Kubernetes for Newbies

Greg Horie

Overview

- Containers
  - System Containers vs. Application Containers
- Kubernetes - What is it?
  - Design & Use Cases
- Instantiate a Pod
  - Design & Use Cases
- Deployment
  - Design & Use Cases
- Service
  - Design & Use Cases
- Summary
- Future Topics

Linux Containers - Quick Recap

- Run on a compute host (node) enabling process and resource isolation
- Portable - Each container encapsulates its dependencies, reducing conflicts and e
  behavior across different environments
- Efficient - Containers share the host's kernel, resulting in lower overhead compare
  virtualization
- Scalable - Containers can be spun up or shut down quickly, enabling rapid deploy
  applications
- Versioning / Rollback - Containers support versioning, packaging apps and depend
- This facilitates easy (automatable) rollbacks, enhancing reliability of software
- CI/CD Automation - Define infrastructure-as-code, facilitating continuous integratio
  delivery (CI/CD) pipelines

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 / LXD

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
- High level description
- Short history
-

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:

```
$
```