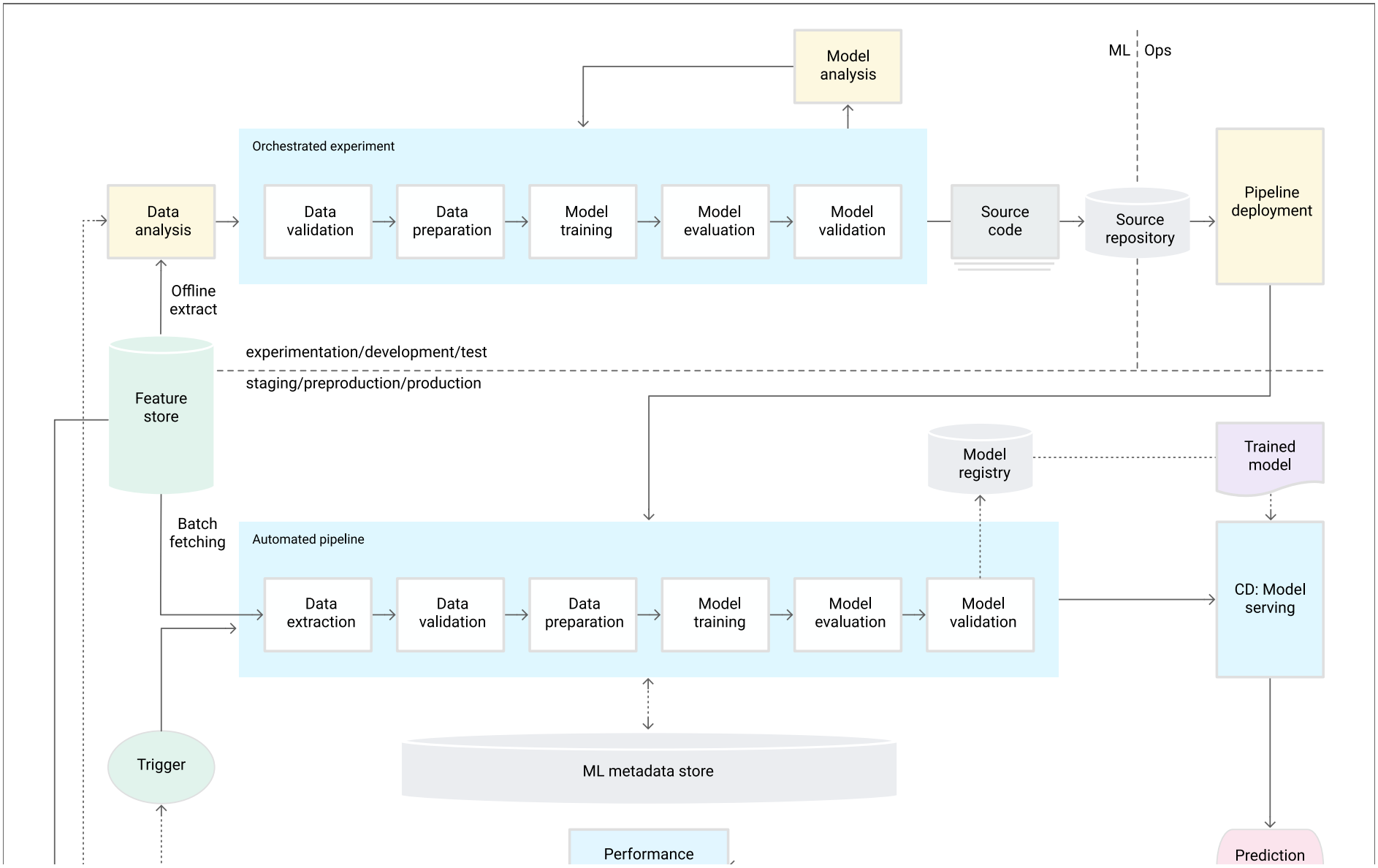
Reproducible development environment



Codespace is actually a container...

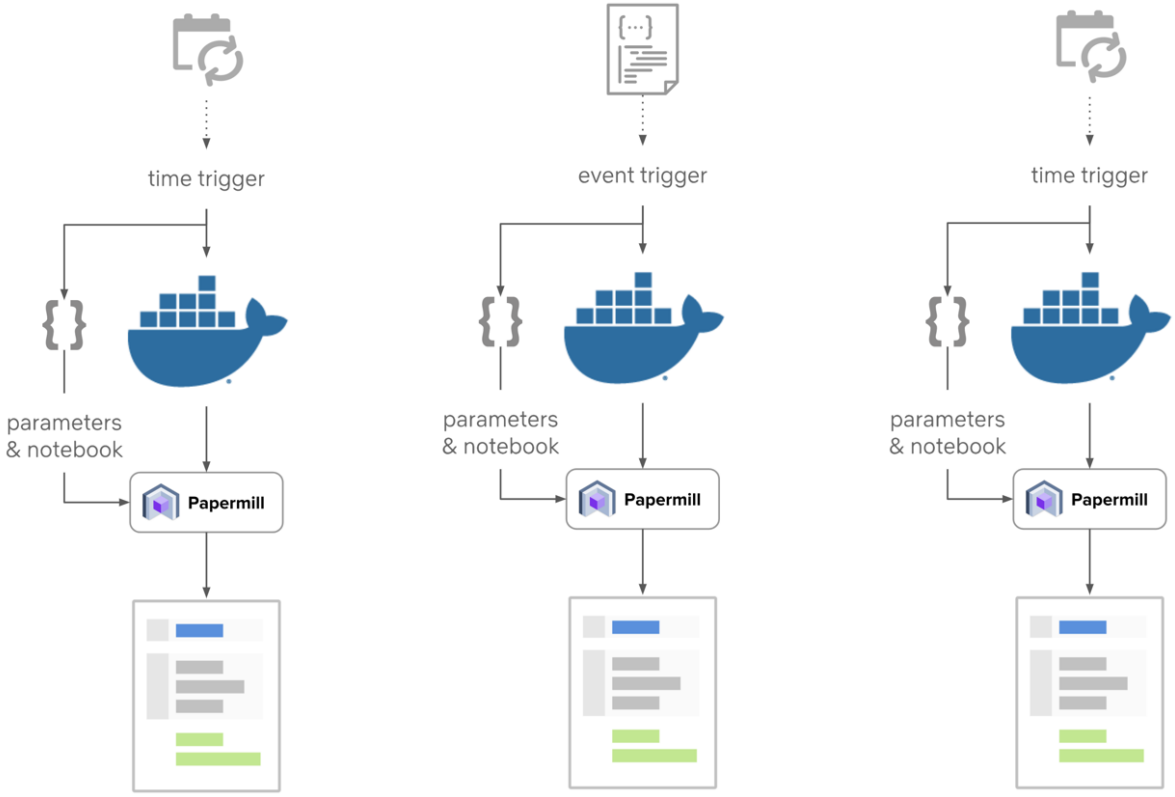


Machine Learning **at scale** !

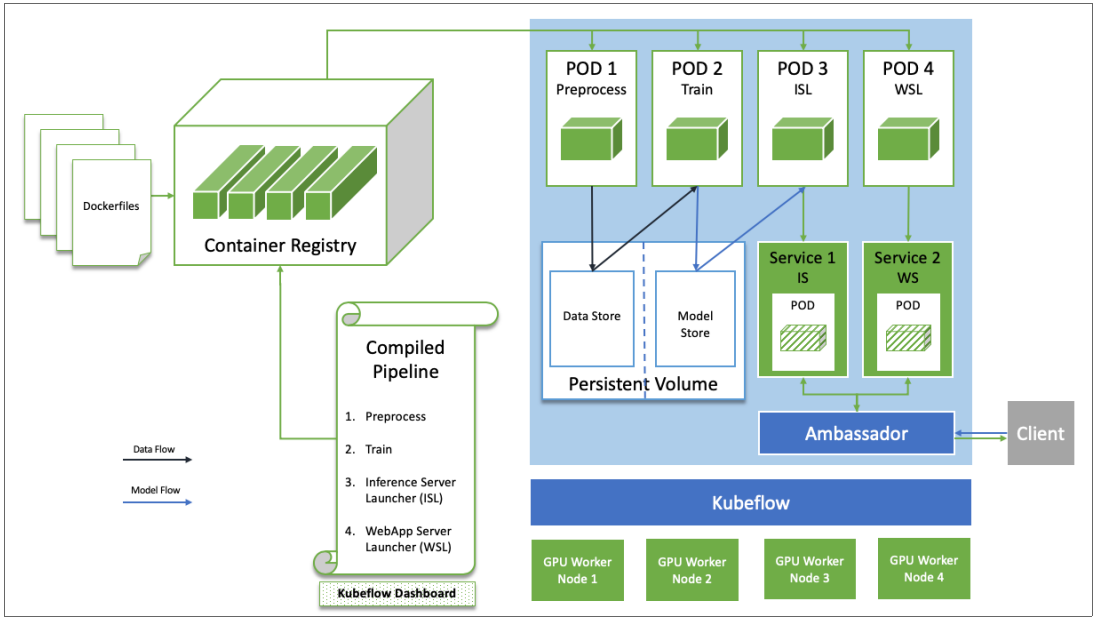



monitoring
service

Netflix and notebook scheduling

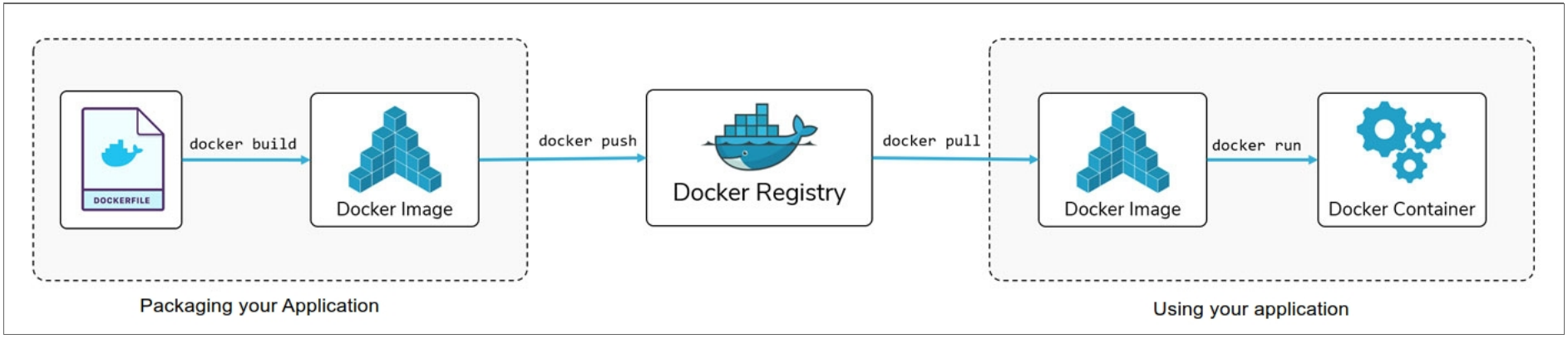


https://www.kubeflow.org/



https://polyaxon.com/

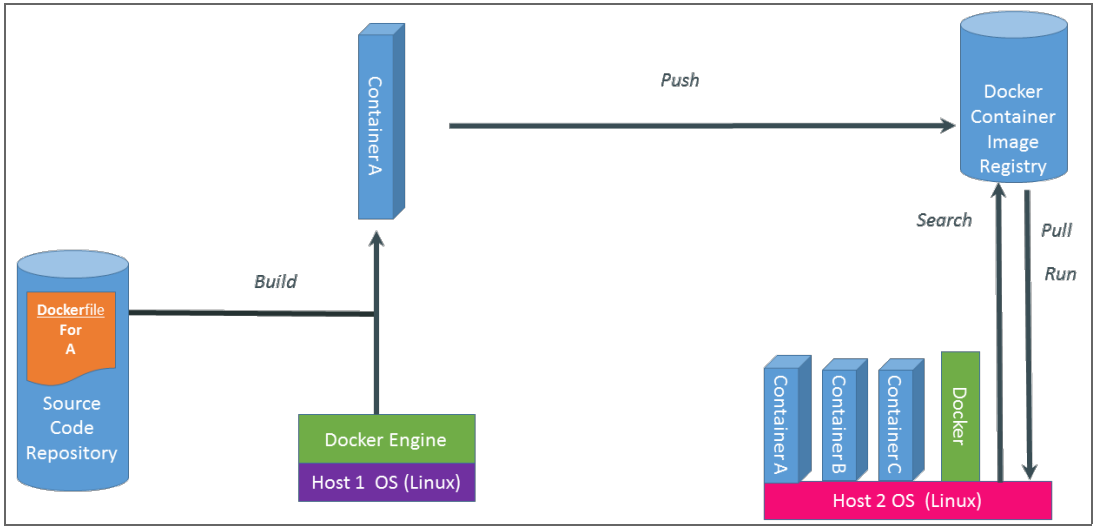
USING DOCKER IN PRACTICE



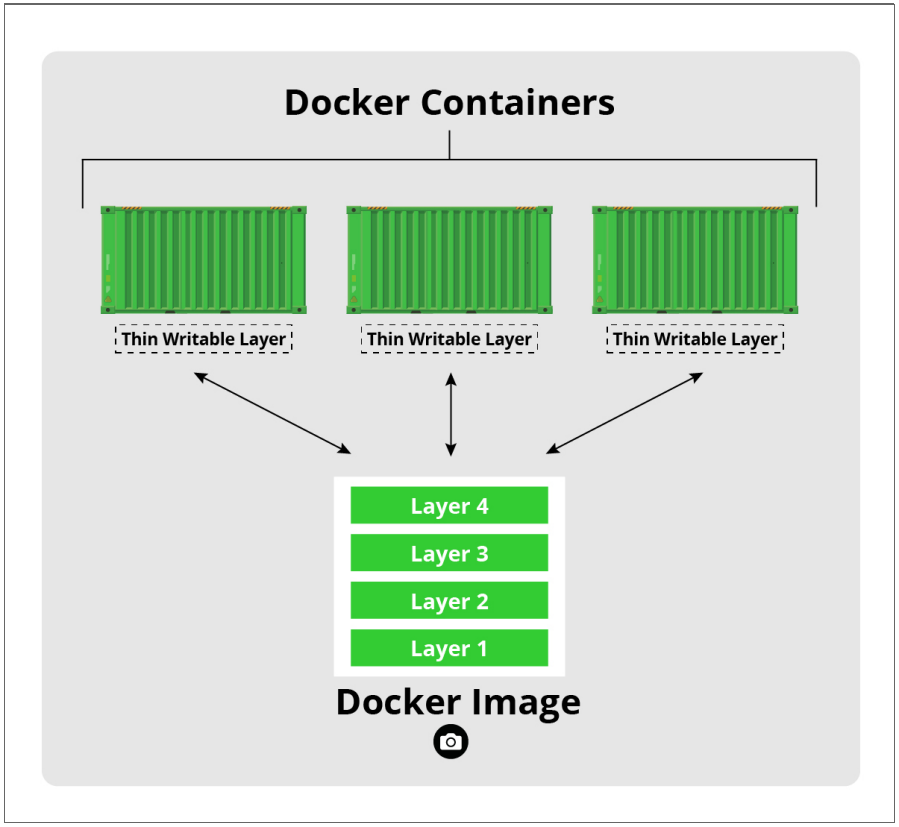
VOCABULARY OF DOCKER

- **Layer**: Set of read-only files to provision the system
- **Image**: Read-Only layer "snapshot" (or blueprint) of an environment.
- **Images**: can inherit from other **Images**. Images must have a *name* and a *tag*
- **Container**: Read-Write instance of an **Image**
- **DockerFile**: Description of the process used to build an Image
- **Container Registry**: Repository of Docker Images
- **Dockerhub**: The main container registry of docker.com

WORKFLOW


Container A
Push
Docker Container Image Registry
Search
Pull
Run
Build
Dockerfile For A
Source Code Repository
Docker Engine
Host 1 OS (Linux)
Container A
Container B
Container C
Docker
Host 2 OS (Linux)

LAYERS, CONTAINER, IMAGE



IMAGE

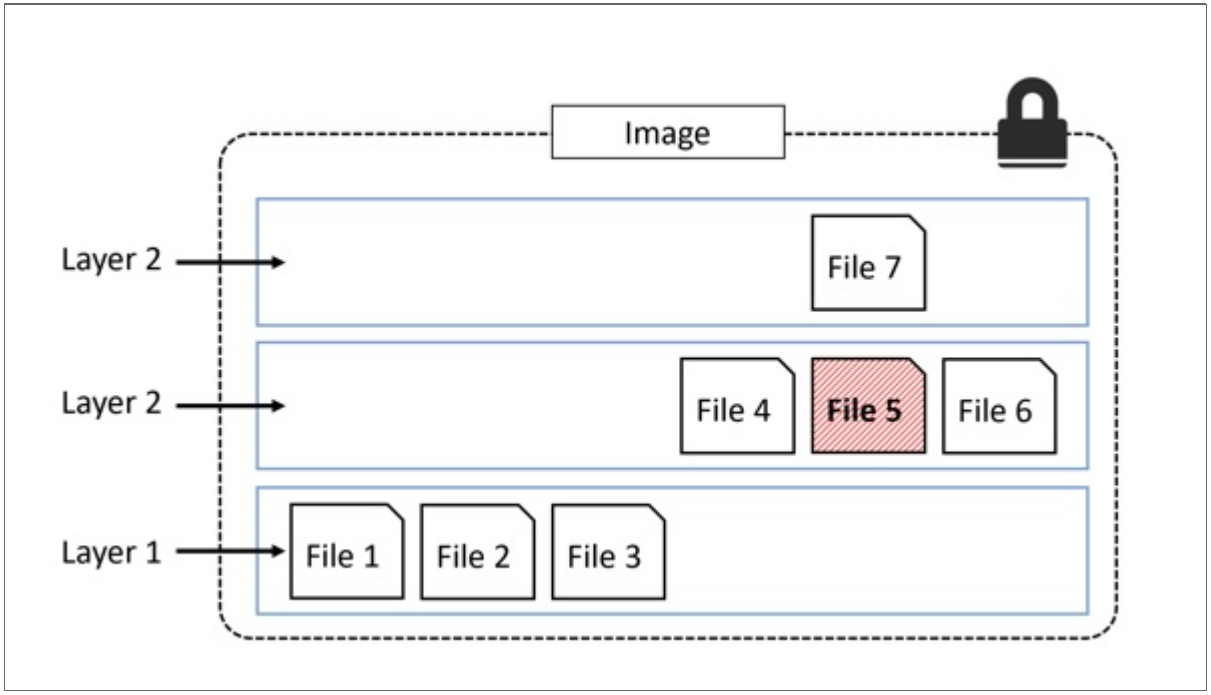


Image
Layer 2
File 7
Layer 2
File 4
File 5
File 6
Layer 1
File 1
File 2
File 3

LAYER / IMAGE ANALOGY

Docker:

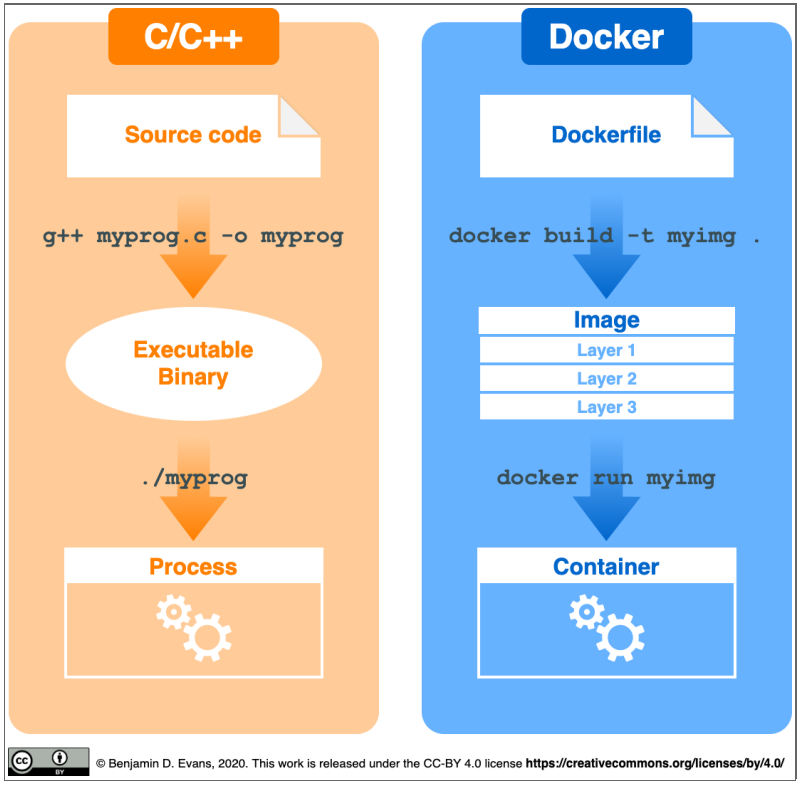
```
FROM python:3.11
RUN pip install torch
CMD ipython
```

```
docker build -f Dockerfile -t my-image:1.0 .
docker run my-image
```

Python:

```
class BaseImage:
    def __init__(self, a):
        self.a = a

class NewImage(BaseImage):
    def __init__(self, a, b):
        super(NewImage, self).__init__(a=a)
        self.b = b

container = NewImage(a=0,b=1)
```

DOCKERFILE / LAYER / IMAGE / CONTAINER ANALOGY



DOCKERFILE

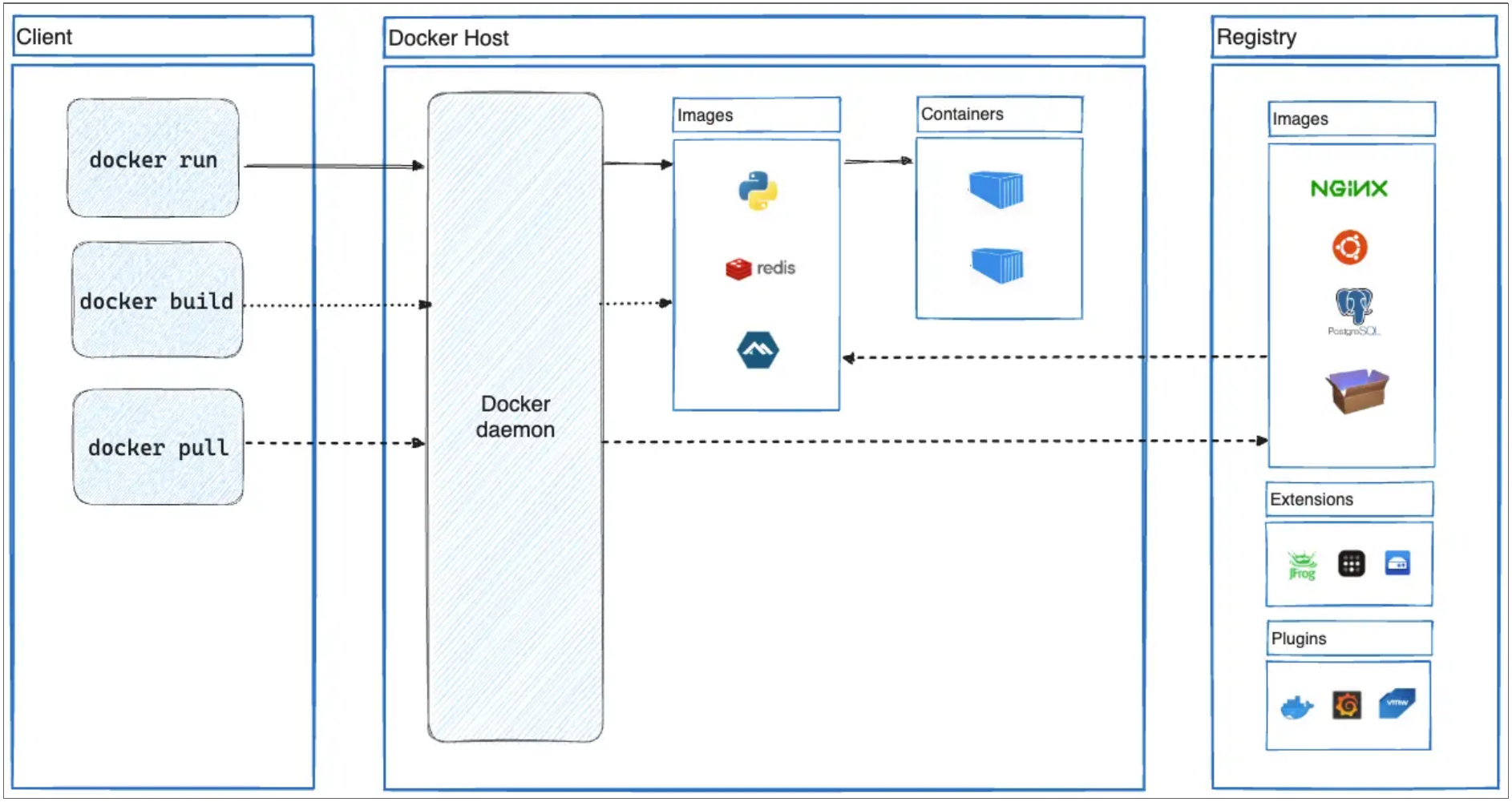
- Used to build Images

```
FROM python:3.11
ENV MYVAR="HELLO"
RUN pip install torch
COPY my-conf.txt /app/my-conf.txt
ADD my-file.txt /app/my-file.txt
EXPOSE 9000
WORKDIR "/WORKDIR"
USER MYUSER
ENTRYPOINT ["/BIN/BASH"]
CMD ["ECHO" , "${MYVAR}"]
```

```
docker build -f Dockerfile -t my-image:1.0 .
docker run my-image
```

- Reproducible (if you include static data)
- Can be put under version control (simple text file)

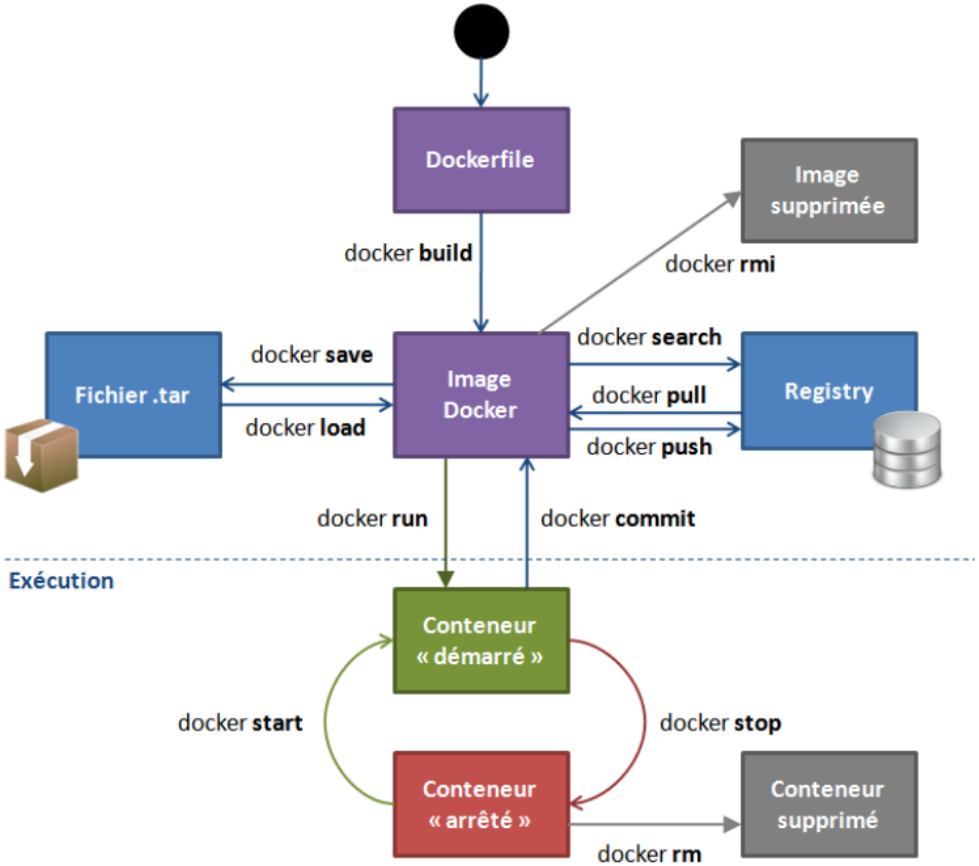
ARCHITECTURE



REGISTRY

- Local registry: All images/containers in your machine
- **https://hub.docker.com/**
- GCP Container Registry
- Social Dimension (share docker images to speed up development/deployment)

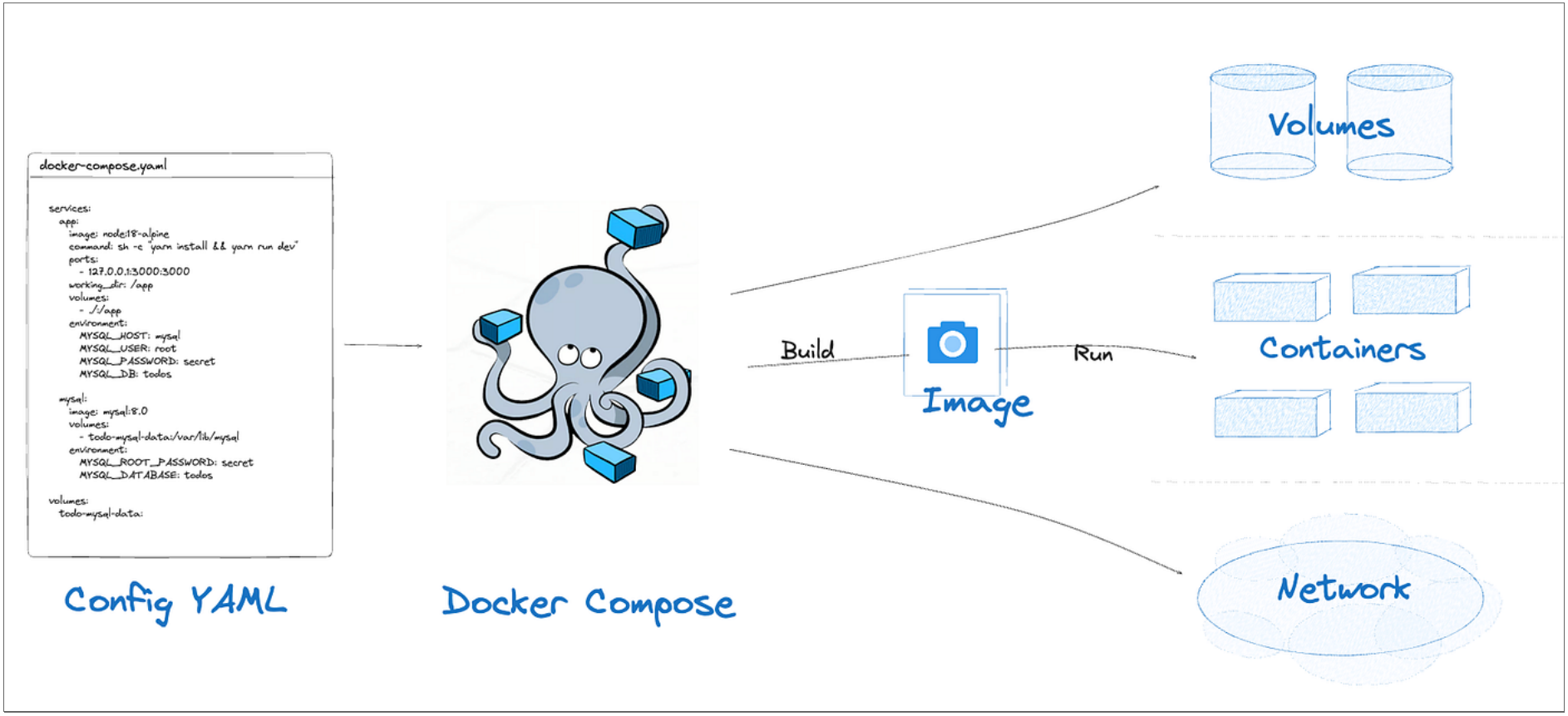
IN PRACTICE


Dockerfile
docker build
Image supprimée
docker rmi
Fichier .tar
docker save
docker load
Image Docker
docker search
docker pull
docker push
Registry
docker run
docker commit
Exécution
Conteneur « démarré »
docker start
docker stop
Conteneur « arrêté »
Conteneur supprimé
docker rm

WHAT ABOUT MULTI-APPLICATIONS CONTAINERS ?

DOCKER COMPOSE

- Multi-containers application with networking (communication)
- "Glue" for complex applications and microservices



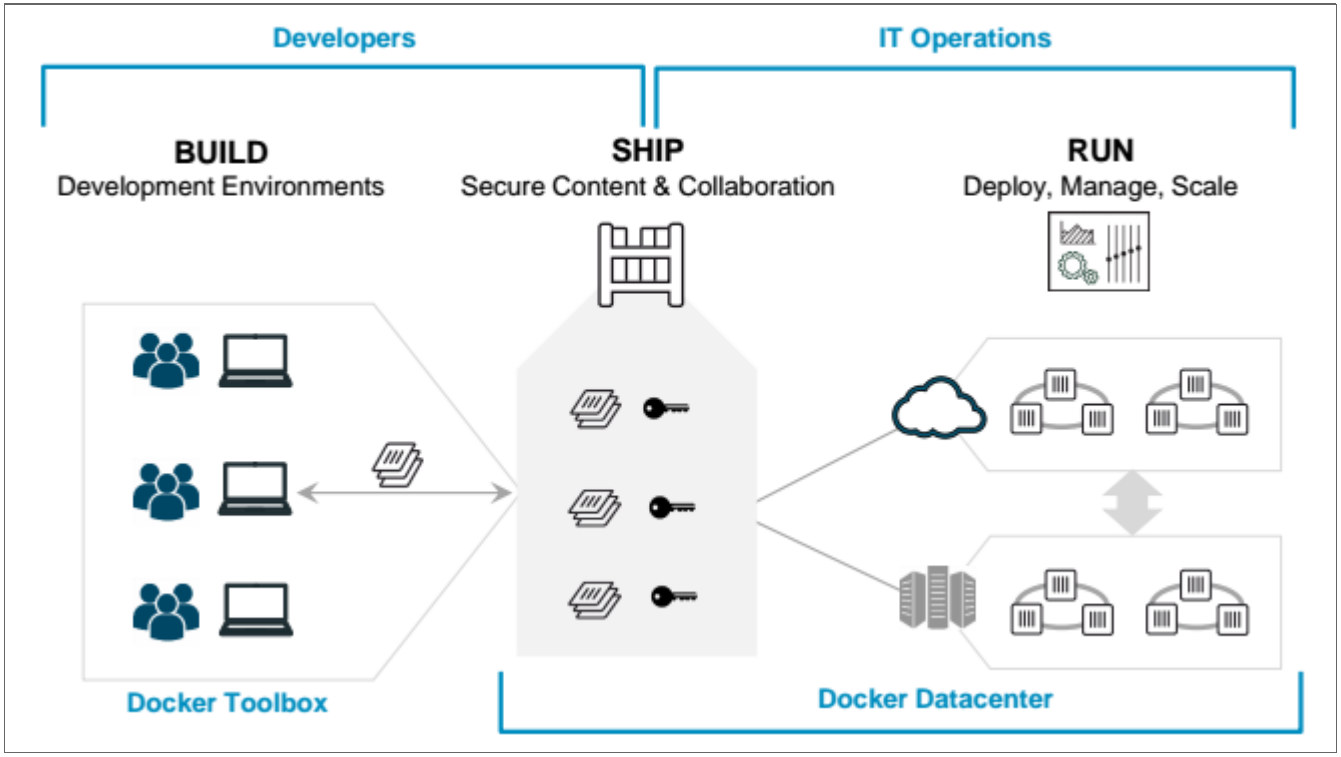
DOCKER COMPOSE (EXAMPLE)

A database and a webapp

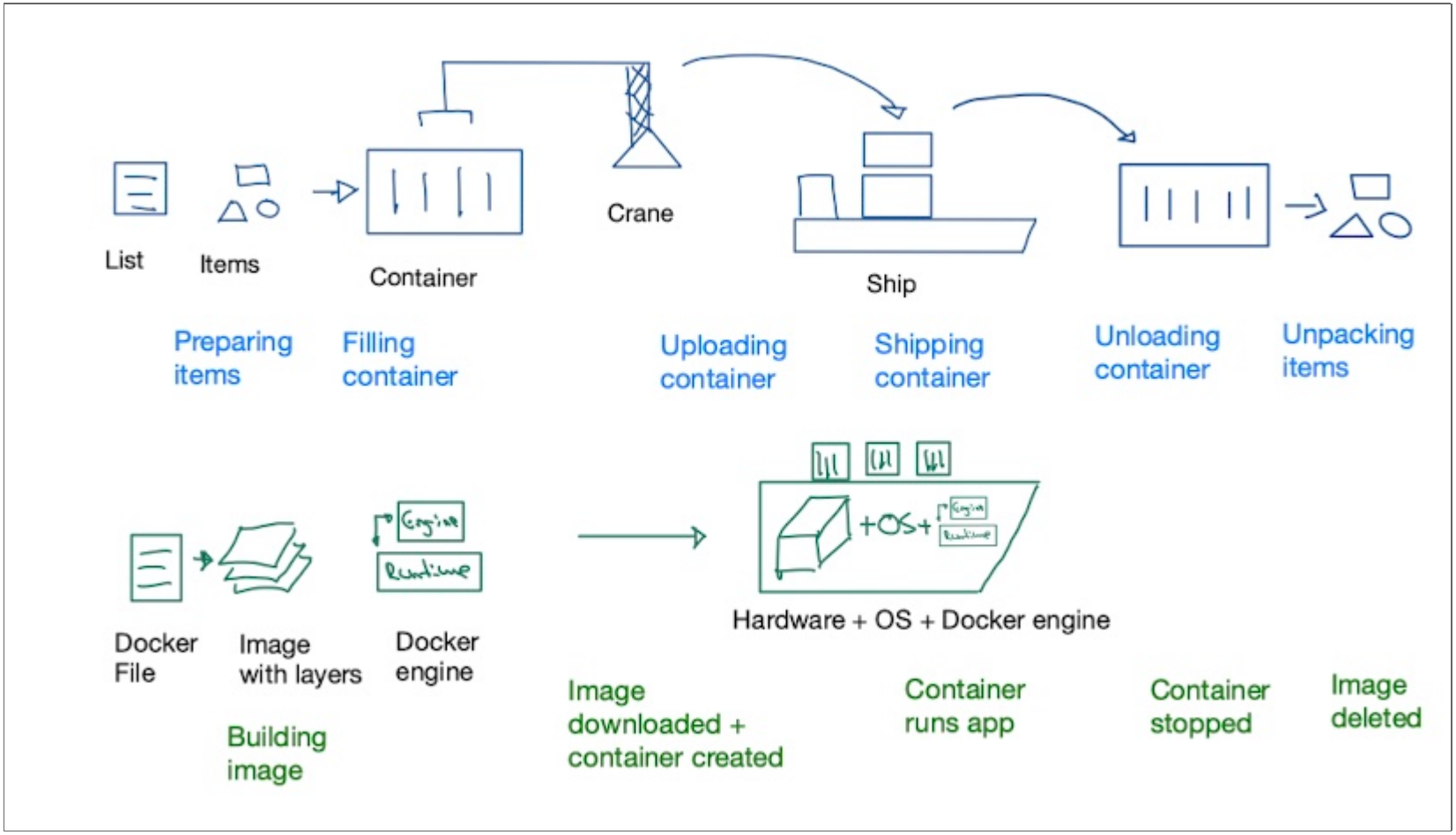
```yaml
services:
  app:
    build: .
    image: takacsmark/flask-redis:1.0
    environment:
      - FLASK_ENV=development
    ports:
      - 5000:5000

  redis:
    image: redis:7-alpine
```

`docker compose up` starts both containers (you will see that next week)

REMEMBER THIS !



AN ANALOGY...



https://bernhardwenzel.com/2022/the-mental-model-of-docker-container-shipping/

DOCKER AND GCP

GCP & DOCKER

- The per-project dockerhub is called **Container Registry**
- Your images look like this `eu.gcr.io/project-id/a/b/c:1.0`
- You can use **Google Cloud Build** to build dockerfiles remotely
- `gcloud builds submit --tag gcr.io/[PROJECT_ID]/quickstart-image .`
- To use gcloud with docker: `gcloud auth configure-docker`
- You can even deploy **"virtual machines" with containers directly**

DEMO TIME

Speaker notes