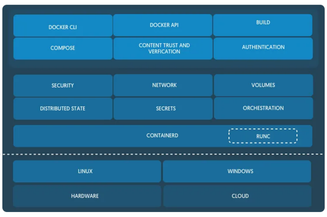
What is containerd ?

MICHAEL CROSBY
Aug 07 2017

We have done a few talks in the past on different features of containerd, how it was designed, and some of the problems that we have fixed along the way. Containerd is used by Docker, Kubernetes CRI, and a few other projects but this is a post for people who may not know what containerd actually does within these platforms. I would like to do more posts on the featureset and design of containerd in the future but for now, we will start with the basics.

I think the container ecosystem can be confusing at times. Especially with the terminology that we use. Whats this? A runtime. And this? A runtime... containerd as the name implies, not contain nerd as some would like to troll me with, is a container daemon. It was originally built as an integration point for OCI runtimes like runc but over the past six months it has added a lot of functionality to bring it up to par with the needs of modern container platforms like Docker and Kubernetes.



Since there is no such thing as Linux containers in the kernelspace, containers are various kernel features tied together, when you are building a large platform or distributed system you want an abstraction layer between your management code and the syscalls and duct tape of features to run a container. That is where containerd lives. It provides a client layer of types that platforms can build on top of without ever having to drop down to the kernel level. It's so much nicer towork with Container, Task, and Snapshot types than it is to manage calls to clone() or mount().

Containerd was designed to be used by Docker and Kubernetes as well as any other container platform that wants to abstract away syscalls or OS specific functionality to run containers on linux, windows, solaris, or other OSes. With these users in mind, we wanted to make sure that containerd has only what they need and nothing that they don't. Realistically this is impossible but at least that is what we try for. Things like networking are out of scope for containerd. The reason for this is, when you are building a distributed system, networking is a very central aspect. With SDN and service discovery today, networking is way more platform specific than abstracting away netlink calls on linux. Most of the new overlay networks are route based and require routing tables to be updated each time a new container is created or deleted. Service discovery, DNS, etc all have to be notified of these changes as well. It would be a large chunk of code to be able to support all the different network interfaces, hooks, and integration points to support this if we added networking to containerd. What we did instead is opted for a robust events system inside containerd so that multiple consumers can subscribe to the events that they care about. We also expose a task API that lets users create a running task, have the ability to add interfaces to the network namespace of the container, and then start the container's process without the need for complex hooks in various points of a container's lifecycle.

Another area that has been added to containerd over the past few months is a complete storage and distribution system that supports both OCI and Docker image formats. You have a complete content addressed storage system across the contained API that works not only for images but also metadata, checkpoints, and arbitrary data attached to containers.

We also took the time to rethink how "graphdrivers" work. These are the overlay or block level filesystems that allow images to have layers and you to perform efficient builds. Graphdrivers were initially written by Solomon and I when we added support for devicemapper. Docker only supported AUFS at the time so we modeled the graphdrivers after the overlay filesystem. However, making a block level filesystem such as devicemapper/lvm act like an overlay fillesystem proved to be much harder to do in the long run. The interfaces had to expand over time to support different features than what we originally thought would be needed. With contained, we took a different approach, make overlay filesystems act like a snapshotter instead of vice versa. This was much easier to do as overlay filesystems provide much more flexibility than snapshotting filesystems like BTRFS, ZFS, and devicemapper as they don't have a strict parent/child relationship. This helped us build out a smaller interface for the snapshotters while still fulfilling the requirements needed from things like a builder as well as reduce the amount of code needed, making it much easier to maintain in the long run.

So what do you actually get using containerd? You get push and pull functionality as well as image management. You get container lifecycle APIs to create, execute, and manage containers and their tasks. An entire API dedicated to snapshot management. Basically everything that you need to build a container platform without having to deal with the underlying OS details. I think the most important part of containerd is having a versioned and stable API that will have bug fixes and security patches backported.

> What's #containerd? All you need to know about #Docker's open and reliable #container runtime
>
> CLICK TO TWEET

Learn more about containerd:

- Check out the containerd GitHub Repo
- Join the containerd Slack channel
- Read about Docker's Open Source projects

containerd, Docker runtime, graphdrivers, Kubernetes, oci, runc

Be the first to write a comment.

Leave a Reply

Your name (required)

Your name

Your email address(required, but will not be published)

Your email address

Your website if you have one (not required)

Your website url

Your comment

Your comment

Post comment

Related Posts

Docker and CNCF Join Forces for "Container Garage" Event Series
By William Quiviger March 17 2021

Docker Compose: From Local to Amazon ECS
By Anca Iordache March 15 2021

Guest Post: Calling the Docker CLI from Python with Python-on-whales
By Gabriel de Marmiesse March 11 2021

Post Tags

- containerd
- Docker runtime
- graphdrivers
- Kubernetes
- oci
- runc

Categories

- All
- Products
- Community
- Engineering
- Company