DOCKER

- an open-sourced platform for running, shipping and deploying an application.
- It separates the application from the infrastructure!
- no more "works on my machine"
- Docker uses a client/server architecture. The server uses the Docker engine to cache images, run and manage containers, handle logs and much more.

Containers >> VM

Docker packages the applications with all necessary dependencies, libraries, and system tools together into the **container**.
A container is a standardized unit that has everything required to run an application.
an isolated environment,is portable, easily shared and distributed.

Containers are executable units of software in which application code is packaged, along with its libraries and dependencies, in common ways so that it can be run anywhere, whether it be on desktop, traditional IT, or the cloud.

To do this, containers take advantage of a form of operating system (OS) virtualization in which features of the OS (in the case of the Linux kernel, namely the namespaces and cgroups primitives) are leveraged to both isolate processes and control the amount of CPU, memory, and disk that those processes have access to.

Containers are small, fast, and portable because unlike a virtual machine, containers do not need include a guest OS in every instance and can, instead, simply leverage the features and resources of the host OS.

How to reach the container build ?

Docker images

It is an executable application artifact/blueprint to the docker container.
It includes applications source code, but also the complete environment configuration.

Dokerfiles

A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image.

Docker registry

A storage and distribution system for docker images. helps in storing and sharirng !
eg : docker hub , aws !

How docker works

- A Dockerfile is the source code.
- An image is the binary resulting from compiling and building your Dockerfile
- A container is the running instance of an image, a process by all means.
- Images are cached and containers are run by a Docker host (a machine running the Docker engine).

Images are stored in a registry, think of it as a repository for package managers such as apt, yum, pip or npm. Docker offers a public registry that anyone can use to store images, as long as these are kept public

open diff folder to see the diff btw cmd , entrypoint and run !

layering and cache-ing

images are built as layers on the base image where each layer is an instruction from the dockerfile!

```
FROM ubuntu:18.04
RUN apt-get update
RUN apt-get install nginx -y

COPY index1.html /usr/share/nginx/html/index1.html
COPY index2.html /usr/share/nginx/html/index2.html
COPY index3.html /usr/share/nginx/html/index3.html

RUN rm /etc/nginx/nginx.conf
COPY default.conf /etc/nginx/nginx.conf

CMD ["nginx", "-g", "daemon off;"]
```

docker build -t name:tag location

images are pile of layers.
each exec statement adds a layer

run gets executed during image build
hence :8/9
cmd during container run

total Building time : 1481.1s

```
=> => sha256:72d9f18d70f395ff9bfae4d193077ccea3ca583e3da3dd66f5c84520c0100727 25.69MB / 25.69MB      13.5s
=> => extracting sha256:72d9f18d70f395ff9bfae4d193077ccea3ca583e3da3dd66f5c84520c0100727            4.3s
=> [internal] load build context                                                                   0.1s
=> => transferring context: 1.45kB                                                                 0.0s
=> [2/8] RUN apt-get update                                                                      371.3s
=> [3/8] RUN apt-get install nginx -y                                                           1081.1s
=> [4/8] COPY index1.html /usr/share/nginx/html/index1.html                                        0.2s
=> [5/8] COPY index2.html /usr/share/nginx/html/index2.html                                        0.1s
=> [6/8] COPY index3.html /usr/share/nginx/html/index3.html                                        0.1s
=> [7/8] RUN rm /etc/nginx/nginx.conf                                                              0.6s
=> [8/8] COPY default.conf /etc/nginx/nginx.conf                                                   0.1s
=> exporting to image                                                                              1.3s
=> => exporting layers                                                                             1.3s
```

cmd gets executed during container run :

```
\Users\Admin>docker ps
NTAINER ID   IMAGE         COMMAND                  CREATED          STATUS          PORTS                    NAMES
4086e7d934   demo:latst    "nginx -g 'daemon of…"   21 seconds ago   Up 19 seconds   0.0.0.0:8000->90/tcp     jovial_snyder
```

```
=> [internal] load metadata for docker.io/library/ubuntu:18.04                                                    1.4s
=> [1/4] FROM docker.io/library/ubuntu:18.04@sha256:a3765b4d74747b5e9bdd03205b3fbc4fa19a02781c185f97f24c8f4f84ed7bbf  0.0s
=> [internal] load build context                                                                                  0.1s
=> => transferring context: 124B                                                                                  0.0s
=> CACHED [2/4] RUN apt-get update  && apt-get install nginx -y  && rm /etc/nginx/nginx.conf                      0.0s
=> CACHED [3/4] COPY index1.html index2.html index3.html /usr/share/nginx/html/                                   0.0s
=> CACHED [4/4] COPY default.conf /etc/nginx/nginx.conf                                                           0.0s
=> exporting to image                                                                                             0.0s
=> => exporting layers                                                                                            0.0s
=> => writing image sha256:9045d9655a94d9c0c3dc82920f657b9fe2b2e7f351a9ddc868f783211b653de4                       0.0s
=> => naming to docker.io/library/demo:latest                                                                     0.0s
```

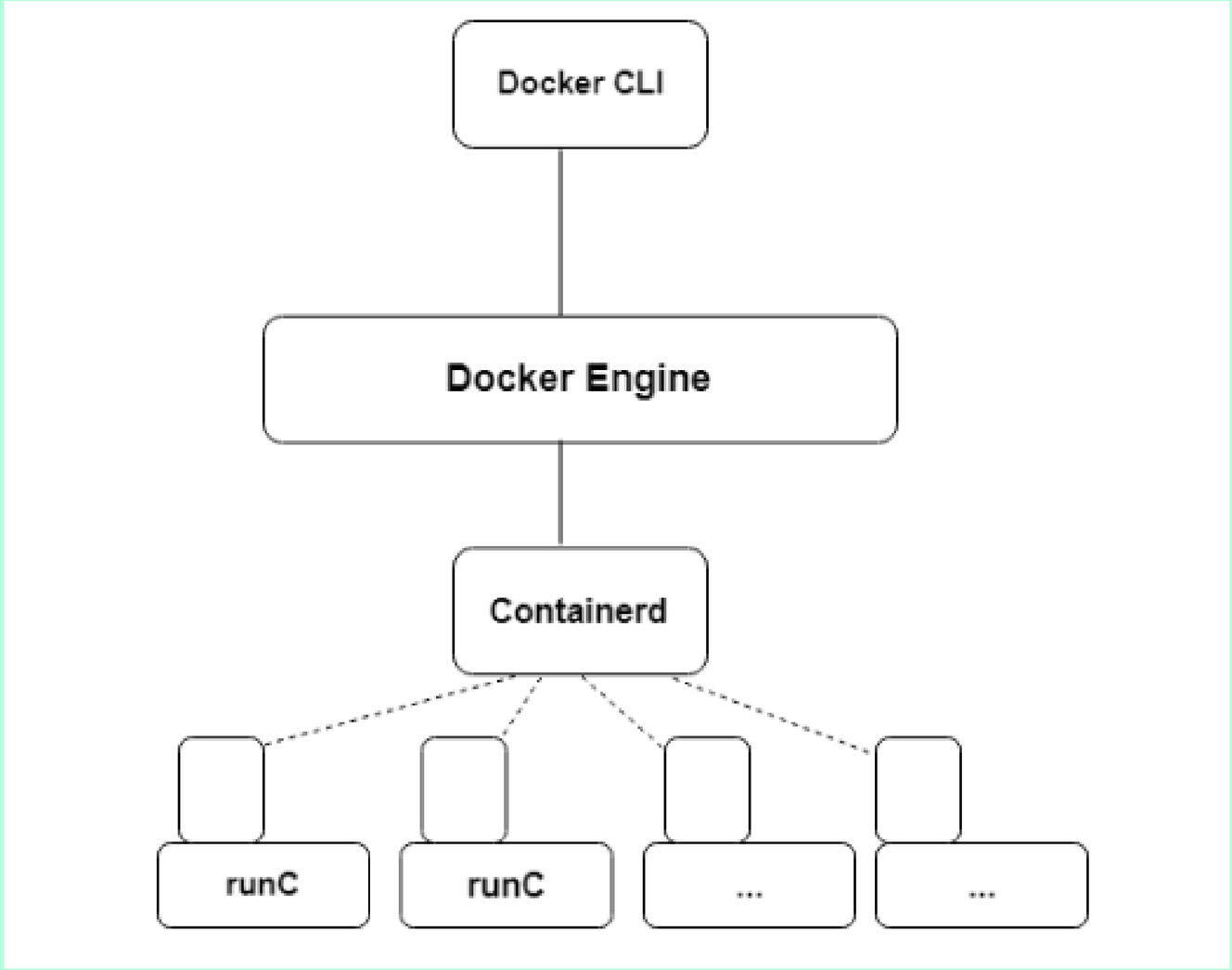
https://medium.com/@hashedin/a-guide-to-optimize-build-time-in-docker-e2c9abe074e4

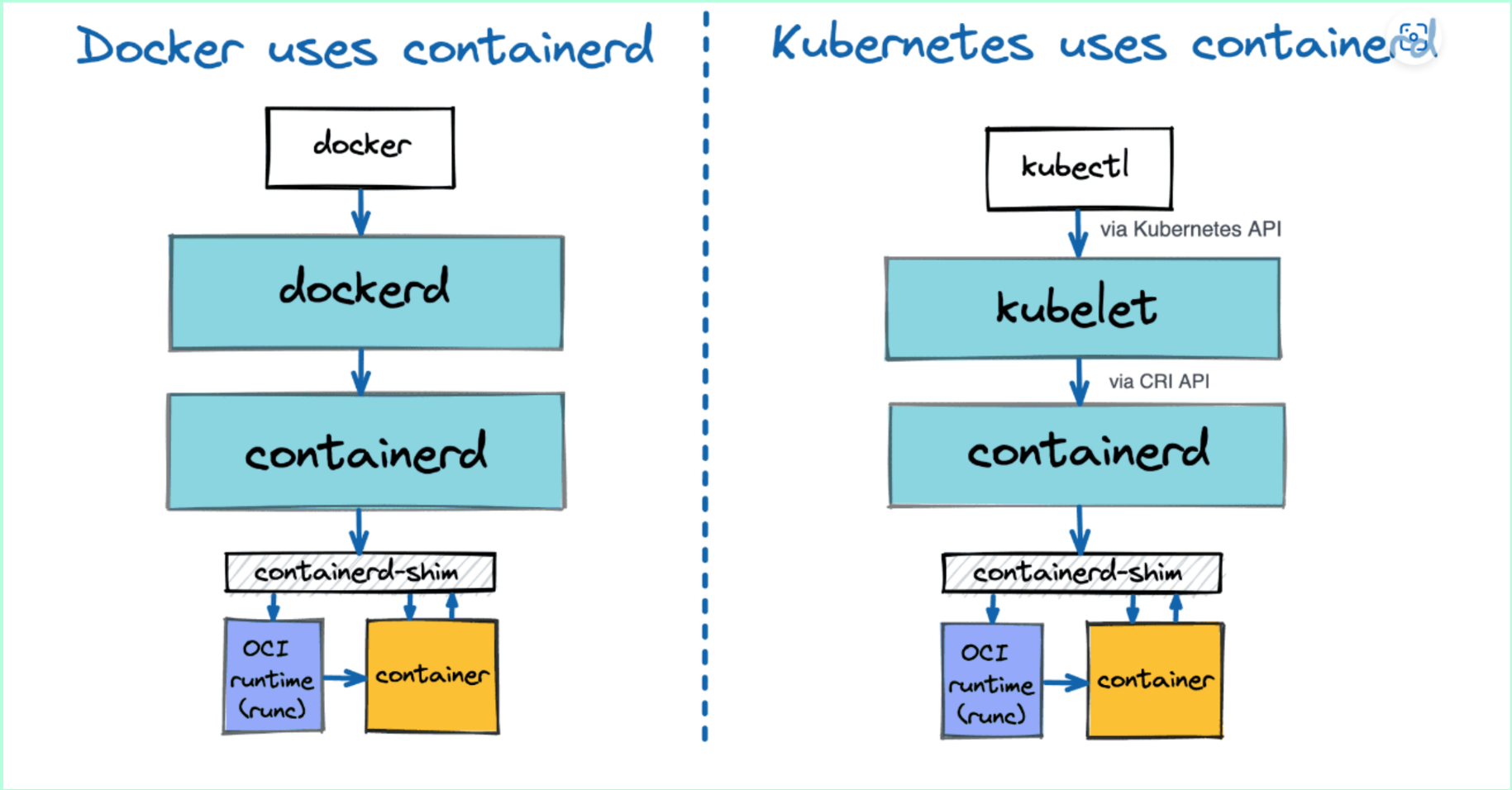
--no-cache : to rebuild from the start!

FROM is not affected ------- use -pull to get the latest tag!

containerd

containerd is available as a daemon for Linux and Windows. It manages the complete container lifecycle of its host system, from image transfer and storage to container execution and supervision to low-level storage to network attachments and beyond.


Docker CLI
Docker Engine
Containerd
runC
runC
...
...



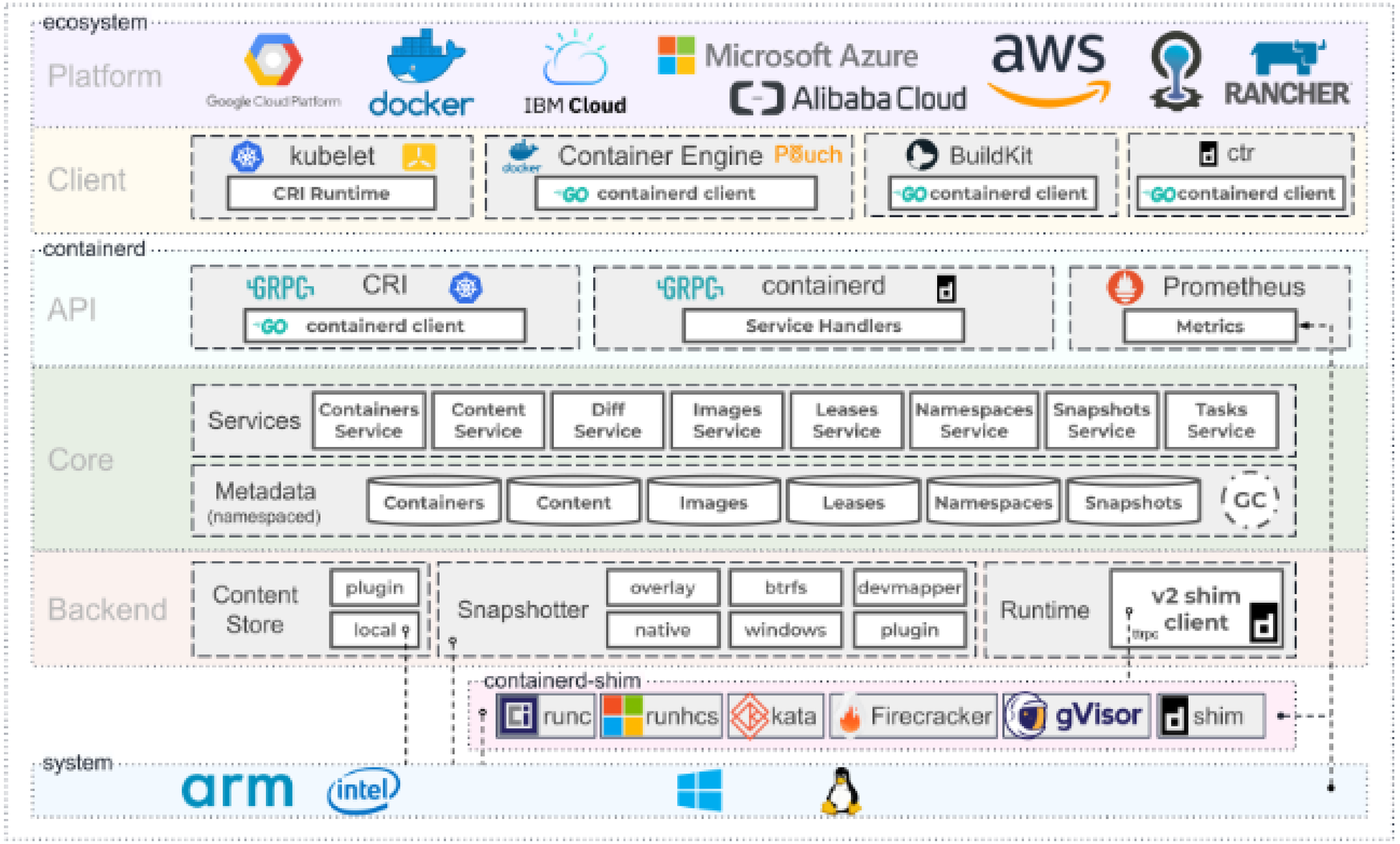
containerd with nerdctl , ctr can be used to accesss it directly

Docker is a broad set of technologies that are used to work with containers. containerd is an example of a container runtime. A container runtime is that process that does the actual work of creating, running, and destroying containers. Docker uses containerd as its runtime

The host operating system has no concept of a container. However, it does provide the features such as namespaces, cgroups and file system overlays that make a container possible. containerd along with another component called a low-level runtime, in this case, runc, does the work of interacting with the host operating system's kernel to do the low-level workdsthat goes with creating a container.

Today containerd is part of the Cloud Native Computing Foundation (CNCF), an organization that supports Kubernetes and Docker based deployments. Docker is still an independent project.

containerd vs CRI vs OCI vs CRIO vs RUNC
https://www.docker.com/blog/what-is-containerd-runtime/


ecosystem
Platform
Google Cloud Platform
docker
IBM Cloud
Microsoft Azure
Alibaba Cloud
aws
RANCHER
Client
kubelet
CRI Runtime
Container Engine
containerd client
BuildKit
containerd client
ctr
containerd client
containerd
API
GRPC
CRI
containerd client
GRPC
containerd
Service Handlers
Prometheus
Metrics
Core
Services
Containers Service
Content Service
Diff Service
Images Service
Leases Service
Namespaces Service
Snapshots Service
Tasks Service
Metadata
(namespaced)
Containers
Content
Images
Leases
Namespaces
Snapshots
GC
Backend
Content Store
plugin
local
Snapshotter
overlay
btrfs
devmapper
native
windows
plugin
Runtime
v2 shim
client
ttrpc
containerd-shim
runc
runhcs
kata
Firecracker
gVisor
shim
system
arm
intel

PROBLEMS WITH DOCKER

- **cannot run applications as fast as a bare-metal server.**
- **does not provide cross-platform compatibility.**
- **Hard to run applications with graphical interfaces**
- **Does not solve all your security problems.**
- **speed !**

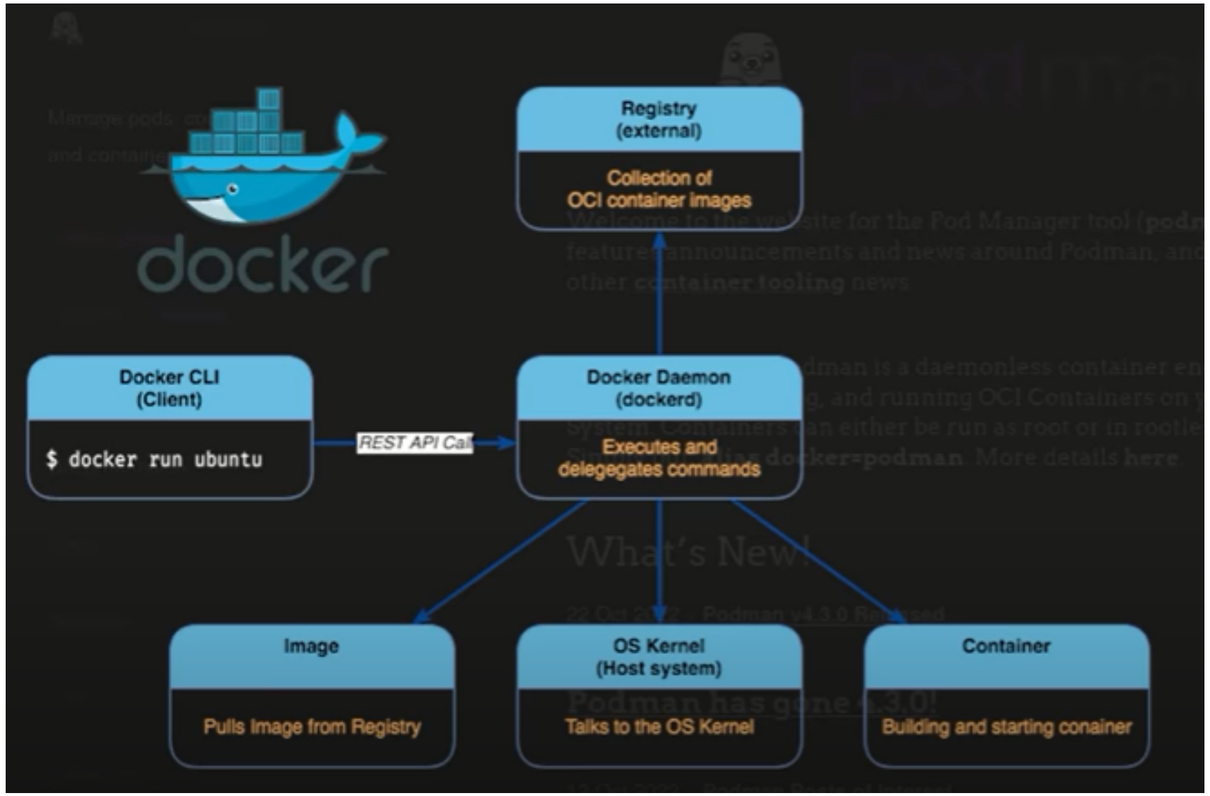
https://www.channelfutures.com/open-source/when-not-to-use-docker-understanding-the-limitations-of-containers

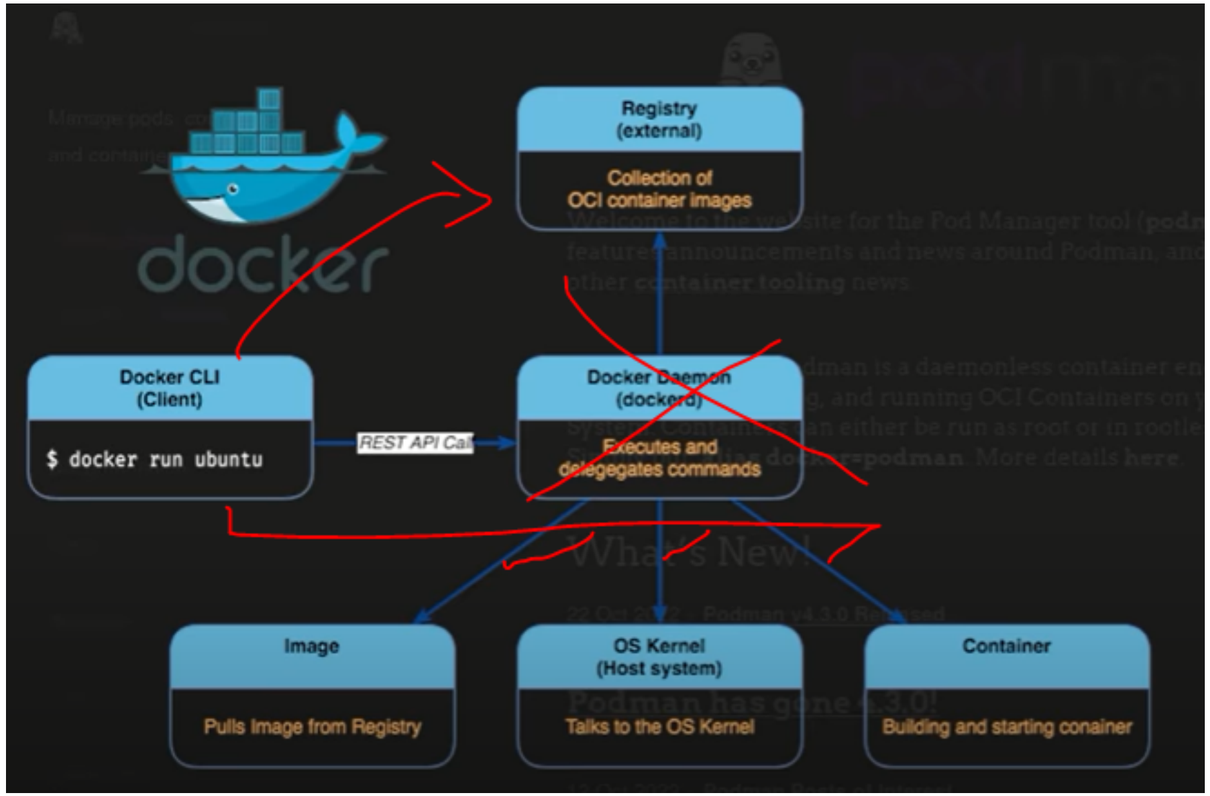
https://www.freecodecamp.org/news/7-cases-when-not-to-use-docker/

https://www.tigera.io/learn/guides/container-security-best-practices/docker-security/#:~:text=Unrestricted%20Access-,What%20is%20the%20risk%3F,root%20access%20to%20the%20host.

https://snyk.io/learn/docker-security/top-5-vulnerabilities/#:~:text=A%20Docker%20vulnerability%20is%20any,and%20Exposures%20(CVE)%20list.

PODMAN





- PODMAN IS A daemon-less open sourrce LINUX tool
- Containers under the control of Podman can either be run by root or by a non-privileged user.
- Podman manages the entire container ecosystem which includes pods, containers, container images, and container volumes using the libpod library.

HOW podman is more secure?

https://cloudnweb.dev/2019/10/heres-why-podman-is-more-secured-than-docker-devsecops/

https://www.solutelabs.com/blog/podman-vs-docker#:~:text=Docker%20daemon%20has%20a%20security,%2C%20by%20default%2C%20improves%20security.

https://www.redhat.com/sysadmin/basic-security-principles-containers#:~:text=Podman%20takes%20advantage%20of%20user,and%20not%20impact%20the%20host.

buildah

Buildah is an open source, Linux-based tool used to build Open Container Initiative (OCI)-compatible containers, meaning the containers are compatible with Docker and Kubernetes as well.

Buildah is a tool for building OCI-compatible images through a lower-level interface. Similar to Podman, Buildah doesn't depend on a daemon such as Docker and it doesn't require root privileges. Buildah provides a command-line tool that replicates all the commands found in a Dockerfile. This allows you to issue Buildah commands from a scripting language such as Bash.

- Builds container images with or without Dockerfiles (a text document that contains all the commands a user could call on to assemble an image)
- Creates container images from scratch or from an existing container image starting point
- Doesn’t include build tools within the image itself, reducing the size of built images, increasing security, and allowing for easier transport using fewer resources
- Is compatible with Dockerfiles, allowing for easy transition from Docker
- Creates user-specific images so that images can be sorted by the user who created them.