docker

Docker with Kubernetes

Building from the foundation with Docker Developer Tools

Building from the foundation with Docker Developer Tools

Kubernetes is a powerful tool for orchestrating container-based applications. You can use Kubernetes to test, deploy, and scale your containerized apps. The automation capabilities built into Kubernetes dramatically minimize labor costs and eliminate user error for an efficient and portable framework that's equally at home in the server room or cloud. But if you want to get the most out of Kubernetes, your applications must be ready.

If you take a close look under the hood, you'll see that Kubernetes isn't a single tool – it's actually a collection of several tools and components that combine to create a complete and seamless container orchestration framework. At the heart of the container environment is an essential feature known as the *container engine* or *container runtime*. The container engine is the component that runs and manages the container on the host system (Figure 1). In other words, the container engine handles tasks such as loading container images, isolating and monitoring local system resources, and managing the container lifecycle.

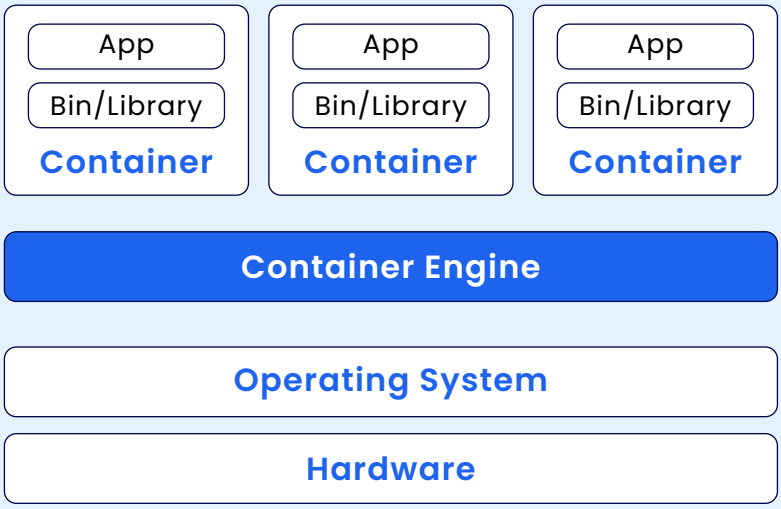

Figure 1: The container engine (or container runtime) is a key component in the container architecture.

Container Development and Orchestration Together

Docker Engine was the original Kubernetes container engine. In fact, Kubernetes was built around the foundation of Docker Engine as a tool for managing container operations in the host environment. Now, Kubernetes offers a few other container engine choices, including Docker Engine, as well as the containerd engine, which was also created by the Docker team and donated by Docker to the Cloud Native Computing Foundation.

A container engine gets you off the ground, allowing basic functionality of the Kubernetes environment, but if you're a professional developer, you'll want something more. Tinkering with do-it-yourself containers can be time-consuming and often leads to duplication of efforts and security problems. If you write containerized applications for a living, or if you are tasked with converting conventional applications to a containerized setting, you'll need a more complete environment for the development process. This environment should include tools that deliver convenience, enhanced security, and features such as auditing and centralized management. Docker builds on its long history with Kubernetes to offer a complete platform of tools for industrial-strength container development.

The Docker Developer Ecosystem

Docker is much more than the engine at the heart of Kubernetes. The tools of the Docker ecosystem provide a complete environment for the container developer, allowing you to dial up efficiency and dial down security risks. Docker's tools also add value in the testing and development phase. Docker's containerization ensures consistency throughout the product lifecycle. A stable and consistent development environment minimizes compatibility issues, streamlines testing, and reduces the likelihood of bugs or errors making their way into production. Docker tools help you optimize cost, trim down resource usage, and build more secure container applications in a hybrid or multi-cloud setting.

Docker's developer tools come out-of-the-box ready with a variety of tools, including:

- **Docker Desktop** – a local development package that combines Docker Engine, build tools (Build and Buildkit), Docker Compose, one-click Kubernetes, graphic user environments, and more. Docker Desktop lets you manage containers, applications, and images from your desktop, offering a single interface for more intuitive development.
- **Docker Scout** – a unified security solution that helps to identify and fix vulnerabilities in Docker containers.
- **Docker Hub** – a container registry for developers who wish to find, run, and share Docker container images. Docker Hub is the world's largest container image registry, with over 100,000 images. It includes Docker Trusted Open Source Content, a collection of secure base images sourced from vetted publishers.
- **Docker Extensions** – a collection of interfaces and extensions for integrating Docker with third-party tools.

One application that is not in the Docker Developer Tools is Docker Swarm, a container orchestration utility that used to be part of the Docker toolkit. Docker Swarm was sold and is no longer developed by Docker. Although Docker works with a number of orchestration alternatives (including Swarm, ECs, Nomad, and more), Kubernetes is currently the most popular Docker orchestration tool and is the primary focus of the Docker orchestration environment.

The tools of the Docker environment work together with Kubernetes to optimize container development, deployment, and management. Docker packages applications and their dependencies into portable containers, ensuring consistent behavior across different computing environments. Kubernetes orchestrates these containers, automating tasks such as scaling, load balancing, and self-healing.

Efficiency and Flexibility

For a professional Kubernetes developer, time is money. Docker's tools accelerate the development process by anticipating the developer's needs, adding convenience, automation, and centralized management to support cross-platform development.

Docker Desktop serves as a central point for managing the Docker environment, allowing the developer to access many of the most important options with only a couple mouse clicks (Figure 2). You can run Docker Desktop on Windows, macOS, and Linux systems, allowing for easy cross-platform collaboration. Docker Desktop handles port mappings, file system integrations, and other default settings that simplify integration with the host machine. You can deploy images from the desktop to AWS or Azure and connect to services on the localhost of the host computer. Docker Desktop also offers one-click installation and configuration, and it comes with an integrated K8 runtime and load balancer.

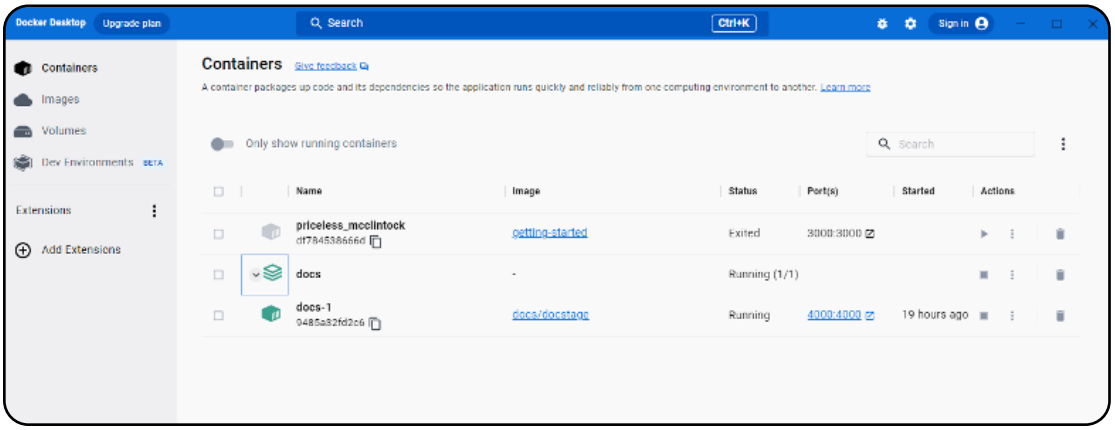

Figure 2: Manage containers in Docker Desktop.

Docker Hub provides many container images and image templates to save you time and help you avoid reinventing the wheel (Figure 3). Meanwhile, Docker Compose brings the declarative model of configuration automation to container development.

The Docker Extensions collection offers dozens of extensions that save you time by connecting the Docker environment with applications you're using right now. Included in the extensions are security tools, monitoring tools, and development aids. Other extensions prepare your containers for the OpenShift cloud or interface with the Oracle or PostgreSQL database engines.

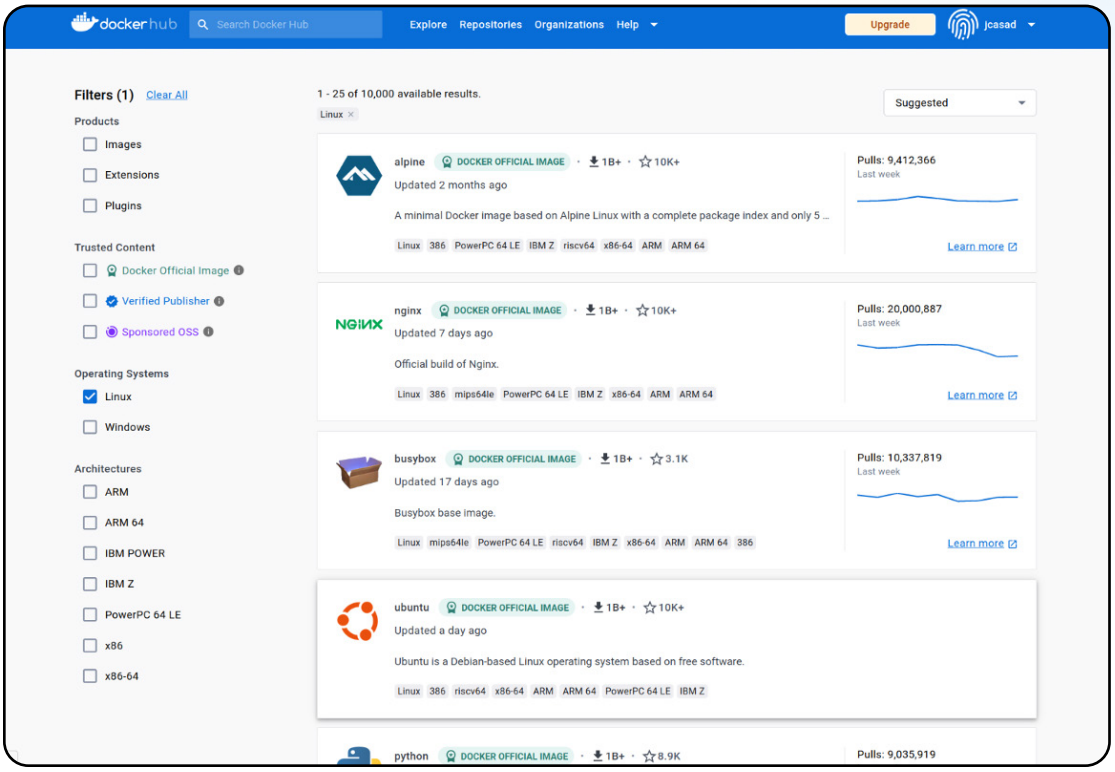

Figure 3: Browsing for a continer image at Docker Hub.

Security

Security is an important consideration for any developer. But the extra layer of complexity that comes with the container space makes it even more important to get security right. The Docker toolset comes with a variety of powerful security features for locking down the development environment–whether you're a single user or part of a large development team. For instance, Docker Desktop provides automated security patches, VPN integration, local file system access controls, and integrated container image vulnerability scanning.

Docker ensures that developers have access to trusted and reliable building blocks for assembling a container environment. The Docker Trusted Open Source Content system includes the Docker Official Images collection, a curated set of verified repositories hosted at Docker Hub. Content publishers can also join the Docker Verified Publisher program, a vetted group of content creators who are known to offer safe and reliable content. Additionally, Docker Content Trust (DCT) uses digital signatures to verify the content sent or received from remote Docker registries. DCT helps you ensure that a Docker image comes from a reliable source.

Developers often work in teams, and the best practice is to ensure that the team uses a pre-approved set of methods and components for uniform development. Docker provides a set of enhanced security features, known as Hardened Docker Desktop, that let the admin set security controls to manage the development process. Hardened Docker Desktop features include:

- **Settings management** – manage and control Docker Desktop configuration settings.
- **Enhanced container isolation** – prevent containers from running as root in the Docker Desktop VM, and ensure that settings management controls can't be bypassed or modified in containers.
- **Registry access management** – control which registries developers can access.
- **Image access management** – control which images developers can pull from Docker Hub.

After you've hardened the development environment, you still need to look for errors and vulnerabilities that could slip in as part of the development process. The Docker Scout tool offers a whole new approach to secure container development, with the goal of resolving security issues before they make it into production. Docker Scout checks your containers automatically, analyzing every image layer to identify vulnerabilities, suggest fixes, and offer remediation advice.

Logging and Auditing

Docker's audit log records container-related events at the organization or repository level. The audit log lists all changes, with the date of the change and who originated it. The log also records any changes to configuration or privacy settings. You can use the audit log to ensure your Docker environment is secure and protected from unauthorized use. Docker also logs information at the container level, allowing the developer or admin to look for problems and keep watch over potential changes.

Conclusion

Docker and Kubernetes have a common history and evolved together into the powerful and well-integrated container orchestration solution you see today. Docker Engine was the first Kubernetes container engine, and it remains a popular and powerful Kubernetes container runtime option.

Now, when using Docker and Kubernetes, you can quickly deploy and scale applications, reducing time to market for new features and services. This agility allows development teams to respond promptly to ever changing market demands and user needs.

If you develop containerized applications, you'll need something more than a container engine to work productively and keep your applications safe. Docker provides powerful developer tools that bring convenience, efficiency, and security to Kubernetes development. Docker offers a number of different subscription levels for individuals and teams looking for an edge in container development. The tools of Docker increase productivity, enhance security, and lock down the development environment. Contact the Docker team for more information on the deep integration of Docker with Kubernetes.

Get started with Docker today!

Learn more about the #1 most used development tool at www.docker.com.