containerd vs. Docker: Understanding Their Relationship and How They Work Together

Savannah Ostrowski

Posted

Mar 27, 2024

Post Tags

containerd
containers
Docker
open source
Popular Topics
vs

Categories

Community
Company
Engineering
Products

During the past decade, containers have revolutionized software development by introducing higher levels of consistency and scalability. Now, developers can work without the challenges of dependency management, environment consistency, and collaborative workflows.

When developers explore containerization, they might learn about container internals, architecture, and how everything fits together. And, eventually, they may find themselves wondering about the differences between containerd and Docker and how they relate to one another.

In this blog post, we'll explain what containerd is, how Docker and containerd work together, and how their combined strengths can improve developer experience.

What's a container?

Before diving into what containerd is, I should briefly review what containers are. Simply put, containers are processes with added isolation and resource management. Containers have their own virtualized operating system with access to host system resources.

Containers also use operating system kernel features. They use namespaces to provide isolation and cgroups to limit and monitor resources like CPU, memory, and network bandwidth. As you can imagine, container internals are complex, and not everyone has the time or expertise to become an expert in the low-level bits. This is where container runtimes, like containerd, can help.

What's containerd?

In short, containerd is a runtime built to run containers. This open source tool builds on top of operating system kernel features and improves container management with an abstraction layer, which manages namespaces, cgroups, union file systems, networking capabilities, and more. This way, developers don't have to handle the complexities directly.

In March 2017, Docker pulled its core container runtime into a standalone project called containerd and donated it to the Cloud Native Computing Foundation (CNCF). By February 2019, containerd had reached the Graduated maturity level within the CNCF, representing its significant development, adoption, and community support. Today, developers recognize containerd as an industry-standard container runtime known for its scalability, performance, and stability.

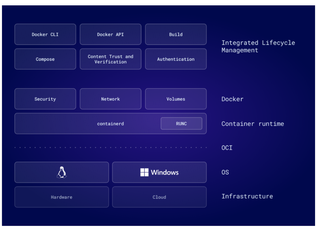
Containerd is a high-level container runtime with many use cases. It's perfect for handling container workloads across small-scale deployments, but it's also well-suited for large, enterprise-level environments (including Kubernetes).

A key component of containerd's robustness is its default use of Open Container Initiative (OCI)-compliant runtimes. By using runtimes such as runc (a lower-level container runtime), containerd ensures standardization and interoperability in containerized environments. It also efficiently deals with core operations in the container life cycle, including creating, starting, and stopping containers.

How is containerd related to Docker?

But how is containerd related to Docker? To answer this, let's take a high-level look at Docker's architecture (Figure 1).

Containerd facilitates operations on containers by directly interfacing with your operating system. The Docker Engine sits on top of containerd and provides additional functionality and developer experience enhancements.



How Docker interacts with containerd

To better understand this interaction, let's talk about what happens when you run the `docker run` command:

- After you select enter, the Docker CLI will send the `run` command and any command-line arguments to the Docker daemon (dockerd) via REST API call.
- dockerd will parse and validate the request, and then it will check that things like container images are available locally. If they're not, it will pull the image from the specified registry.
- Once the image is ready, dockerd will shift control to containerd to create the container from the image.
- Next, containerd will set up the container environment. This process includes tasks such as setting up the container file system, networking interfaces, and other isolation features.
- containerd will then delegate running the container to runc using a shim process. This will create and start the container.
- Finally, once the container is running, containerd will monitor the container status and manage the lifecycle accordingly.

Docker and containerd: Better together

Docker has played a key role in the creation and adoption of containerd, from its inception to its donation to the CNCF and beyond. This involvement helped standardize container runtimes and bolster the open source community's involvement in containerd's development. Docker continues to support the evolution of the open source container ecosystem by continuously maintaining and evolving containerd.

Containerd specializes in the core functionality of running containers. It's a great choice for developers needing access to lower-level container internals and other advanced features. Docker builds on containerd to create a cohesive developer experience and comprehensive toolchain for building, running, testing, verifying, and sharing containers.

Build + Run

In development environments, tools like Docker Desktop, Docker CLI, and Docker Compose allow developers to easily define, build, and run single or multi-container environments and seamlessly integrate with your favorite editors or IDEs or even in your CI/CD pipeline.

Test

One of the largest developer experience pain points involves testing and environment consistency. With Testcontainers, developers don't have to worry about reproducibility across environments (for example, dev, staging, testing, and production). Testcontainers also allows developers to use containers for isolated dependency management, parallel testing, and simplified CI/CD integration.

Verify

By analyzing your container images and creating a software bill of materials (SBOM), Docker Scout works with Docker Desktop, Docker Hub, or Docker CLI to help organizations shift left. It also empowers developers to find and fix software vulnerabilities in container images, ensuring a secure software supply chain.

Share

Docker Registry serves as a store for developers to push container images to a shared repository securely. This functionality streamlines image sharing, making maintaining consistency and efficiency in development and deployment workflows easier.

With Docker building on top of containerd, the software development lifecycle benefits from the inner loop and testing to secure deployment to production.

Wrapping up

In this article, we discussed the relationship between Docker and containerd. We showed how containers, as isolated processes, leverage operating system features to provide efficient and scalable development and deployment solutions. We also described what containerd is and explained how Docker leverages containerd in its stack.

Docker builds upon containerd to enhance the developer experience, offering a comprehensive suite of tools for the entire development lifecycle across building, running, verifying, sharing, and testing containers.

Start your next projects with containerd and other components by checking out Docker's open source projects and most popular open source tools.

Learn more

- Subscribe to the Docker Newsletter.
- Get the latest release of Docker Desktop.
- Have questions? The Docker community is here to help.
- New to Docker? Get started.

containerd, containers, Docker, open source, Popular Topics, vs

Related Posts

How Docker Streamlines the Onboarding Process and Sets Up Developers for Success
By Yiwen Xu January 22, 2025

Mastering Docker and Jenkins: Build Robust CI/CD Pipelines Efficiently
By Vladimir Mikhalev January 16, 2025

How to Dockerize a Django App: Step-by-Step Guide for Beginners
By Lance Haig January 8, 2025