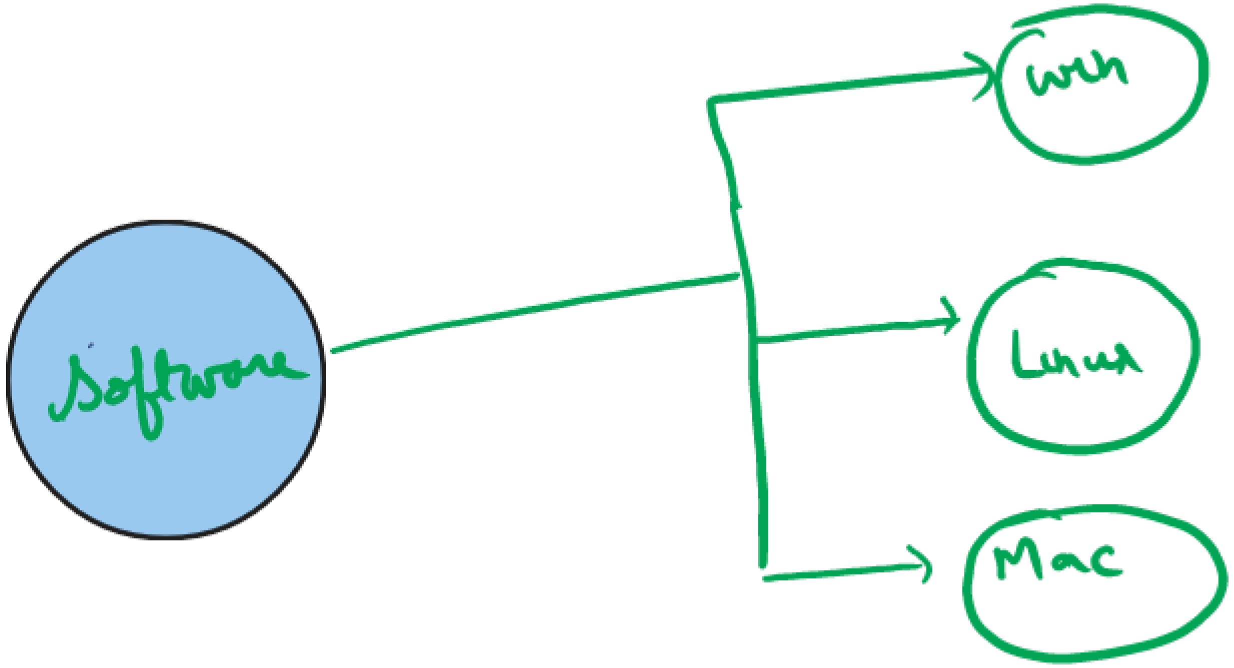
Introduction to docker

Build once use anywhere

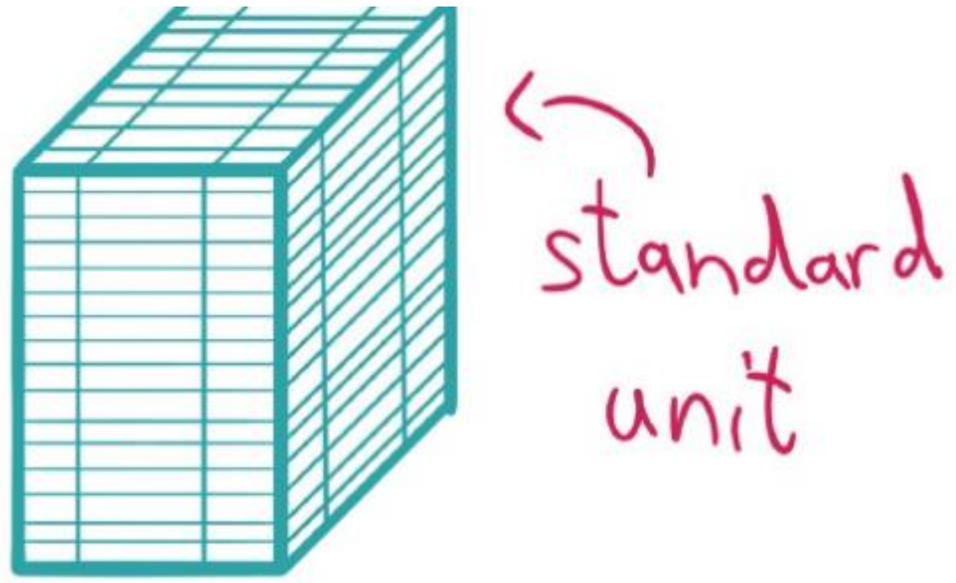
Installation of a software

- Download the installer
- Run the installer
- It may fail during the installation? Why
  - Failed due to dependency
  - Environmental issues
  - OS compatibility issue
  - Older version has issue

Current Challenges



A standardized package



History

- In 2008 a project called *Linux Containers* (LXC) started to pop-up in the wild, which should revolutionize the container world. LXC combined cgroup and namespace technologies to provide an isolated environment for running applications
- This means that Google started their own containerization project in 2007 called *Let Me Contain That For You* (LMCTFY), which works mainly at the same level as LXC does. With LMCTFY, Google tried to provide a stable and API driven configuration without users having to understand the details of cgroups and its internals.

..

- 2013 we see that there was a tool written called *Docker*, which was built on top of the already existing LXC stack. One invention of Docker was that the user is now able to package containers into images to move them between machines.
- Some years later they began to work on libcontainer, a Go native way to spawn and manage containers
- In 2015, where projects like Kubernetes hit version 1.0. A lot of stuff was ongoing during that time: The CNCF was founded as part of the Linux Foundation with the target to promote containers. The Open Container Initiative (OCI)was founded 2015 as well, as an open governance structure around the container ecosystem.

Container runtime

- Application which can run container. For example, systemd is able to run containers via systemd-nspawn, and NixOS has integrated container management as well.
- All the other existing container runtimes like CRI-O, Kata Containers, Firecracker, gVisor, containerd, LXC, runc, Nabla Containers and many more. A lot of them are now part of the Cloud Native Computing Foundation (CNCF)

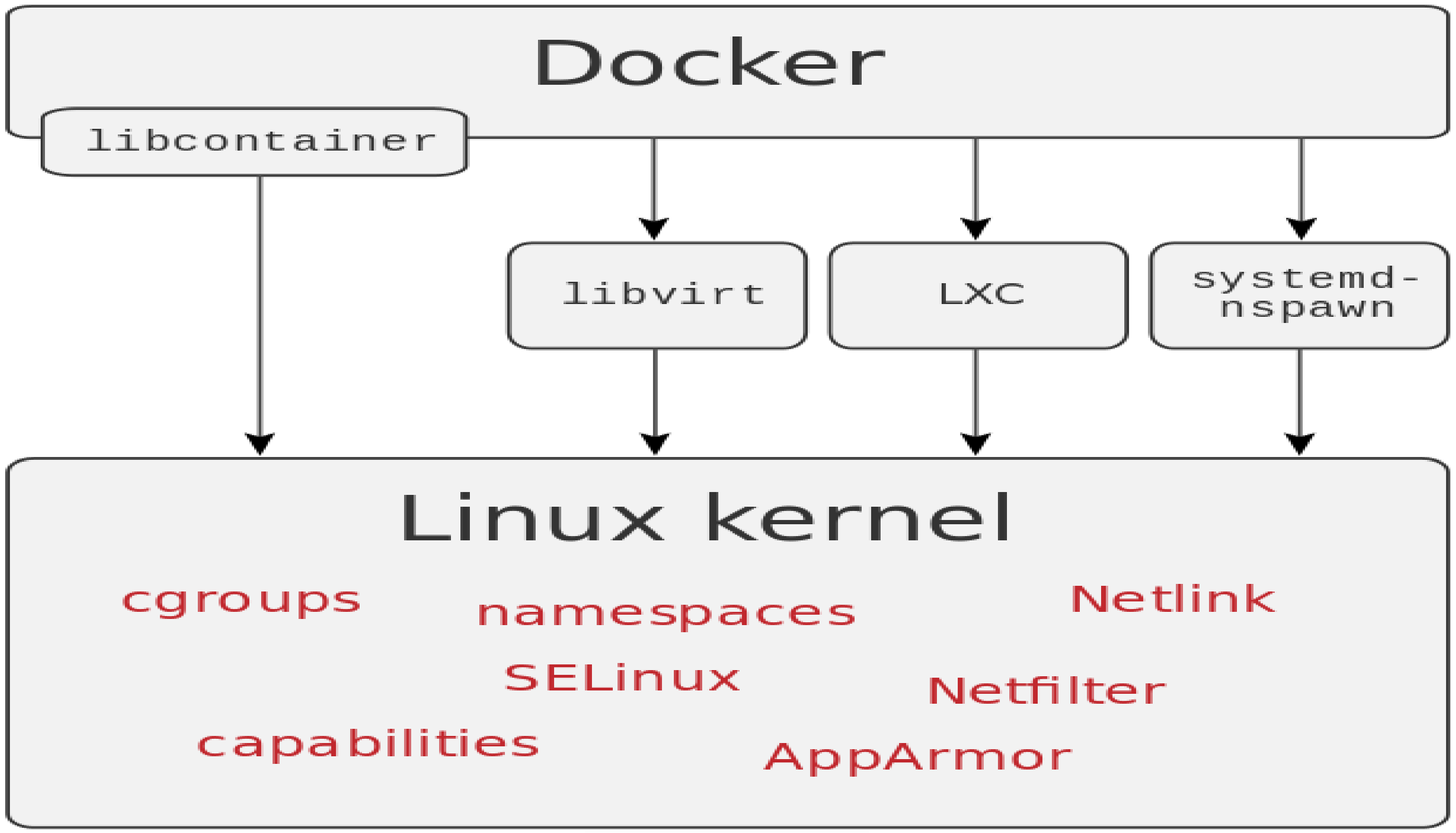
Docker

- Docker is a set of platform as a service products that use OS-level virtualization to deliver software in packages called container.
- Containers are isolated from one another and bundle their own software, libraries and configuration files.
- They can communicate with each other through well defined channels
- All containers are run by a single operating-system kernel

Container

- Docker container is a standardized unit which can be created on the fly to deploy a particular application or environment.
- It can be ubuntu/centos/or some app container (nginx)
- Containers are light-weight
- Docker takes advantage of several features of linux kernel to deliver its functionality
- Question: If we want to use container for application what are the org requirements??

Docker internals



Namespaces

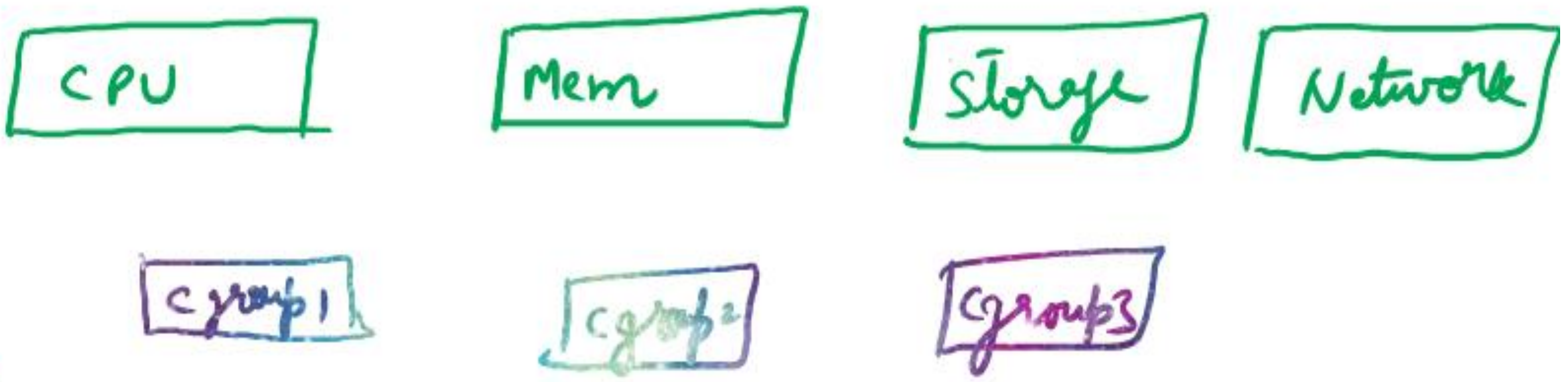
- Docker makes use of kernel namespace to provide the isolated workspace called container
- When we run container, docker creates a set of namespaces for that container. These namespace provides a set of isolation.
- Each aspect of container run in separate namespace and its access is limited to that namespace
  - PID namespace for process isolation
  - NET namespace for managing network interface
  - IPC namespace for managing access to IPC resources
  - MNT namespace for managing filesystem mount points
  - UTS namespace for isolation kernel and version identifier

Cgroups

- Docker makes use of kernel control group for resource allocation and isolation
- A cgroup limits an application to a specific set of resources
- Control group allow Docker Engine to share available hardware resources to containers and optionally enforce limits and constraints
  - Memory cgroup
  - CPU cgroup
  - Devices cgroup
  - BlkIO cgroup

. .

Docker grounds up

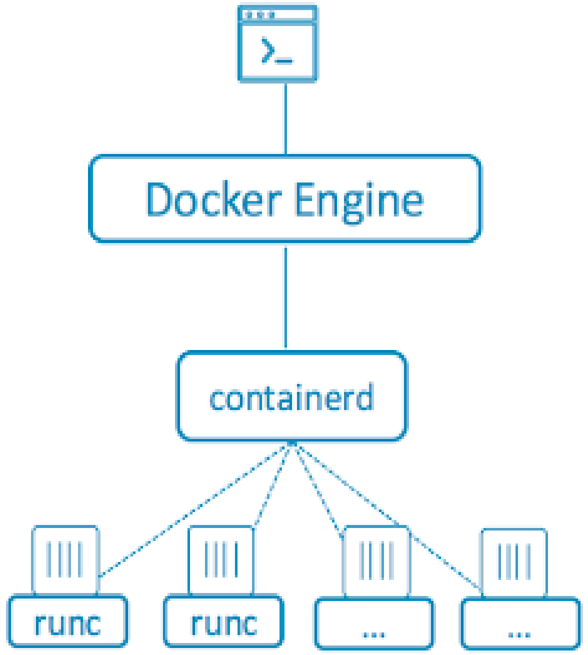


- CPU
- Mem
- storage(blk)
- Network

Union File systems

- Union file systems operates by creating layers, making them lightweight and fasters
- Docker Engine uses UnionsFS to provide the building blocks for containers
- Union filesystem works on the top of the other file system. It works on mounting mechanism
- In other words, it mounts multiple directories to a single root.
- AUFS,OverlayFS are few popular UnionFS

• •



Same Docker UI and commands

User interacts with the Docker Engine

Engine communicates with containerd

containerd spins up runc or other OCI compliant runtime to run containers

Containerd

- **What Is containerd?**
  - containerd is a Docker-developed container runtime that manages the life cycle of a container on a physical or virtual machine (i.e., a host). It creates, starts, stops, and destroys containers. It can also pull container images from container registries, mount storage, and enable networking for a container.
  - containerd is a daemon, meaning it’s a computer program that runs as a background process rather than being under the direct control of an interactive user. It’s available for both Linux and Windows.
  - containerd manages the complete container life cycle of its host system—from image transfer and storage to container execution and supervision to low-level storage to network attachments and more.

containerd

- containerd is a Docker-made runtime solution. This daemon is available for Linux and Windows OSes. As part of the Docker project, containerd manages image transfer and storage, as well as container creation, execution and supervision.
- Kubernetes does not need the entire Docker platform to use containerd. With the CRI compatibility plugin, Kubernetes and containerd can communicate directly.

Open Container Initiative (OCI)

- Docker and other important container industry actors established the Open Container Initiative (OCI) in 2015. The OCI aims to create standards for container formats and runtimes. Currently, the OCI has two specifications:
  - image-spec - the image specification that outlines how to create an OCI-compliant image.
  - runtime-spec - the runtime specification for unpacking the filesystem bundle.

runC

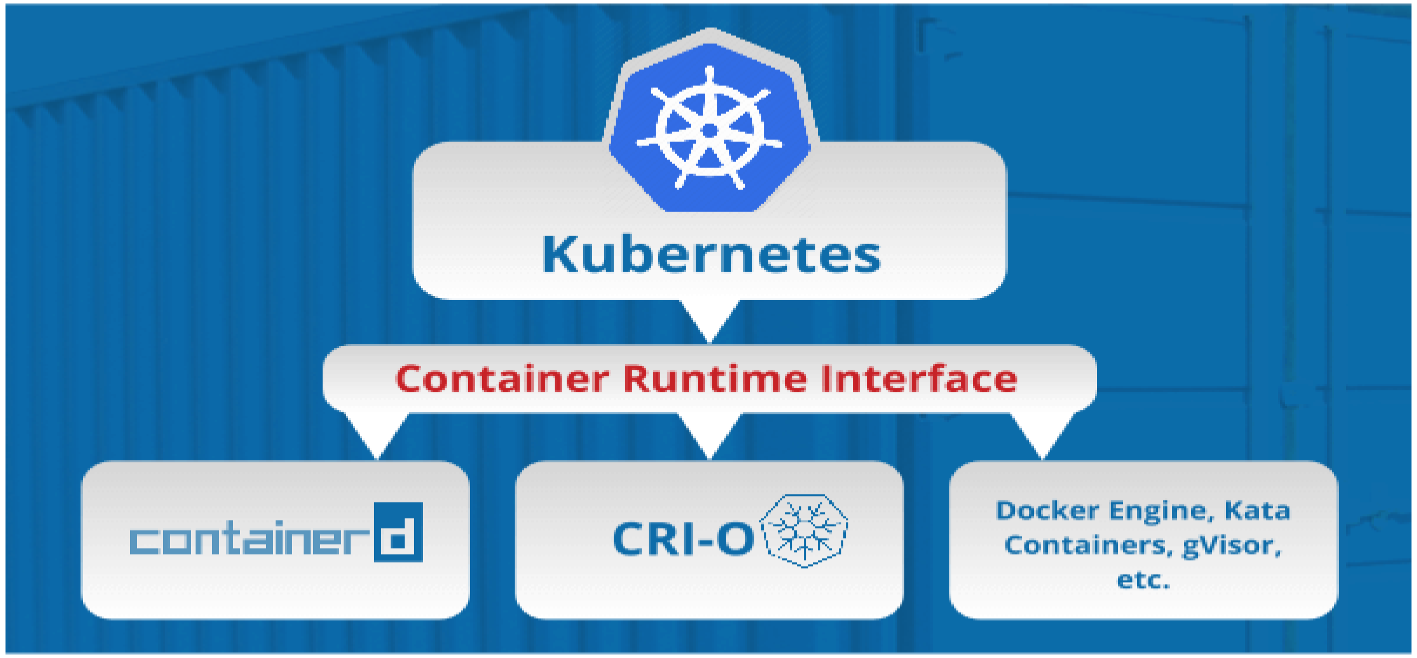
- runC is a universal container runtime created by Docker. Although it is a part of the Docker set of tools, it does not require Docker platform to run.
- Some important features of runC are:
  - Full Linux namespaces support.
  - Native support for Linux security features, such as AppArmor, SELinux, etc.
  - Windows 10 containers native support.
  - Containers that runC creates and manages are OCI compliant.

Container Runtime Interface (CRI)

- Although Kubernetes is a container orchestration platform, at the lowest level, it also needs to create and manage containers. To achieve this, Kubernetes uses container runtimes.
- In the beginning, Docker Engine was the only available runtime on the platform. But the popularity of containerization resulted in competing solutions and the need for Kubernetes to support them all. With the Container Runtime Interface plugin, Kubernetes can communicate with all major runtimes.

•

Container Runtime Interface


Kubernetes
Container Runtime Interface
containerd
CRI-O
Docker Engine, Kata Containers, gVisor, etc.