Kubernetes Beginners Workshop

@Vendevio by Tobias Mühlberger

26.05.2021

Workshop Structure

- Intro
- Terminology
- Architecture and Concepts
- Hands-on Exercises

Materials: github.com/t-muehlberger/k8s-beginners-workshop-vendevio

Prerequisites

- Basic Docker/Container knowledge
- Installed **`kubectl`** locally
- Access to a Kubernetes cluster
  - minikube
  - microk8s
  - **Vendevio Dev Cluster**

What you will learn

- How to interact with Kubernetes using **`kubectl`** and the rancher web-ui
- How to deploy an application to Kubernetes
- How to troubleshoot problems with your application
    - Access logs of your application
    - Access ports of your application
    - Access your database
- Basic insights into what happens under the hood

What is Kubernetes

- Container orchestration platform
- for automating deployment, scaling, and management of containerized applications
- across clusters of hosts.
- “The Operating System of the Cloud”

Why Kubernetes

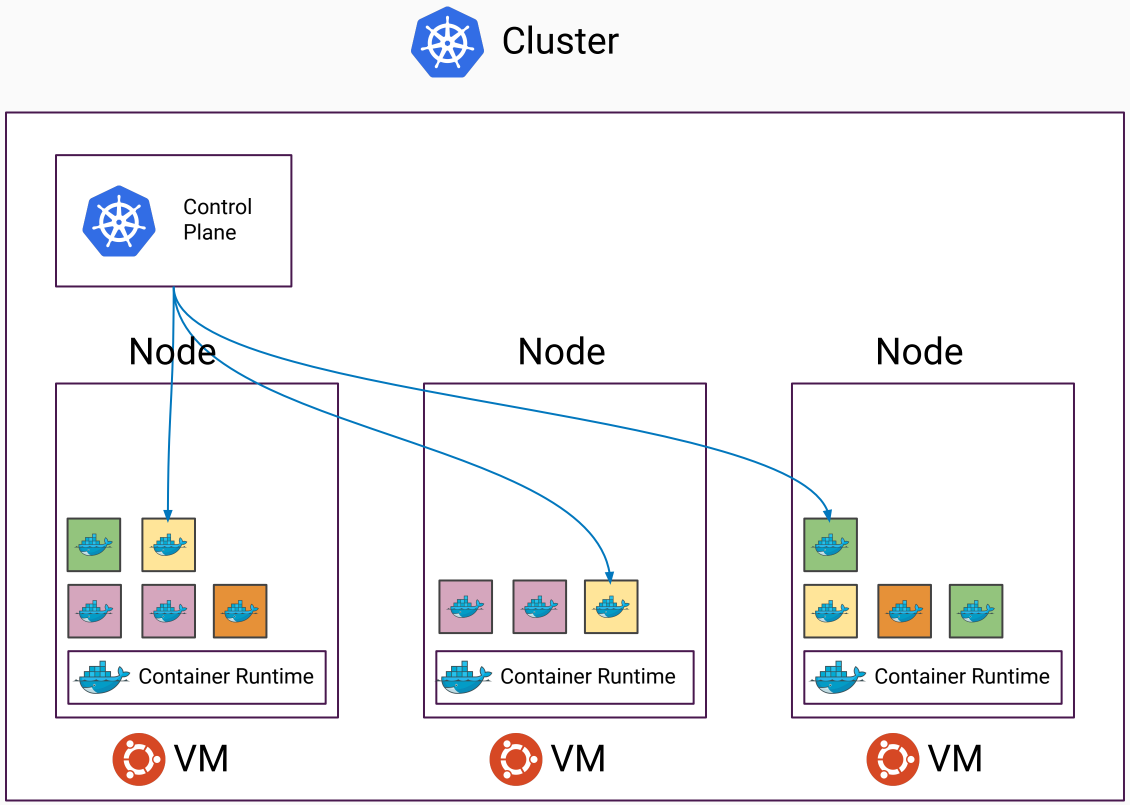
- Infrastructure as Code
- Desired state configuration
- Self-healing
- Cross-Cloud portability
- Open source ecosystem
- Layer between Devs and Ops

Kubernetes vs. Docker vs. Docker-Compose

- Declarative vs. Imperative
- Multiple Servers vs. Single Server

Terminology

- Cluster
- Node
- Pod
- Workload
- Control Plane
- Container runtime
- Kubernetes Objects
- Helm Charts



Cheat Sheet

```
kubectl get all -n <namespace>

kubectl get <kind> <name>

    kubectl get service postgres

kubectl describe <kind>/<name>

kubectl logs -f <pod-name>

kubectl apply -f config.yml

kubectl delete -f config.yml
```

Exercise 1: Namespace

- "Kubernetes namespaces help different projects, teams, or customers to share a Kubernetes cluster."
- Namespaces help organize and avoid naming conflicts
- Separate projects should be deployed in separate namespaces

```
kubectl create namespace <first_name>
```

Exercise 2: Deployment

- Describes a Workload
- Deployment Controller will create Pods
- Controls Replication
- Rolling Updates
- Use Deployments rather than creating a Pod manually

Exercise 3: Service

- Abstraction for access to Pod
- Service Discovery
  - Caller does not need to know the name or IP address of the Pod
  - A single DNS name for a set of Pods
  - Acts as internal load balancer for replicated Pods
- Type
  - **ClusterIP**
  - NodePort
  - ...

Exercise 4: Ingress

- Reverse Proxy
  - Nginx
- Alternative to Traefik in Docker-Compose Deployment
- Routing
- SSL Termination

Exercise 5: Persistence

- Required for stateful applications such as databases
- Persistent Volume Claim (PVC) is an Abstraction
- Storage Controller will create a Persistent Volume (PV) according to specification
- Different Storage Classes for different type of storage

Exercise 5.1: Postgres

- Connect to Postgres via Port-Forwarding
  - `kubectl port-forward -n <namespace> service/postgres 5432:5432`
- Access to Console
  - kubectl exec -ti -n <namespace> <postgres-pod> -- /bin/bash
- Create DB Dump
  - kubectl exec -n <namespace> <postgres-pod> -- pg_dump -U postgres > db-dump.sql
- Restore DB Dump
  - kubectl exec -ti -n <namespace> <postgres-pod> -- psql -U postgres postgres < db-dump.sql

Exercise 6: ConfigMaps + Secrets

- Enables separation between configuration and Deployment
- Config can be used as
  - Environment Variable
  - Mounted config File
- Protect Secrets

Exercise 7: (Cron) Jobs

- One time Job or recurring Job
- Should be used for:
    - DB Migrations
    - Backups

Extending Kubernetes

- Custom Resource Definition (CRD)
- Operators
- Example
    - Velero Backup CRD

Useful Ressources

https://kubernetes.io/docs/concepts/

https://kubernetes.io/docs/tutorials/

https://kubernetes.io/docs/setup/best-practices/

https://www.youtube.com/watch?v=X48VuDVv0do