

Kubernetes for Newbies

Greg Horie

Overview

- Linux Containers
 - Quick intro / recap
 - System Containers vs. Application Containers
- Kubernetes - What is it?
- Kubernetes Pod
- Kubernetes Deployment
- Kubernetes Service
- Summary
- Future Topics

Linux Containers - Quick Intro / Recap

- A lightweight method for running multiple isolated applications under one Linux host
- Feature - Portable
 - Each container encapsulates its dependencies
 - Reduces conflicts and ensures consistent behavior across environments
- Feature - Efficient
 - Containers share the host's kernel
 - Results in lower overhead compared to traditional virtualization
- Feature - Scalable
 - Containers can be spun up or shut down quickly
 - Enables rapid deployment and scaling of applications
- Feature - Versioning / Rollback
 - Containers support versioning, packaging apps and dependencies together
 - Facilitates easy (automatable) rollbacks

System Containers vs. Application Containers

System Containers:

- Primarily focused on running system-level processes and services
- Serve as lightweight environments for system-level tasks
- Designed to encapsulate and deploy components of the operating system, including system daemons, background services, and other essential processes.
- Examples
 - LXC / LXK

System Containers vs. Application Containers

Application Containers:

- Geared towards encapsulating and deploying specific applications and their dependencies, fostering portability across different environments
- Primarily concerned with encapsulating application-specific resources, optimizing performance and scalability for the application itself
- Examples
 - Docker
 - Containerd
 - Podman
 - CRI-O
 - Kubernetes

Containers - Challenges

- Containers provide an isolated environment where an application can run
- To move these applications from isolated environments to production services requires some "extras"
- Challenges to consider:
 - Shared file systems, configurations, secrets
 - Networking
 - Load balancing
 - Scheduling - i.e. where to provision container workload
 - Distribution - i.e how to deploy the container workload
 - etc.

Kubernetes - What is it?

- K8s = Kubernetes
- K8s is an open-source container orchestration platform
- Automates the deployment, scaling, and management of containerized applications across a set of hosts (cluster)
- Supports load balancing, rolling updates, and application monitoring / scheduling

Kubernetes - History

- Kubernetes was originally developed by Google engineers
 - Written in the Go programming language
 - Based on an internal container orchestration platform called Borg
- Kubernetes was open-sourced by Google in 2014
 - Graduated as a project of the Cloud Native Computing Foundation (CNCF) in 2015
- Kubernetes gained widespread industry adoption
 - By 2017, it became the standard for container orchestration
- Continues to evolve with a strong community contributing to its development and improvement

Kubernetes - Design & Use Cases

Prep

Ubuntu 18.04

```
$ sudo apt update
```

```
$
```

CentOS 7

```
$ sudo yum check-update
```

```
$
```

Run a K8S Pod

- Most likely use MiniKube
- Consider spinning up a GKE cluster for demo (ACG)
- Demo running a single Pod

K8S Pod - Design & Use Cases

- Discuss how it's built
- Show how it leveraged the underlying container runtime

Run a K8S Deployment

- Demo a Deployment with multiple replicas
- Kill a pod
-

K8S Deployment - Design & Use Cases

Run a K8S Service

- Probably can only do a NodePort on MiniKube
- See what ACG provides for GKE cluster

K8S Service - Design & Use Cases

Summary

Possible Future Discussions

- Orchestration
 - Hashicorp Nomad
 - K8S ConfigMaps, Secrets, Persistent Volumes
 - K8S Ingress, Gateway
- Monitoring
 - Prometheus / Grafana
 - ELK / EFK
- Messaging
 - RabbitMQ / ActiveMQ
- Data Pipelines
 - Airflow / Dagster
- Other ideas welcome!



Backup Slides



Exercise 1 - ...

Setup:

\$

Try:

\$

