KUBERNETES WORKSHOP

VORTRAGENDE

- Sandro Koll

- Pascal Rimann

TEILNEHMER

- Kurze Vorstellung
- Erfahrungen?
  - Docker
  - Kubernetes
- Erwartungen?

TAG 1

AGENDA

1. Setup
2. Container
3. Monolithen vs. Microservices
4. Container-Orchestrierung
5. Prinzipien hinter Kubernetes

SETUP

```
sudo usermod -aG docker ${USER}
git clone https://github.com/x-cellent/k8s-workshop.git
cd k8s-workshop
make
```

=> bin/w6p

BINARY

```
Usage:
  w6p [flags]
  w6p [command]

Available Commands:
  cluster     Runs the workshop cluster or exercises
  exercise    Runs the given exercise
  help        Help about any command
  install     Installs tools on local machine
  slides      Shows or exports workshop slides

Flags:
  -h, --help   help for w6p
```

CONTAINER

Software wird schon seit Jahrzehnten in Archive oder Single-Binaries verpackt

- Einfache Auslieferung
- Einfache Verteilung

ABER

- Installation notwendig
- Dependency Hell
- No cross platform functionality

LÖSUNG

- Verpacken der Software *mitsamt* aller Dependencies (Image)
  - Nichts darüber hinaus (Betriebssytem notwendig?)
- Container-Runtime für alle Plattformen

UMSETZUNG

- Linux
- Idee: Container teilen sich Kernel
- LXC; basierend auf Kernel-Funktionalitäten
  - namespaces
  - cgroups
- Docker erweitert LXC um
  - CLI zum Starten und Verwalten von Containern

VORTEILE CONTAINER

1. Geringere Größe
2. Erhöhte Sicherheit
3. Funktional auf allen Systemen
4. Immutable
    - Damit Baukastenprinzip möglich (DRY)

VORTEILE GEGENÜBER VMS

1. Geringere Größe
2. Geringerer Ressourcenverbrauch
3. Viel schnellere Startup-Zeiten
4. Auch geeignet für Entwicklung und Test

NACHTEILE GEGENÜBER VMS

1. Geringere Sicherheit
2. Keine echte Trennung
    - z.B. kein Block-Storage möglich

Container und VMs schließen sich aber nicht gegenseitig aus

DOCKER KOMPONENTEN

1. Image
   - Layer
   - Dockerfile
2. Container
3. Image Registry

DOCKERFILE

- Image-*Rezept* mit u.a. folgenden Instruktionen:
  - FROM
  - COPY/ADD
  - RUN
  - USER
  - WORKDIR
  - ARG/ENV

BEISPIEL

```
# Basis-Image
FROM alpine:3.15

# Installiert busybox-extras ins Basis-Image
RUN apk add --no-cache busybox-extras
# ...und committed den FS-Diff als neuen Layer


# Installiert auf dem obigen Layer mysql-client
RUN apk add --no-cache mysql-client
# ...und commited das FS-Diff als neuen Layer

# Default command
ENTRYPOINT ["mysql"]

# Default arg(s)
CMD ["--help"]
```

EINIGE DOCKER COMMANDS

- docker build
  - Baut ein Image von Dockerfile
- docker images / docker image ls
  - Listet alle (lokalen) Images
- docker tag
  - Erstellt Image "Kopie" unter anderem Namen
- docker rmi / docker image rm

- docker run
  - Startet ein Image -> Container
- docker ps [-q]
  - Listet alle (laufenden) Container
- docker rm
  - Löscht einen Container
- docker logs

- docker [image] inspect
  - Zeigt Metadaten von Container/Images
- docker cp
  - Kopiert eine Datei aus Container ins Host-FS und umgekehrt
- docker save/load
  - Erzeugt Tarball aus Image und umgekehrt

IMAGE BAUEN

```
docker build -t [REPOSITORY_HOST/]IMAGENAME:IMAGETAG \
    [-f path/to/Dockerfile] path/to/context-dir
```

- Kontext-Verzeichnis wird zum Docker Daemon hochgeladen
  - lokal oder remote (via DOCKER_HOST)
  - Nur darin enthaltene Dateien können im Dockerfile verwendet werden (COPY/ADD)
  - Nach Möglichkeit keine ungenutzten Dateien hochladen

AUFGABE 1

```
bin/w6p exercise docker -n 1
```

Zeit: ca. 5 min

CONTAINER STARTEN

```
docker run [--name NAME] [-i] [-t] [-d|--rm] [--net host|NETWO
    [-p HOST_PORT:CONTAINER_PORT] [-u UID:GID] IMAGE [arg(s)]
```

- Noch viel mehr Flags möglich
- Referenz

COMMAND IN CONTAINER TRIGGERN

```
docker exec [-i] [-t] CONTAINER COMMAND
```

Via Shell in den Container "springen":


```
docker exec [-i] [-t] CONTAINER COMMAND
```

DATEIEN KOPIEREN

...vom Host in den Container:

```
docker cp HOST_FILE CONTAINER_NAME:CONTAINER_FILE
```

...vom Container in das Host-FS:

```
docker cp CONTAINER_NAME:CONTAINER_FILE HOST_FILE
```

AUFGABE 2

```
bin/w6p exercise docker -n2
```

Zeit: ca. 20 min

METADATEN

```
docker inspect IMAGE|CONTAINER
```

- Image Metadaten
  - ID
  - Architecture
  - Layers
  - Env
  - ...

- Container Metadaten
  - ID
  - Image ID
  - NetworkSettings
  - Mounts
  - State
  -

AUFGABE 3

```
bin/w6p exercise docker -n3
```

Zeit: ca. 15 min

LINTING

- hadolint
- Erhältlich als Docker Image:

```
docker run ... hadolint/hadolint hadolint path/to/Dockerfile
```

AUFGABE 4

```
bin/w6p exercise docker -n4
```

Zeit: ca. 5 min

MULTI-STAGE DOCKERFILE

```
# Build as usual in the first stage
FROM golang:1.17 AS builder # named stage

WORKDIR /work
# Copy source code into image
COPY app.go .

# Compile source(s)
RUN go build -o bin/my-app

# Further builder images possible, e.g.
# FROM nginx AS webserver
# ...

# Final stage: all prior images will be discarded after build
FROM scratch

# ...but here we can copy files from builder image(s)
COPY --from=builder /work/bin/my-app /

ENTRYPOINT ["/my-app"]
```

VORTEILE

1. Kompakte Imagegröße
2. Erhöhte Sicherheit

AUFGABE 5

```
bin/w6p exercise docker -n5
```

Zeit: ca. 15 min

DOCKER REGISTRY

- Docker-Hub
  - öffentlich
- private Registries möglich
  - Image registry
  - absicherbar
  - praktisch in jeder Firma eingesetzt

AUFGABE 6

```
bin/w6p exercise docker -n6
```

Zeit: ca. 15 min

WAS DOCKER NICHT BIETET:

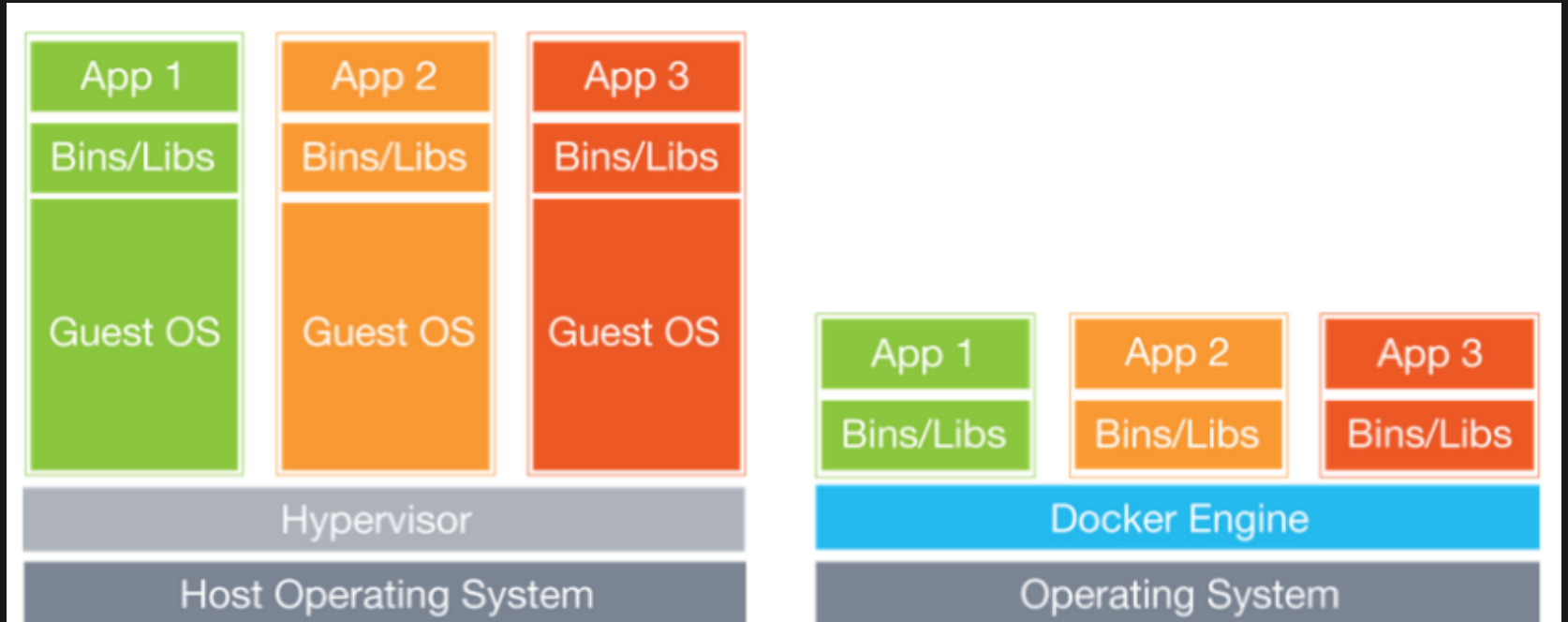
1. Orchestrierung
2. Ausfallsicherheit

=> Kubernetes - bietet beides - ...und noch viel mehr

DOCKER COMPOSE

- Für Multi-Container Docker Anwendungen
- docker-compose.yaml
  - Definition der Container
- docker-compose up/down
  - Start/Stop aller Anwendungen in einem Rutsch
- Rudimentäre Funktionalitäten
- Geeignet für sehr kleine (Dev-/Test-)Umgebungen

MONOLITH VS MICROSERVICES


App 1
Bins/Libs
Guest OS
App 2
Bins/Libs
Guest OS
App 3
Bins/Libs
Guest OS
Hypervisor
Host Operating System
App 1
Bins/Libs
App 2
Bins/Libs
App 3
Bins/Libs
Docker Engine
Operating System

CONTAINER ORCHESTRIERUNG

WIESO?

- Orchestrierung von Containern

WARUM KUBERNETES?

- Warum nicht Docker Swarm?
- Mehr Flexibilität
- Eingebautes Monitoring und Logging
- Bereitstellung von Storage
- Größere userbase

PRINZIPIEN HINTER KUBERNETES

DER POD

- kein Container
- beinhaltet mindestens einen Container
- kann init container beinhalten
- kann sidecar container beinhalten
- kleinste Einheit in Kubernetes

WO LAUFEN PODS?

- Auf (Worker-)Nodes

ORDNUNGSELEMENTE

- ReplicaSet
- Deployment
- StatefulSet
- DaemonSet
- Job
- CronJob

FRAGEN

- Hab ihr noch Fragen an uns?

AUSBLICK TAG 2

- Architektur von Kubernetes
- Basis Objekte von Kubernetes

TAG 2

AGENDA

1. Architektur von Kubernetes
2. Einrichtung euerer Umgebung
3. Basisobjekte Kubernetes mit Übungen

KUBERNETES

Kubernetes ist ein Open-Source-System zur Automatisierung der Bereitstellung, Skalierung und Verwaltung von Container-Anwendungen

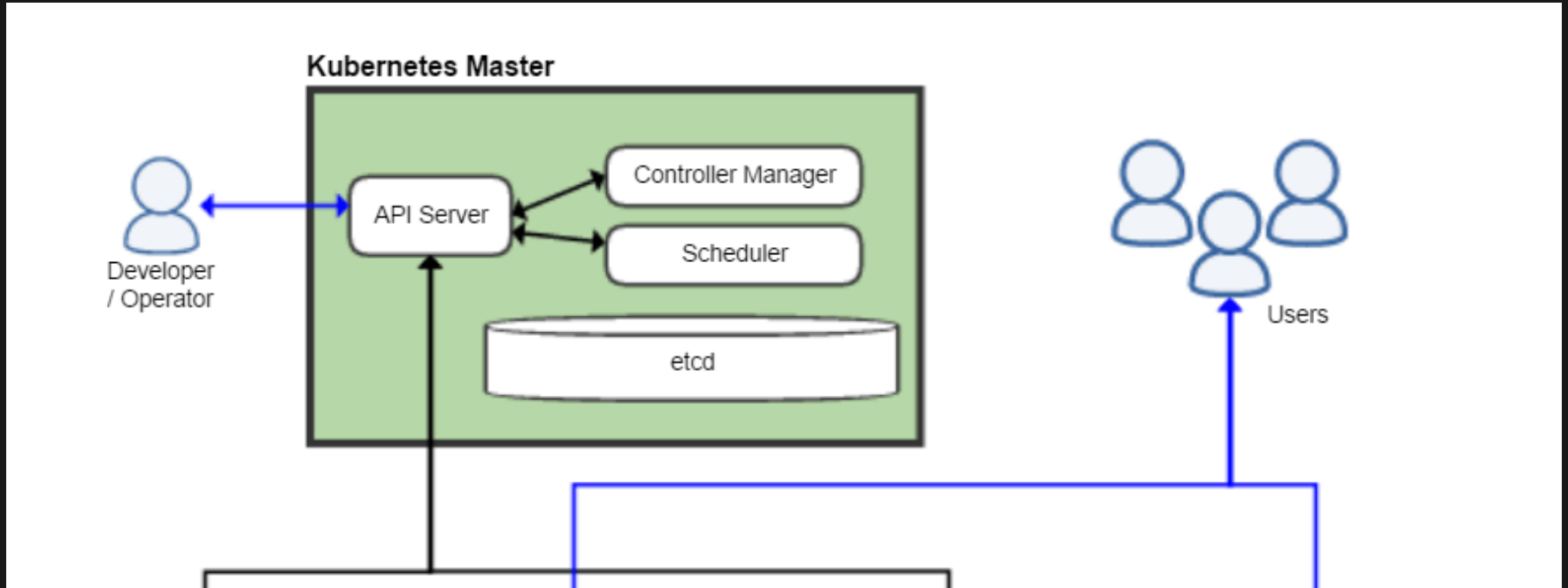
KUBERNETES

- Ursprünglich 2014 entwickelt von Google
- Abgegeben 2015 an die Cloud Native Compute Fondation (CNCF)

CNCF

- Cloud Native Computing Foundation
- 2015 Gegründet
- äber 500 Hersteller und Betreiber

ARCHITEKTUR VON KUBERNETES



ARCHITEKTUR DES CLUSTERS

1. Modular und austauschbar
    1. Control-Plane
        - etcd
        - API-Server
        - Scheduler
        - Kube-Controller-Manager
    2. Nodes

CONTROL-PLANE

ETCD

- entwickelt von CoreOS
- key-value Database
- kann nicht getauscht werden
- speichert stand von cluster
- Consistency notwending

API-SERVER

- Ansprechpunkt des Users
- Validation der Daten
- bekanntester ist kube-apiserver
- horizontale skalierbarkeit

SCHEDULER

- Verteilt workload
- verantworlich für pods ohne node

KUBE-CONTROLLER-MANAGER

- bringt cluster von "ist" -> "soll"
- Managed Nodes
- mitteilung an scheuduler wenn node down

NODES

KUBELET

- verwaltet pods
- auf jeden node installiert
- verantwortlich für status

KUBE PROXY

- verwaltet Netzwerkanfragen
- routet traffic zu gewünschten pod
- loadbalancer

WEITERE KOMPONENTEN

- CNI
- Container-Runtime

OPENSOURCE

NAMESPACES

- separierungseinheit in Kubernetes

AUSFALLSICHERHEIT

1. Container Health Check
    1. readyness
    2. liveness
2. Hostsystemausfall
3. Update

FRAGEN

- Hab ihr noch Fragen an uns?

WICHTIGE RESSOURCEN

1. kubectl cheat sheet:
   https://kubernetes.io/docs/reference/kubectl/cheatsheet
2. kubernetes docs:
   https://kubernetes.io/docs/concepts/overview

SETUP EUERER UMGEBUNG

- clonen dieses Repos und erstellen des kommandozeilen-tools

```
git clone https://github.com/x-cellent/k8s-workshop.git
cd k8s-workshop
make
```

INSTALL TOOLS

Kubernetes Dokumentation:

- kubectl
- krew
- helm
- kind
- k9s

INSTALL KUBECTL PLUGINS

- node-shell

OBJEKTTYPEN IN K8S

+++

POD

- umfasst einen oder meherere Container
- niedrigstes verwaltbares Objekt
- jeder Pod bekommt IP addresse

AUFGABE 1

```
bin/w6s exercise k8s -n1
```

Zeit: ca. 5m

LÖSUNGSBESPRECHUNG

SERVICE

- Objekt um Pod im Netzwerk erreichbar zu machen
- Loadbalancing
- Dynamische IP's von Pods

Services werden genutzt um pods im Netzwerk

erreichbar zu machen

hat eine loadbalancing funktion, wenn mehere pods mit gleichem Label im Namespace sind wird die last aufgeteilt

geht an die pods mit einem Label, daher sind dynamische IPs bei Pods keine Probleme

AUFGABE 2

```
bin/w6s exercise k8s -n2
```

Zeit: ca. 5m

LÖSUNGSBESPRECHUNG

REPLICASETS

- Pods Replizieren
- Nachträglich nicht änderbar

AUFGABE 3

```
bin/w6s exercise k8s -n3
```

Zeit: ca. 5m

LÖSUNGSBESPRECHUNG

DEPLOYMENT

- Bessere art ReplicaSets zu verwalten
- Updates
- am weitesten verbreitete art

DAEMONSET

- jede Node bekommt ein Replica
- enorm ausfallsicher
- logs
- monitoring

AUFGABE 4

```
bin/w6s exercise k8s -n4
```

Zeit: ca. 5m

LÖSUNGSBESPRECHUNG

- deamonSet aus kubernetes Doku
- kubernetes Doku ist immer gut

STATEFULSET

- persistente Pods
- geordnetes Updaten

JOB

- ausführung eines commandes in einem pod
- datenbank backups

AUFGABE 5

```
bin/w6s exercise k8s -n5
```

Zeit: ca. 5m

LÖSUNGSBESCHREIBUNG

CRONJOBS

- Mischung aus klassischen Cronjobs und Jobs
- regelmäßige ausführung eines jobs

AUFGABE 6

```
bin/w6s exercise k8s -n6
```

Zeit: ca. 5m

LÖSUNGSBESPRECHUNG

CONFIGMAPS

- speicherung von nicht vertraulichen daten
- einbindung in pods als
  - enviroment-variable
  - command-line argument
  - als datei in Volume
- kein reload von pods bei änderung von configmap

AUFGABE 7

```
bin/w6s exercise k8s -n7
```

Zeit: ca. 5m

LÖSUNGSBESPRECHUNG

AUFGABE 8

```
bin/w6s exercise k8s -n8
```

Zeit: ca. 5m

LÖSUNGSBESPRECHUNG

SECRET

- speicherung vertraulicher daten
- unentschlüsselt in etcd db

FRAGEN

- Hab ihr noch Fragen an uns?