

# OpenShift Einsteiger Training

Tobias Derksen

---



# Über mich ...



Tobias Derksen

- DevOps Consultant @codecentric
- RedHat Partner
- OpenShift Trainer
- RedHat Certified Engineer



# Vorstellung

# Agenda

- Einführung in OpenShift
- Unterschiede OpenShift <-> kubernetes
- Web Console & CLI Basics
- Routing
- ImageStreams & Builds
- Logging & Metriken
- Security
- Best Practices

# Einführung in OpenShift

---

Was ein Chaos ...



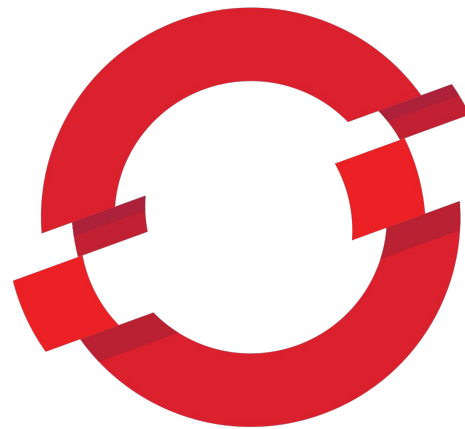
redhat.



kubernetes

OPENSIFT

origin



cri-o



docker

OPENSIFT

# OpenShift <-> kubernetes

# Warum gibt es OpenShift?

- OpenShift ist ein “*opinionated kubernetes*” (RedHat)
- Das bedeutet:
  - Fertig konfiguriert
  - Vorinstallierte Komponenten
  - Ready-to-use Lösungen für häufige Probleme
- Zusätzlich:
  - Hilfstools für Entwickler (CodeReady Workspaces, odo)
  - Lokaler Entwicklungcluster (CodeReady Containers)



# OpenShift ist ... kubernetes plus

- Routing
- Metriken
- Logging
- Web Oberfläche
- Builds
- Image Registry
- Sicherheitsmaßnahmen
- Templates
- Und vieles mehr ...

Mit Red Hat Subscription:

- Trusted Registry
- Security Newsletter
- **Enterprise Support**

# Historische Ressourcen

- DeploymentConfig
  - Funktional Identisch mit Deployments
- ReplicationController
  - Funktional identisch mit ReplicaSet
- Project
  - Synonym für Namespace

# Web Console Basics

---

# Web Console aufrufen

- <https://console-openshift-console.apps.bfarm.cc-openshift.de/>

User: user#

Passwort: user#

# entspricht der zugewiesenen Nummer

# OpenShift CLI Basics

---

# oc - Command Line Interface

- Kann alles was kubectl kann
- Bietet zusätzliche Features
- Bietet Zugriff auf OpenShift spezifische Workflows
  - Anlegen von Projekten
  - Anlegen von Deployments
  - Anlegen von Routen

# Routing

---

# Routing

- OpenShift Konzept: Route
- kubernetes Konzept: Ingress
- Route verbindet HTTP Hostname mit Service
- Automatisch generierte Hostnames über Wildcard DNS  
*<route-name>-<namespace>.apps.<cluster>.<domain>*
- Default TLS Zertifikat verfügbar
- Man kann auch ein TLS Zertifikat pro Route angeben



# TLS Einstellungen

- Termination Type
  - Edge
  - Passthrough
  - Reencrypt
- Insecure Traffic Policy
  - Redirect
  - Allow
  - None

# ImageStream & Builds

# ImageStreams

- Representiert Images in der lokalen Registry
- Kann auch auf externe Images verweisen
- Scheduled imports - prüft regelmäßig auf neue Images
- ReferencePolicy
  - Source (einfacher Verweis auf ein externes Image)
  - Local (externes Images wird lokal gespeichert)

# Builds

- Bietet Images Builds innerhalb des Clusters
- Verschiedene Strategien verfügbar
- Kann in interne oder externe Registries pushen
- Automatische Build Trigger

# Build Strategy

- Source-to-Image (S2I)
- Dockerfile
- Custom Strategy
- JenkinsPipeline (deprecated)

# Logging & Metriken

---

# Metriken

- Zugriff auf Metriken über Developer Console
- Vordefinierte Queries und Dashboard
- Eigene PromQL Queries (aber keine eigenen Dashboards)
- Cluster sammelt automatisch Pod Metriken
  - CPU Auslastung
  - Memory Auslastung
  - Netzwerk & Filesystem Durchsatz

# Logging

- Optionale Komponente muss nachträglich installiert werden
- Cluster sammelt Ausgabe (stdout & stderr) aller Container
- Log Einträge werden in Elasticsearch gespeichert und indiziert
- Man kann die Einträge über Kibana abrufen
- Die Log Einträge sind nur eine bestimmte Zeit verfügbar (Retention Time)



# Security

---

# Übersicht

- Role based access control (RBAC)
- Security Context Constraints (SCC)
- ~~PodSecurityPolicy (PSP)~~

# Rollen & Rechte

- Cluster Rollen
- Projekt Rollen
- Rechte bestehen aus Verb + Objekttype (Beispiel: get projects)
- Rechte eines Accounts = Summe aller erlaubten Aktionen
- Serviceaccounts

## Cluster Rollen:

- cluster-admin
- cluster-reader
- self-provisioner

## Projekt Rollen:

- admin
- edit
- view

# Serviceaccounts (SA)

- Sind praktisch technische User des Clusters
- Können genauso Rechte bekommen wie normale User
- Jeder Pod läuft mit den Rechten eines ServiceAccounts
- In jedem Projekt werden SA automatisch angelegt:
  - default
  - builder
  - deployer

# Serviceaccounts (SA)

- Ordner: **/var/run/secrets/kubernetes.io/serviceaccount**
- Beinhaltet:
  - Cluster Root Zertifikat
  - Cluster Service Zertifikat
  - Access Token für ServiceAccount
  - Aktueller Namespace
- API URL: **https://kubernetes.default.svc**

# Security Context Constraints (SCC)

- Kontrolliert die Rechte die ein Pod anfordern kann
- Ohne SCC werden erweiterte Rechte vom Scheduler zurückgewiesen
- Erlaubt Pods:
  - Zugriff auf Host Dateisystem
  - Zugriff auf Host Netzwerk
  - Starten als spezifischer User (z.B. root)
  - Erweiterte Möglichkeiten mit Gruppen

## Was man **NIEMALS** tun sollte ...

- Rechte an den default Service Account geben
- SCC an den default Service Account geben
- “privileged” SCC vergeben
- *Container als root laufen lassen weil man zu faul ist es richtig zu machen*

*oc adm policy add-scc-to-user privileged -z default*

# Persistent Storage

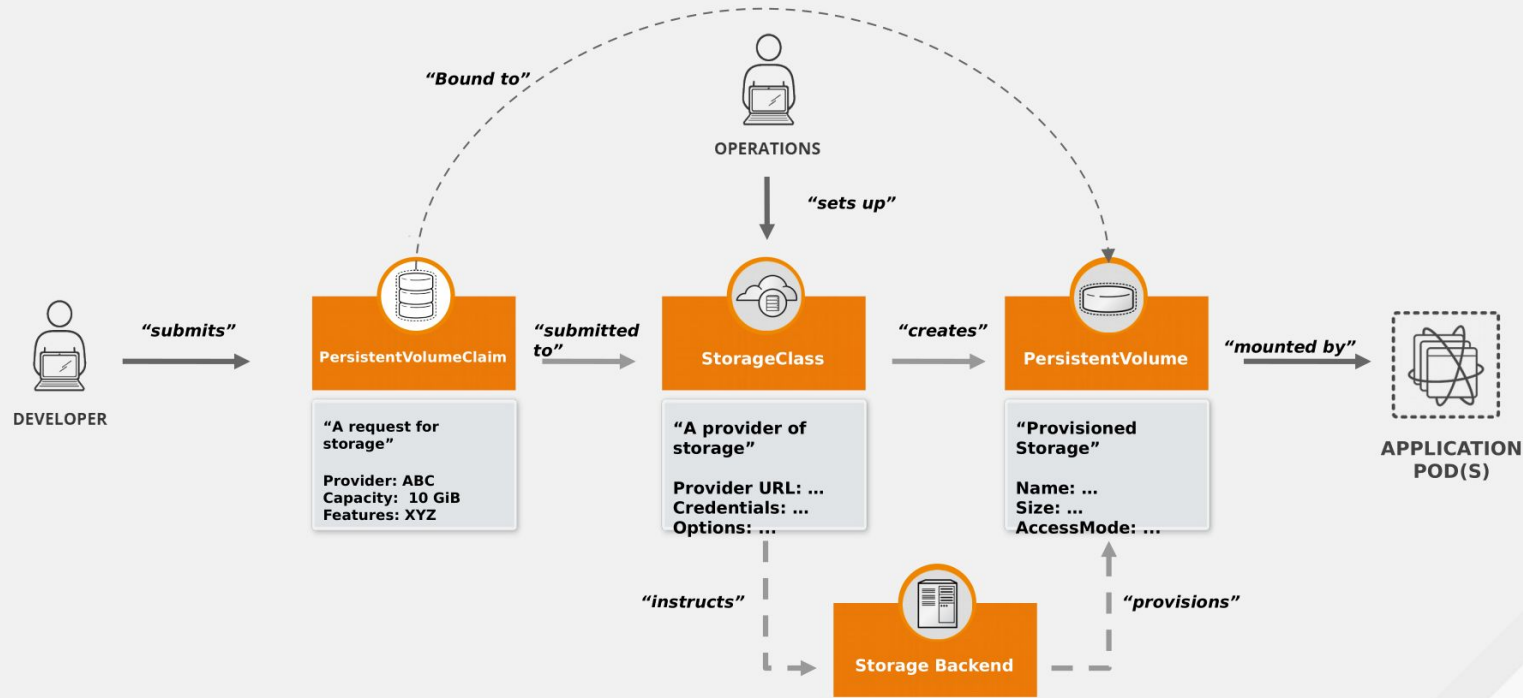
---



# Access Modes

- Read Only (ROX)
- Read Write Once (RWO)
- Read Write Many (RWX)

# OPENSHIFT PERSISTENT STORAGE FRAMEWORK



# Best Practices

---

# Best Practices

- Nicht alle Applikationen eignen sich dafür
  - Monolithen -> schlechte Skalierung
  - Datenbanken -> von schneller Storage abhängig
  - Nicht HTTP basierter Traffic
- Trennen von Development und Production
- Wiederverwendbare Images -> externe Konfiguration (ConfigMap, Env)
- Immer sinnvolle Ressourcen Anforderungen einstellen
- Immer sinnvolle Health Checks konfigurieren

# Best Practices - Security

- Secrets und ConfigMaps sinnvoll trennen
- non-root Container
- Container Scanning nach Sicherheitslücken
- Nichts aus offenen Registries laden (z.B. Docker Hub)
- Traffic Encryption
- **Regelmäßige Updates der Base Images**

# Ende

---

# Upcoming Events

- Zurzeit leider digital ...
- OpenShift Anwender Treffen ([openshift-anwender.de](https://openshift-anwender.de))  
**30. September 2020**
- OpenShift Slack DE ([openshift-de.slack.com](https://openshift-de.slack.com))

# OpenShift Ready Applications



# The cluster is your friend ... but it needs your help

- Cluster ensures a certain state  
*Tell the cluster the state you desire; how it gets there is not your problem.*
- Cluster needs to know what resources you need
- Cluster needs information about the state of the application to ensure it has the desired state

# Health Checks

- **Liveness Probe**

*Checks whether the container is alive*

If fail, container is restarted

- HTTP GET
- Shell command
- Open TCP ports

- **Readiness Probe**

*Checks whether the container is able to accept traffic*

If fail, container will not get any traffic from service layer

# Resource Allocation

- Resources are valuable and expensive
- Know what you need ... and tell the cluster
- Requests are guaranteed ... limits are not
- *If there is no node with enough resources, the pod will not start at all*

```
apiVersion: v1
kind: "Pod"
metadata:
  name: "test"
  labels:
    app: test
spec:
  containers:
    - image: mysql
      resources:
        requests:
          cpu: 500m
          memory: 1Gi
        limits:
          cpu: 2
          memory: 2Gi
```

# Failing is a totally valid option

- Expect that any pod is killed by kubernetes at any time
- Allow your container to fail ... as early as possible

Reasons why a pod is killed:

- Manual interaction (Admin, Developer, etc)
- Node failure or maintenance
- Network issues
- Pod / Container out-of-memory
- **Node out-of-memory**

## Noteworthy points

- Make your application **timezone aware**  
*by default container run in UTC*
- Avoid file system writes; keep it to “/tmp”
- Log messages to stdout and stderr
- Reduce dependencies; especially hard dependencies

# Stay connected



Adresse  
codecentric AG  
Am Mittelhafen 14  
48155 Münster



Contact Info  
E-Mail: [tobias.derksen@codecentric.de](mailto:tobias.derksen@codecentric.de)  
[www.codecentric.de](http://www.codecentric.de)



Telephone  
Telefon: +49 (0) 170 2295 733

Hello, World!











# CodeReady Workspaces

# HA for Applications



---



## cc\_primary template colours (included in master template)

	#FFFFFF		#15584C
	#000000		#1FB18A
	#F0F6F4		#2CE6AF
	#004452		
	#007891		
	#00AED2		
	#03BDEC		

Link colour

	#D6B32C
	#9C954E

## cc\_secondary template colours (you need to build by yourself)

	#EF5E1B
	#D6B32C
	#E61B77

# cc\_icons

