[DOCKER]

Container

software unit consisting of all essential nodes & dependencies for app\service to run
* a process that runs a bunch of processes
* isolates software from host environment
* ensures containerized software always run the same regardless of:
  - ↳ infrastructure
  - ↳ differences in development & staging

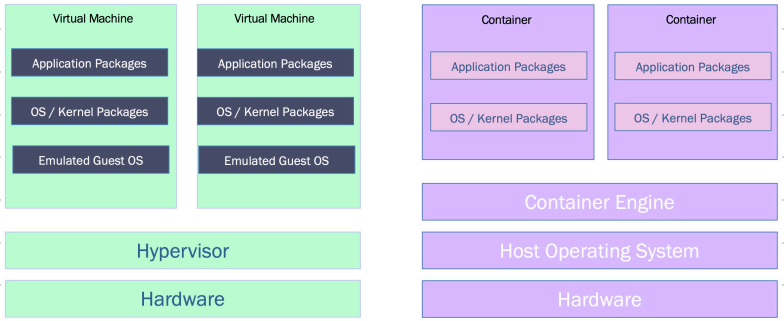
Container Image

blueprint that provides instructions to build docker container
* has multiple "layers"
  - ↳ each layer has instructions\cmds that mod. container file sys.
  - ↳ layers result from docker cmds that caused a change in previous img layer
* Docker imgs become containers when they're run on Docker Engine

VS Virtual Machine

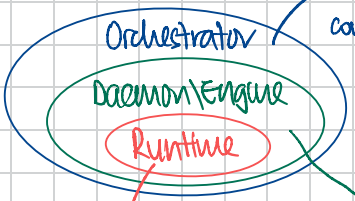
CONTAINER
* lightweight
* fast
* sharing os\kernel resources
* orchestrated by container engine

VM
* bulky
* slow
* has own software stack that emulates os\kernel resources (ex., CPU, disk, networking devices)
* better isolation for host



Container Registry

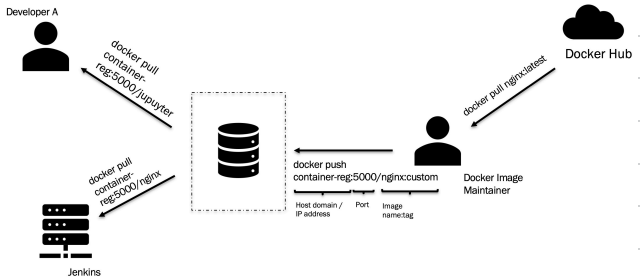
stateless, highly scalable repo — used to store & access container imgs
* intermediary for sharing container imgs btwn systems
* impt in [CI\CD] pipeline
  - ↳ continuous integration + continuous delivery
  - ↳ support container-based app dev, often as part of DevOps processes
  - ↳ save time in creation & delivery of cloud-native apps
* (Docker) default storage driver: local posix sys.
  - ↳ also supports other storage drivers (ex., AWS S3, MS Azure, OpenStack Swift)
* Docker Hub
  - ↳ publicly accessible storage; allows users to pull\push img



Orchestrator
- automates deployment, mgmt, scaling & networking of containers (ex., Docker Swarm)
  - * in charge of multiple Docker nodes
  - * has tools to manage, scale & maintain containerized apps

Runtime
- foundational software allowing containers to operate w\in a host sys.
  - * containerd: high-level runtime that runs multiple runc instances
  - * runc: low-level runtime that runs w\in a single container

Daemon\Engine
- software that accepts user requests (ex., cmd line options, pull imgs) & runs container (end-user perspective)
  - * Docker daemon: creates & manages Docker objs. (ex., imgs, containers, networks, volumes)
    - ↳ CLI uses REST API to ctrl\interact w\in daemon thru scripting or direct CLI cmds
    - ↳ daemon can be local or remote; can also communicate w\in other daemons to manage Docker services



Container Layer
Image Layer 2
Image Layer 1
Base Image Layer

3 read-write layer
3 read-only layer

Container Runtime Layer
Image Layer 1
Base Image Layer

Dockerfile:

```
FROM alpine:3.14          # base layer formed based on alpine:3.14 image
RUN apk add --no-cache nmap   # new layer formed on top of base layer w\in "nmap" pkg added to file sys.
WORKDIR /tmp              # WORKDIR cmd ∅ alter file sys. ⇒ ∅ new img layer added; only runtime layer on top
```