Docker for ML Practitioners

Stevan Radanović
stevan.radanovic@gmail.com

Docker for ML Practitioners

- Goal:
  - Docker 101
  - Making our first ML / DS environment
- Format

Installing Docker

- Docker CE
  - Linux (Ubuntu): http://tiny.cc/8za66y
- Docker Desktop for Windows *
  - Windows: http://tiny.cc/nib66y
- Docker Desktop for Mac *
  - Mac: http://tiny.cc/web66y or `brew cask install docker`
- DockerID: https://hub.docker.com/
- GitHub repo: http://tiny.cc/zwr76y

Why Docker?

- DataOps
- Docker = containerization
- **But we already have VMs, why containers?**

Why Docker?

- DataOps
- Docker = containerization
- But we already have VMs, why containers?
- Both **VMs** and **containers**:
  - isolate an application and its dependencies into a self-contained unit that can run anywhere
  - more efficient use of computing resources

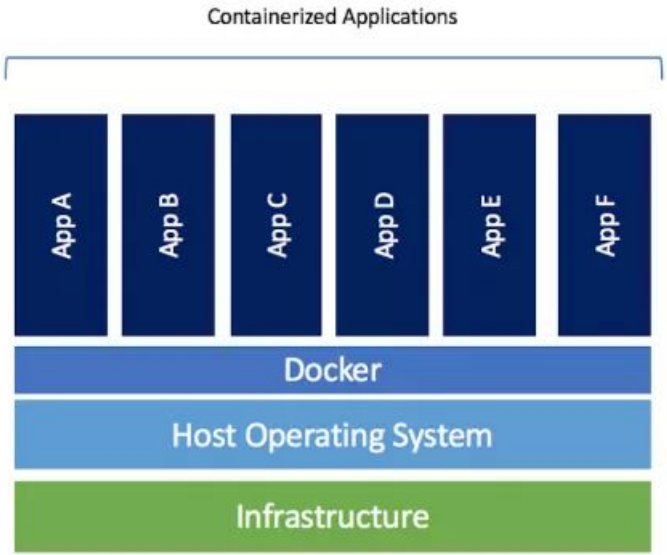
Why Docker?

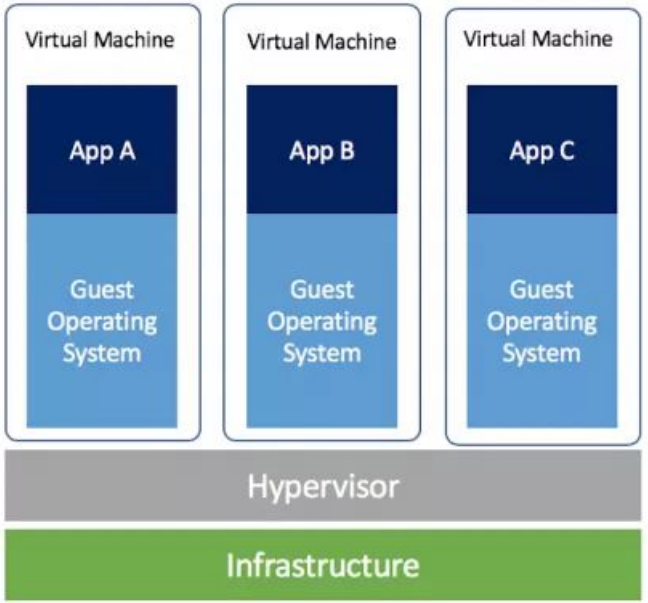
- ...and both **VMs** and **containers**:
  - can execute commands as root
  - can mount file systems
  - have private space for processing
  - have a private network interface
  - etc.

Why Docker?

- ...but the one big difference between **VMs** and **containers**:
  - containers share the host system's kernel,
  - while each VM has separate guest OS

Why Docker?





source: https://blog.docker.com/

Why Docker?

- ...which means **containers**:
  - require fewer computing resources
  - can dynamically allocate resources
  - are faster to spin up
  - are more portable

Why Docker in ML?

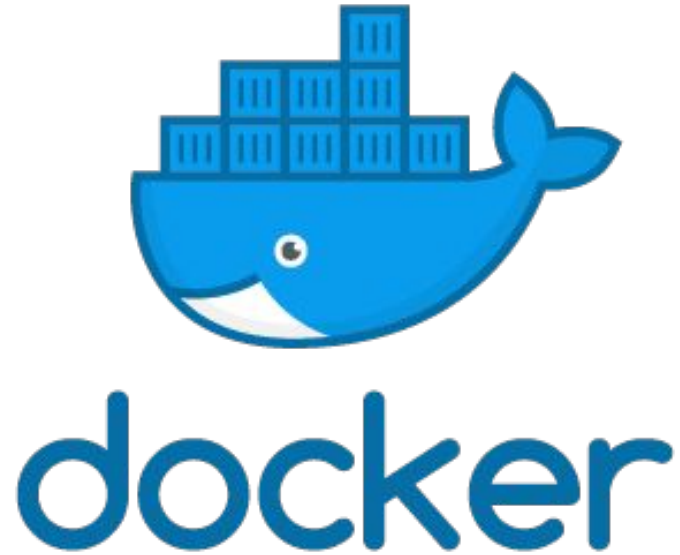
- We have Pipenv, why do we need Docker?

Why Docker in ML?

- We have Pipenv, why do we need Docker?
- Even if all our projects were written in Python...
- ...Docker is how they run in production

Docker concepts

- Container is a good analogy:
  - portable
  - contains programs and libs
  - clear interface

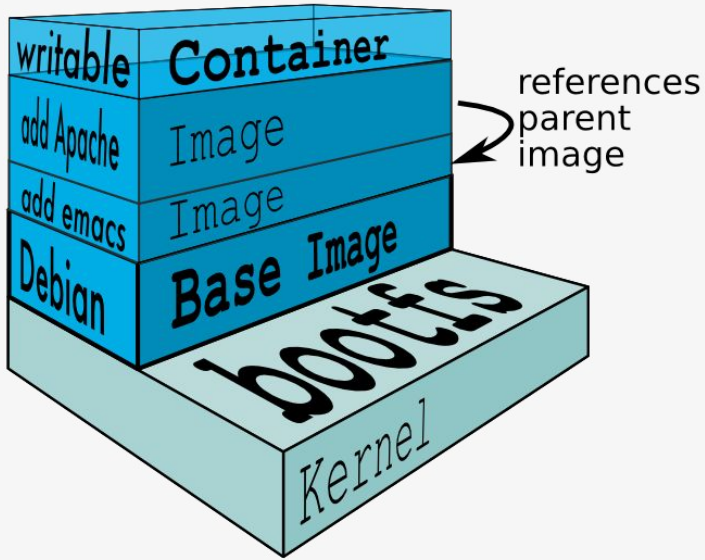


source: https://www.docker.com/

Docker concepts

- **Dockerfile** - recipe for Docker image
- **Docker image** - template for Docker container
- **Docker container** - running instance of Docker image
- **Docker Hub** - default Docker registry where users share Docker images (https://hub.docker.com)
- You can have your own Docker registry

Docker concepts



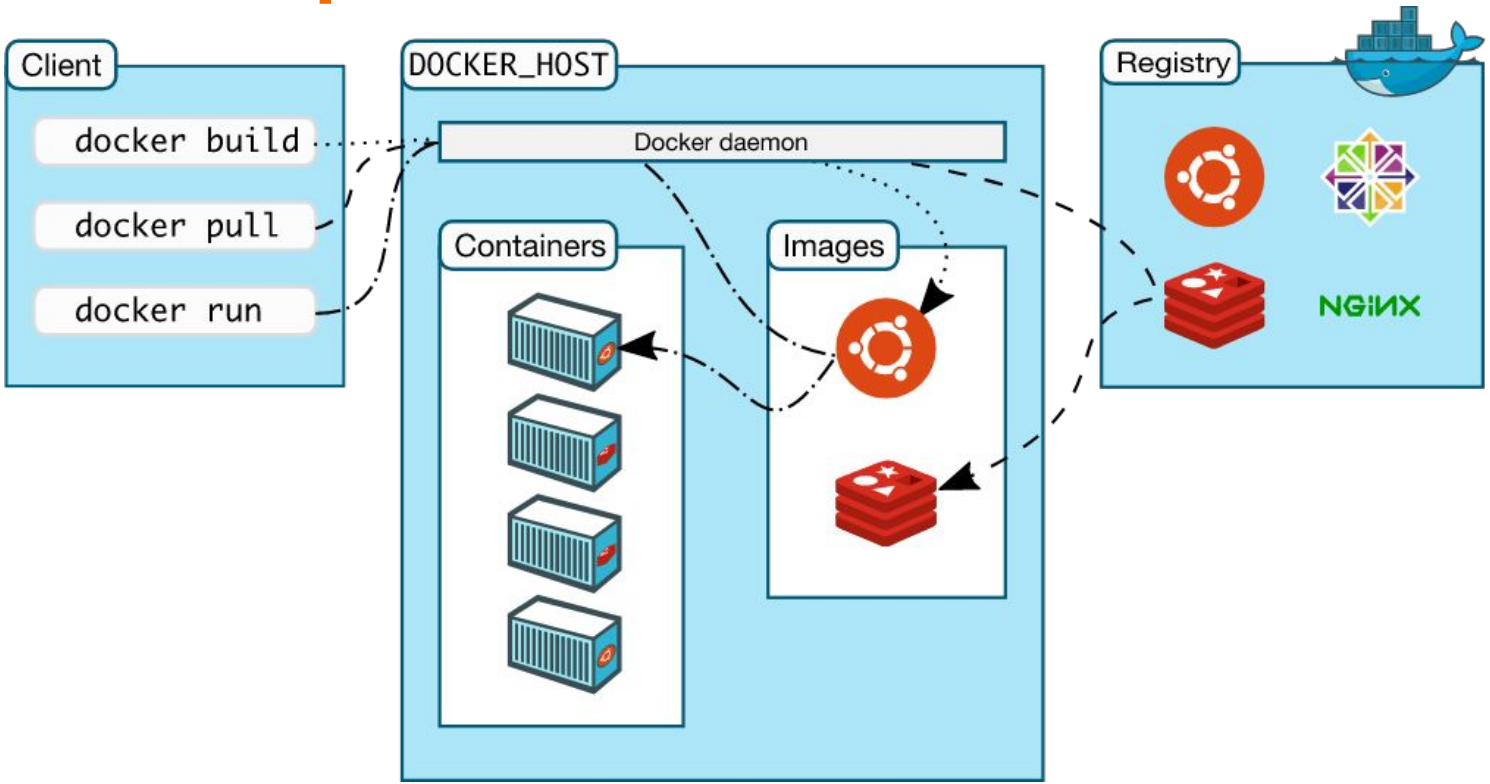
source: https://www.docker.com/

source: https://www.delish.com/

Docker concepts

- **Docker platform** - package and run applications
- **Docker Engine** - client-server application that uses daemon process to receive commands and execute them
  - Docker CE - open-source
  - Docker Enterprise - proprietary
- **Docker client** - issue commands to Docker using CLI
- **Docker daemon** - server listening to Docker API calls
- **Docker volumes** - persistent data

Docker concepts



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

Using Docker

- Check version: `docker --version`
- More detailed info about installation: `docker info`
- Hello world: `docker run hello-world`
- List all (local) images: `docker image ls`
- List all containers: `docker container ls`
- List all running containers: `docker container ls --all`

Dockerfile

- Basic Dockerfile: `FROM ubuntu:18.04`

Dockerfile

- FROM - specifies the base (parent) image
- LABEL - provides metadata (e.g. maintainer info)
- ENV - sets a persistent environment variable
- RUN - runs a command and creates an image layer, used to install packages into containers
- COPY - copies files and directories to the container
- ADD - copies and unpacks files and directories to the container

Dockerfile

- CMD - provides a command and arguments for an executing container, only one CMD per Dockerfile
- WORKDIR - sets the workdir for the following instructions
- ARG - defines a variable to pass to Docker at build-time
- ENTRYPOINT - provides command and arguments for an executing container
- EXPOSE - exposes a port
- VOLUME - creates a directory mount point to access and store persistent data

Docker container

- create - create a container from an image
- start - start an existing container
- run - create a new container and start it
- ls - list running containers
- inspect - info about a container
- logs - print logs
- stop - gracefully stop running container
- kill - stop main process in container abruptly
- rm - delete a stopped container

Docker image

- build - build an image
- push - push an image to a remote registry
- ls - list images
- history - see intermediate image info
- inspect - see info about an image, including the layers
- rm - delete an image

Resources

- https://jupyter-docker-stacks.readthedocs.io/en/latest/