A Developers Introduction to Containers and Orchestration

—

nigel
Developer Advocate, New York City

Everything you wanted to know about how containers work, but didn't want to ask

—

nigel
Developer Advocate, New York City

Containers under the hood

“Put most simply, a **process** is an instance of an executing program…processes are the entities among which the kernel must share the various resources of the computer.”

Michael Kerrisk, *The Linux Programming Interface*

A **container** is a process, or group of processes running in isolation.

Linux Kernel Features for Isolation

1. Namespaces
    a. Process ID (PID)
    b. Unix Time Sharing (UTS)
    c. Network
    d. Interprocess Communication (IPC)
    e. User
    f. Mount
2. Cgroups ~~Process Containers~~

Origins of containers

Some notable points

Chroot (1982)

Allowed a process to run in an isolated environment

Virtuozzo (2000)

Earliest commercial Operating System Level Virtualization

Solaris (2004)

First commercial use of the term Containers with Solaris Containers also known as Zones

Linux Containers, LXC (2008)

First widely-adopted* implementation of containers.

*Anecdotally, most engineer's first encounter with containers

Docker, CNCF, and OCI

Docker didn’t invent containers -- they made them easier to use.

- Docker’s original implementation used LXC until they built their own engine
- Docker, Inc. donated their engine to the Cloud Native Computing Foundation in order to form the Open Container Initiative
- The Open Container Initiative provides a standard for implementing containers in a way that is compatible across platforms and vendors

Implementing containers

Container Ecosystem

1. Images
2. Runtimes
3. Engines

> “…an OCI Image, consists of a manifest, an image index (optional), a set of filesystem layers, and a configuration.”

Open Container Initiative, *Image Format Specification*

> “The Runtime Specification outlines how to run a “filesystem bundle” that is unpacked on disk.”

Open Container Initiative, *Runtime Specification*

But what about the rest?

A container engine’s tasks include

1. Receiving user input
2. Managing storage on the host
3. Calling the container runtime
4. Providing and communicating with APIs for plugins and orchestration

Example engines include docker, Singularity, rkt, LXD, Pouch, and podman