Published in techbeatly

Nived Velayudhan Follow
Jul 13, 2021 · 3 min read · Listen

CONTAINER RUNTIMES — Deep Dive

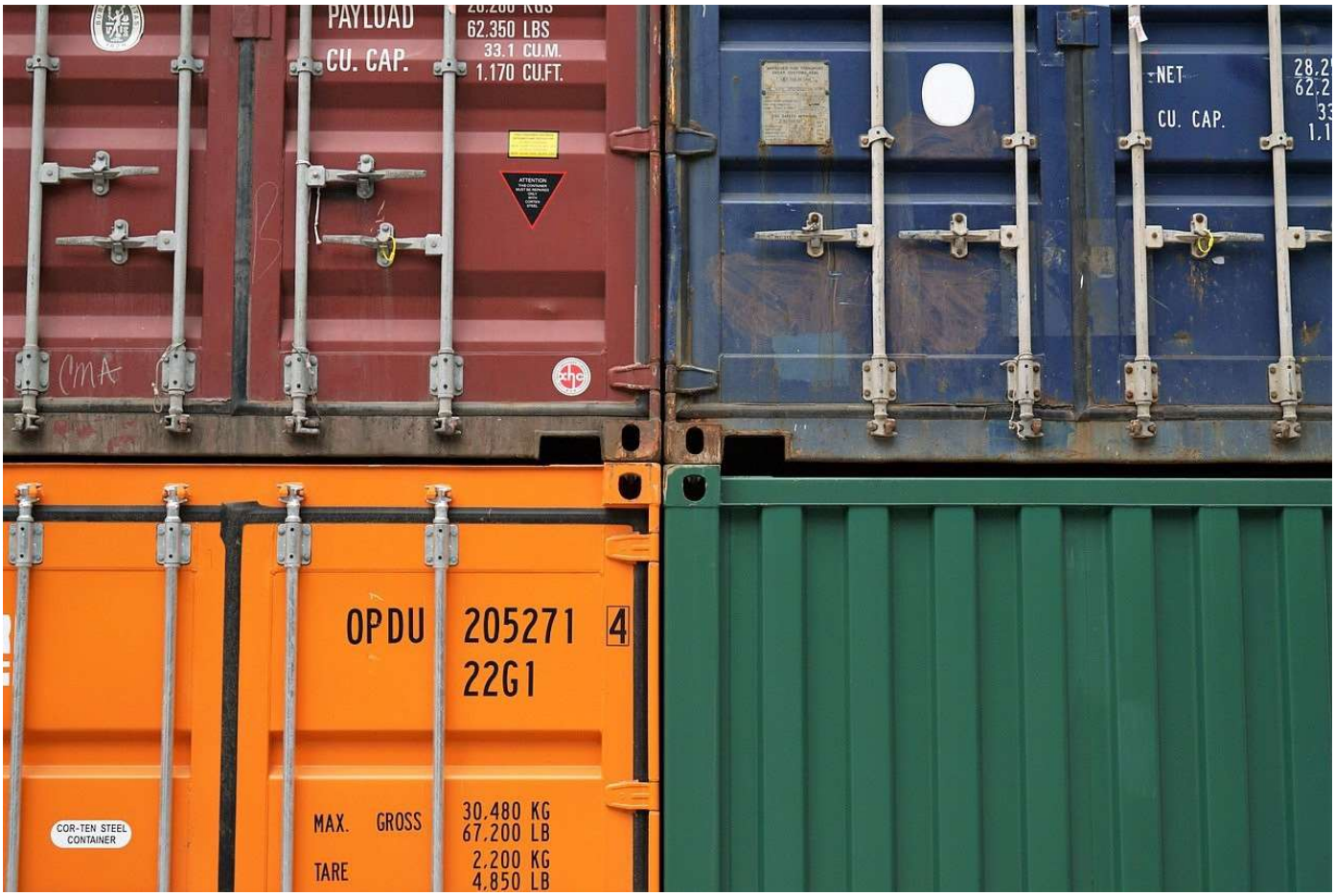

Image by Pexels from Pixabay

So what really happens in the backend when we pass the “docker run” command? Here is an overview for you — step-by-step:

2. The image is then extracted onto a copy-on-write filesystem and all the container layers overlay each other to create a merged filesystem.
3. Preparing a container mount point
4. Setting the metadata from container image like overriding CMD, ENTRYPOINT from user inputs, setting up SECCOMP rules, etc to ensure container runs as expected.
5. The kernel is alerted to assign some sort of isolation like process, networking & filesystem to this container (namespaces).
6. The kernel is also alerted to assign some resource limits like CPU or memory limits to this container (cgroups)
7. Pass system call (syscall) to the kernel to start the container.
8. Making sure SElinux/AppArmor is set up properly.

All of the above is taken care of by the container runtimes. When we think about container runtimes, the things that come to mind are probably runc, lxc, containerd, rkt, cri-o etc. Well, you are not wrong, these are container engines and container runtimes, each of these is built for different situations.

Container runtimes focus more on running containers, setting up namespace and cgroups for containers and are also called lower-level container runtimes and the ones that focus on formats, unpacking, management, and sharing of images and provide APIs for developers needs are called higher-level container runtimes or container engine.

Open Container Initiative (OCI):

The Open Container Initiative (OCI) is a Linux Foundation project with its purpose to design certain open standards or a structure around how to work with container runtimes and container image formats. It was established in June 2015 by Docker, rkt, CoreOS, and other industry leaders.

The goal of this specification is to enable the creation of interoperable tools for building, transporting, and preparing a container image to run.

The high-level components of the spec include:

- Image Manifest — a document describing the components that make up a container image
- Image Index — an annotated index of image manifests
- Image Layout — a filesystem layout representing the contents of an image
- Filesystem Layer — a changeset that describes a container's filesystem
- Image Configuration — a document determining layer ordering and configuration of the image suitable for translation into a runtime bundle
- Conversion — a document describing how this translation should occur
- Descriptor — a reference that describes the type, metadata and content address of referenced content

2. Runtime specification (runtime-spec):

This Specification aims to specify the configuration, execution environment, and lifecycle of a container. The container configuration is specified in the config.json file for all supported platforms and details the field that enables the creation of a container. The execution environment is specified along with the common actions defined for a container's lifecycle to ensure that applications running inside a container have a consistent environment between runtimes.

The Linux container specification uses various kernel features like namespaces, cgroups, capabilities, LSM, and filesystem jails to fulfill the spec.

Note: *Information about image-spec and runtime-spec on OCI has been taken from the*


Originally published at LinkedIn.