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 - Portability**
  - Each container encapsulates its dependencies
  - Ensures consistent behavior across environments
- **Feature - Efficiency**
  - Containers share the host's kernel
  - Lower overhead compared to traditional virtualization
- **Feature - Scalability**
  - Containers can be spun up or shut down quickly
  - Enables rapid deployment and scaling of applications
- **Feature - Versioning / Rollback**
  - Containers support versioning
  - Facilitates easy (automatable) rollbacks

System Containers vs. Application Containers

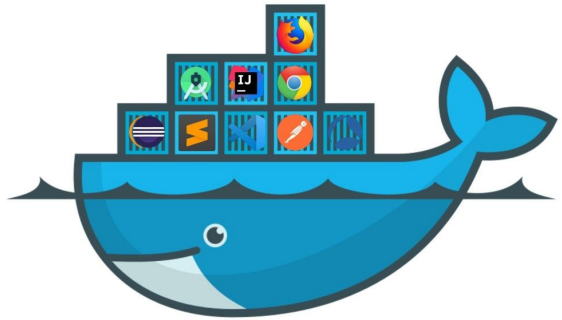
System Containers:

- Running system-level processes and services
    - Like a mini virtual machine
- A lightweight environment for system-level tasks
- Designed to encapsulate and deploy components of the operating system
    - System daemons - systemd, cron, rsyslog, etc.
- No goal to decouple services any more than a traditional VM or bare-metal host
- Examples
    - LXC / LXD

System Containers vs. Application Containers

Application Containers:

- Designed to encapsulate individual applications and their dependencies
    - Fosters portability across different environments
    - Enables a microservices architecture
- Optimizes performance and scalability for the application itself
    - Avoids oversubscribing resources for more efficient use of host resources
- Examples
    - Docker
    - Containerd
    - Podman
    - CRI-O
    - Kubernetes

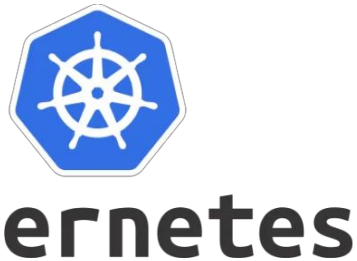


Containers - The Challenges

- Container runtime provides an a single host where an application can run
- Moving containerized applications from single hosts to production-worthy services requires some "extras"
- Challenges to consider:
  - Shared file systems, configurations, secrets
  - Networking across redundant hosts
  - Load balancing across instances of a service
  - Scheduling - i.e. where to provision container workload
  - Distribution - i.e how to deploy the container workload
  - etc.

Kubernetes - What is it?

- K8s = Kubernetes
- Kubernetes is Greek for helmsman or pilot
  - Following the Docker shipping metaphone
- 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



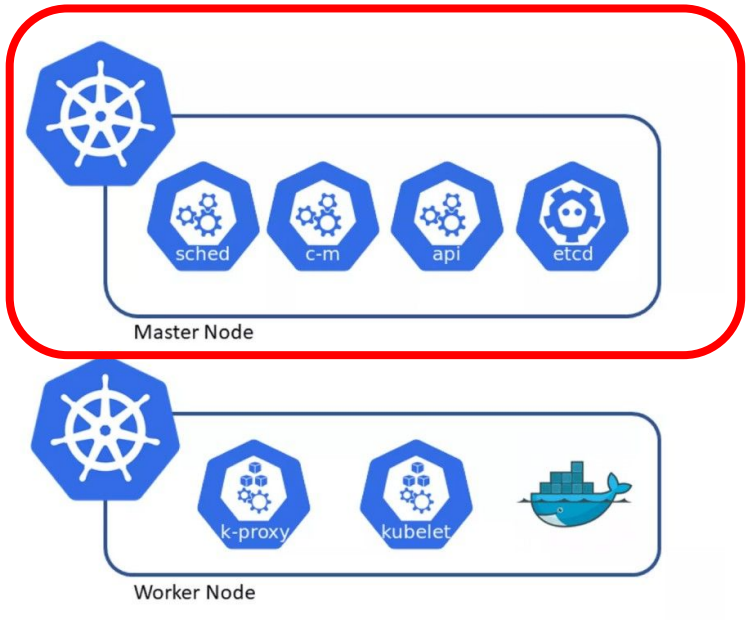
Kubernetes - History

- **2003-2004 - Borg**
  - Early orchestration platform built at Google to manage container-based applications
- **2014 - Kubernetes open-sourced**
  - Google engineers open sourced K8S (based on Borg)
    - Written in the Go programming language
- **2015 - Kubernetes 1.0 released**
  - Google partners with the Linux Foundation
    - Forms the Cloud Native Computing Foundation (CNCF)
  - CNCF goes on to host many open source projects - containerd, prometheus, etcd, etc.
- **2017 - Winner of the container wars**
  - Industry rallies around K8S - Docker, Microsoft AKS, Amazon EKS, etc.
- **Today**
  - Continues to evolve with a strong community contributing to its future

Kubernetes - Architecture

Control plane (master) nodes

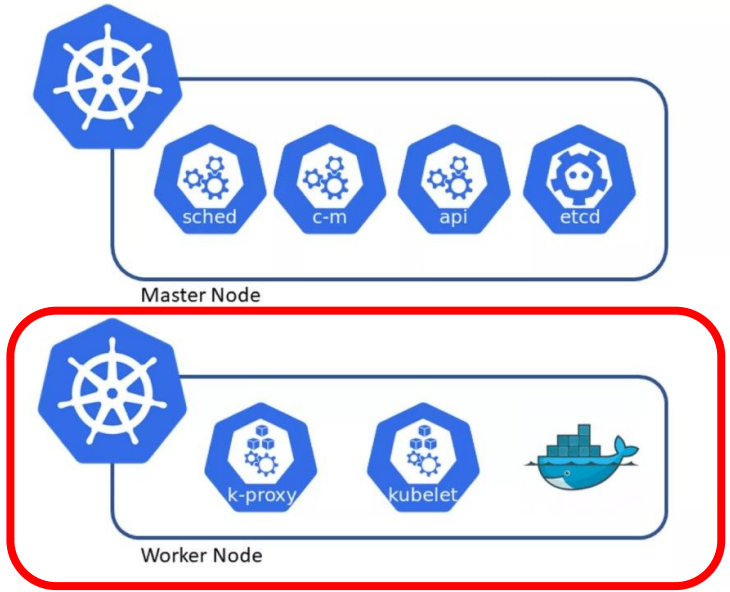
- **kube-apiserver** - Exposes Kubernetes API for cluster management.
- **kube-controller-manager** - Manages desired cluster state via controllers.
- **kube-scheduler** - Assigns Pods to nodes based on resources.
- **etcd** - Distributed key-value store for cluster data.
- … and many more depending on your environment.



Kubernetes - Architecture

Worker nodes

- **kubelet** - Agent running on workers, executing instructions from the control plane.
- **container runtime** - Software responsible for running containers (e.g. containerd).
- **kube-proxy** - Maintains network rules and enables communication across the cluster.
- **CNI (Container Network Interface)** - Plugin enabling Pod networking across nodes.



Minikube Prep

Note - Assumes container runtime installed (e.g. Docker)

Install minikube

```
$ curl -LO \
https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
```

```
$ sudo install minikube-linux-amd64 /usr/local/bin/minikube
```

Install kubectl

```
$ curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
```

```
$ sudo install kubectl /usr/local/bin/kubectl
```

Minikube Init

Start Cluster

```
$ minikube start --nodes 2 -p newbie-demo

$ minikube status -p newbie-demo

$ minikube ssh -p newbie-demo -n newbie-demo # control plan

$ minikube ssh -p newbie-demo -n newbie-demo-m02 # worker
```

Nodes Provisioned

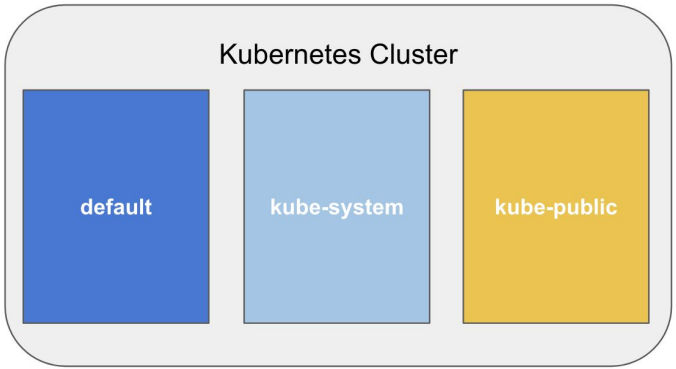
```
$ kubectl get nodes

$ kubectl describe nodes
```

Namespaces

- **Resource Partitioning** - Divides cluster resources into logical groups.
- **Isolation** - Securely share cluster with multiple teams.

Create a namespace:

```
$ kubectl create namespace newbie-ns

$ kubectl get namespace

$ kubectl describe namespace
```



Namespace: kube-system

- **System Components** - Dedicated namespace for essential system components and infrastructure services.
- **Critical Operations** - Hosts management components (schedulers, controllers, network plugins, etc.) essential for cluster operations.
- **Isolation** - Helps isolate critical system components from user workloads, enhancing security and maintainability.

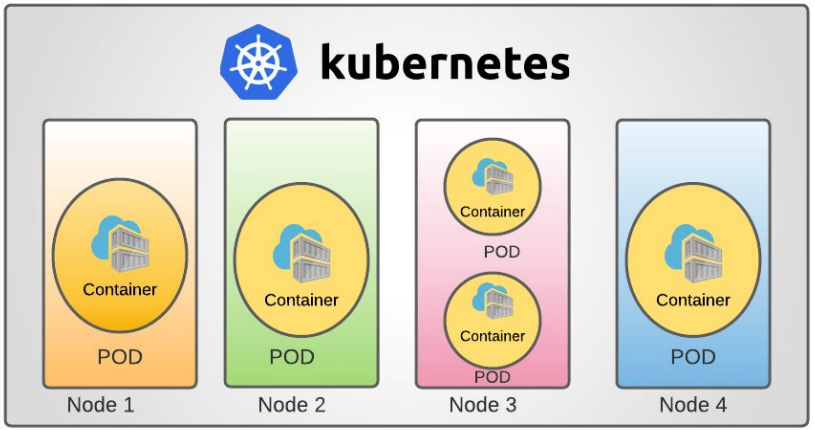
```
$ kubectl get namespace kube-system

$ kubectl describe namespace kube-system
```

Run a K8S Pod

```
# Create a Pod imperatively

$ kubectl run nginx-pod --image=nginx:latest --restart=Never

$ kubectl delete pod nginx-pod

# Create a Pod declaratively

$ kubectl apply -f k8s/nginx-pod.yaml
```

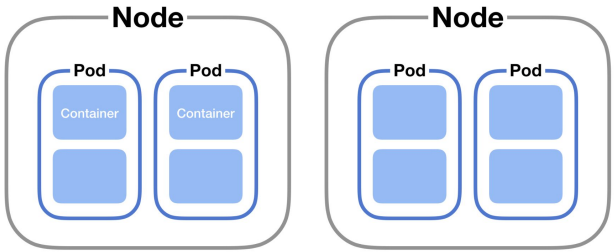
Take a Look at the Pod

```
$ kubectl get pod nginx-pod [-o wide] [-o yaml]

$ kubectl describe pod nginx-pod

$ kubectl logs nginx-pod

$ kubectl exec -it nginx-pod – /bin/bash

    $ curl inside the container

$ docker ps  # check the containers

$ kubectl port-forward nginx-pod 8080:80
```



K8S Pod

- **Basic Unit** -The smallest unit in the Kubernetes object model
- **Isolation** - Pod processes and resource allocations are segregated
  - Helps secure Pod interactions with the rest of the cluster
  - Caps resources at the Pod level to protect other workloads in the cluster
- **Shared Resources** - Containers within a Pod share the same IP and ports
  - Enables easy communication over the localhost
- **Lifespan** - Pods can have a short lifespan
  - Easily created, terminated, and replaced based on the application's requirements.



Run a K8S Deployment

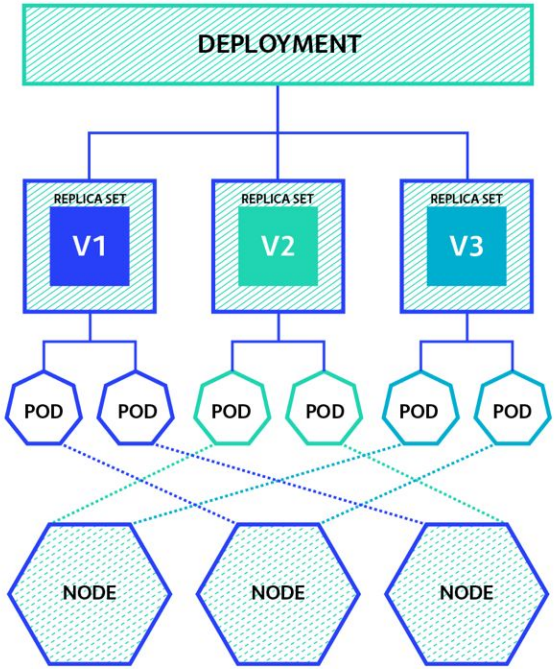
```
# Create a deployment imperatively

$ kubectl create deployment nginx-deployment --image=nginx:latest

$ kubectl scale deployment  nginx-deployment -replicas=3

$ kubectl delete deployment nginx-deployment

# Create a deployment declaratively

$ kubectl apply -f k8s/nginx-deployment.yaml
```

Take a Look at the Deployment

```
$ kubectl get deployment nginx-deployment [-o wide] [-o yaml]

$ kubectl describe deployment nginx-deployment

$ kubectl get replicaset nginx-deployment

$ kubectl get pods  # look at pods supporting this deployment

$ kubectl logs <pod-name> # use pod names discovered

$ docker ps  # check the containers
```

K8S Deployment

- **Scalability** - Scale applications up or down by adjusting replica counts.
- **Load Balancing** - Built-in load balancing for even distribution of traffic.
- **Self-Healing** - Health checks and automatic replacement of unhealthy pods for high reliability.
- **Rolling Updates** - Updates without downtime, with quick rollback options.



- … this is where K8S value starts to show.

Run a K8S Service

```
# Expose the deployment imperatively

kubectl expose deployment nginx-deployment \

  -name=nginx-service --port=80 --type=NodePort

kubectl delete service nginx-service

# Expose the deployment declaratively

kubectl apply -f k8s/nginx-service.yaml
```

Take a Look at the Service

```
$ kubectl get service nginx-service

$ kubectl describe service nginx-service

$ kubectl get endpoints nginx-service

$ NODE_IP=$(kubectl get nodes -o
jsonpath='{.items[0].status.addresses[0].address}')

$ NODE_PORT=$(kubectl get service nginx-service -o
jsonpath='{.spec.ports[0].nodePort}')

$ curl $NODE_IP:$NODE_PORT

$ kubectl logs <pod-name>
```

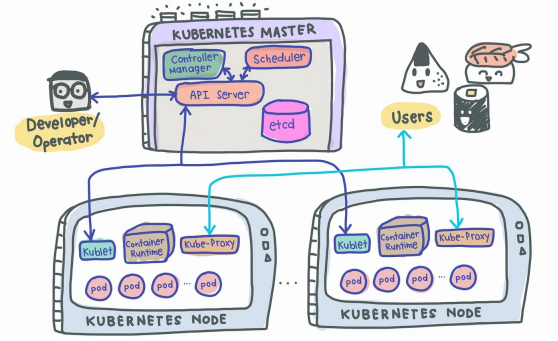
K8S Service

- Load Balancing - Efficiently distributes incoming traffic among pods.
- Service Discovery - Provides a stable endpoint for communicating with pods.
- Dynamic Management - Organizes and manages pods dynamically using labels for scalability and updates.
- Communication Flexibility - Facilitates both internal cluster communication and external requests.

Clean-up

```
$ minikube delete -p newbie-demo

$ minikube status -p newbie-demo
```

Summary


KUBERNETES MASTER
Controller Manager
Scheduler
API Server
etcd
Developer/ Operator
Users
Kublet
Container Runtime
Kube-Proxy
pod
KUBERNETES NODE
Kublet
Container Runtime
Kube-Proxy
pod
KUBERNETES NODE

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