Learn about Containers

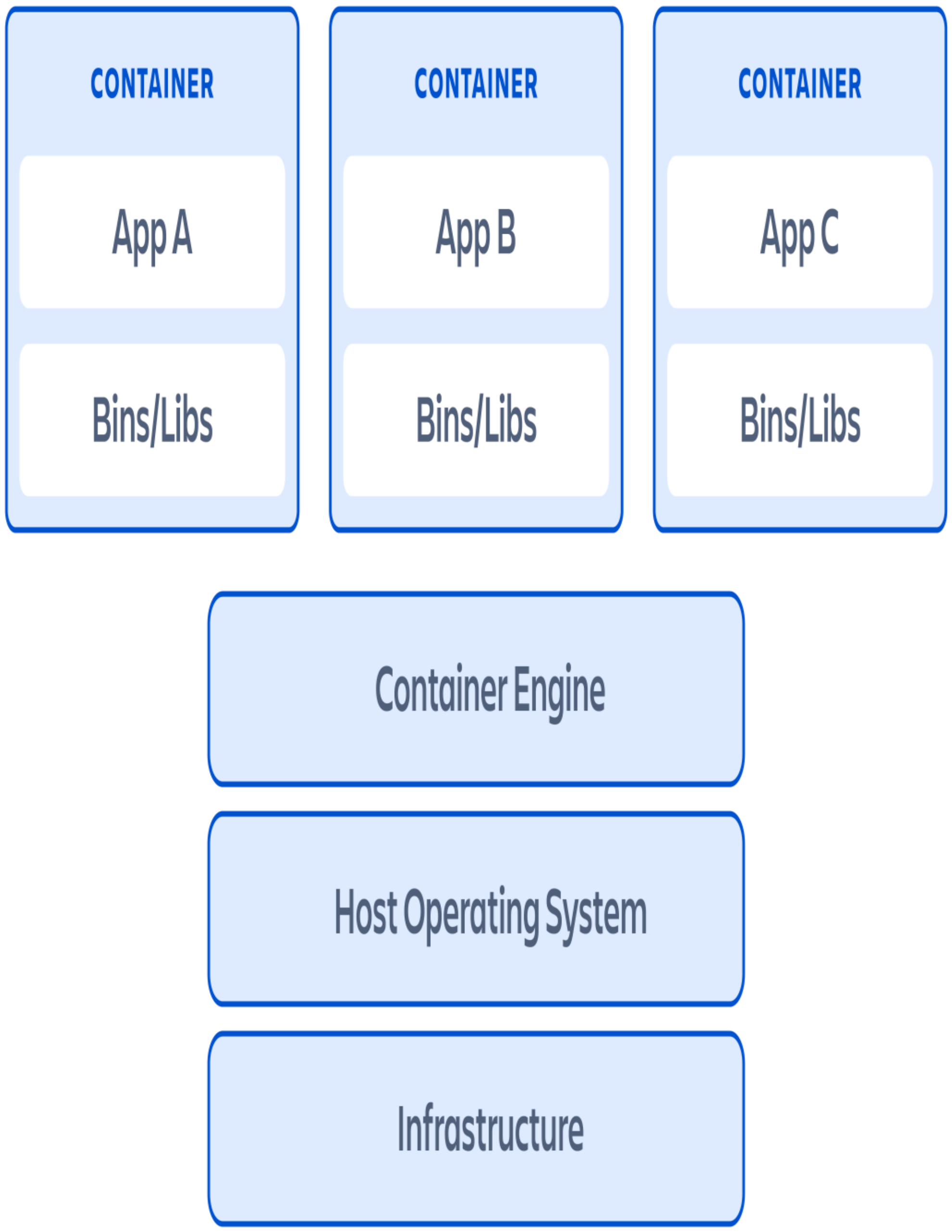
By

Stanley Stephen

Linkedin: https://www.linkedin.com/in/contactstanley/
Github: https://github.com/stanleymca
Email: s.stanley.mca@gmail.com

What is a Container?

A container is a package of software that includes all dependencies: code, runtime, configuration, and system libraries so that it can run on any host system. At runtime, the container is also granted its own isolated slice of operating system resources like CPU, RAM, Disk, and Networking.



Stanley Stephen, M.C.A.,
Linkedin: https://www.linkedin.com/in/contactstanley/
Github: https://github.com/stanleymca
Email: s.stanley.mca@gmail.com

Why do you need Containers?

Containers are extremely useful in scaling DevOps efficiency across multiple codebases and developer headcount. Containers help scale with developer headcount by ensuring that containerized code works consistently on any machine the container is deployed to. Newly hired developers can get a container of the application they will work on and immediately be up to speed to start development.

Multiple codebase scenarios like a micro service architecture benefit from containers by keeping each microservice portable and isolated to a container. Micro service containers coupled with a robust container management platform makes cluster management and orchestration much easier. This enables dynamic scaling features that adapt to increased traffic or load on an application.

How do they work?

To first understand containers we must discuss virtualization. Virtualization is the act of dividing shared computational resources: CPU, RAM, Disk, and Networking into isolated resources that are unaware of the original shared scope. When virtualizing a machine using either virtual machines (VMs) or Containers, the host machine's resources are essentially cut into slices for the virtualized components to use.

Containers virtualize a machines operating system at the user space level. Virtualizing user space leverages the existing mechanisms that divide system resources between separate user accounts and programs on an operating system.

Container systems generally have a separate orchestration utility command or server daemon. DevOps team members will interface with the orchestration utility to create and manage the individual containers. The orchestration utility is responsible for dividing up the host operating systems user space resources, allocating those resources to containers, and then executing and monitoring the containers.

Use Cases

Containers solve one of the classic causes of software distribution failure: "it doesn't works on my machine." In this unfortunate scenario, application code behaves as expected on one machine, but when executed on another, it fails. This is usually due to subtle differences between the two machines. These differences can be mismatched dependency versions or other configuration discrepancies. Containers solve this by creating a static repeatable package of everything the application code needs to execute.

Stanley Stephen, M.C.A.,
Linkedin: https://www.linkedin.com/in/contactstanley/
Github: https://github.com/stanleymca
Email: s.stanley.mca@gmail.com

Containers vs VMs

Containers and VMs are very similar in their goals. They both help distribute application software in a repeatable isolated package. Where they differ is in how much of the hardware stack they attempt to virtualize. VM's simulate the entire machine and operating system. This means VMs have simulated CPU, RAM, Filesystems and Network resources.

Containers only virtualize the user space of an existing operating system. In this sense, containers are much more lightweight than VMs. Containers can be utilized in an existing host operating system.

Container Runtime Tools

A container runtime is the set of tools that is used to specify the dependencies that need to be packaged up into a container. The runtime then builds the container and executes it.

Docker

The most popular and widely used container runtime. Docker kickstarted the modern DevOps explosion of container-based infrastructure. Docker has a public repository full of containers that house popular open source tools.

Linux Containers - LXC

Linux Containers use a collection of native open source Linux utilities to create a container runtime experience. The goal of Linux container is to offer a vendor neutral environment to develop container technologies. Docker actually uses LXC under the hood. Linux Containers focus on keeping the container experience as close as possible to that of a VM.

CRI-O

An emerging container specification targeted for the Kubernetes container management system that claims to be lightweight. CRI-O implements the open container initiative (OCI). OCI is a standard developed by contributors from companies like Red Hat, Intel, and IBM.

Rkt and Rktlet

Pronounced ‘rocket’, rkt is a container specification that takes a security-first approach. Rkt containers do not allow any insecurity functionality unless a user explicitly disables security features. Rkt is pod native which makes it a good fit for the Kubernetes container management system.

Container Management Platforms

Container management platforms are higher level tools which are used to orchestrate a group or cluster of runtime containers. Container Management Platforms use an underlying container runtime system. Some container management platforms allow a mix and match compilation of multiple container runtimes.

Kubernetes

Kubernetes is an open source container management system designed and released by Google. Kubernetes is used to create a distributed cluster of hosted containers. Kubernetes provides robust container cluster orchestration tools that includes health monitoring, deployment, failover, and auto scaling.

Docker

Docker also offers a suite of utilities to launch, orchestrate and manage a cluster of containers. Docker is so widely used that its container definition system is supported by many third party container management systems like Amazon ECS, and Kubernetes.

Amazon ECS

Amazon Elastic Container Service Is an Amazon cloud based service that makes it easy to manage Docker containers in a cluster. ECS has a powerful API and is tightly integrated with the rest of the Amazon cloud suite enabling powerful DevOps workflows.

Openshift

The Openshift container platform is an enterprise Red Hat Linux product. It is a hybrid product offering Kubernetes and Openshift cloud integration. Openshift is a trusted enterprise cloud platform that integrates with other Red Hat tools.

Stanley Stephen, M.C.A.,
Linkedin: https://www.linkedin.com/in/contactstanley/
Github: https://github.com/stanleymca
Email: s.stanley.mca@gmail.com

Conclusion

Containers are a critical part of the modern devops tool set. Regardless of what runtime or container platform you choose, the benefits of containers will help your team execute more effectively. Containers help streamline both the local development process and the remote deployment process while ensuring that you can deliver higher quality software product to your customers.

Stanley Stephen, M.C.A.,
Linkedin: https://www.linkedin.com/in/contactstanley/
Github: https://github.com/stanleymca
Email: s.stanley.mca@gmail.com