Grundlagen und Bestandteile von OpenShift

Die Basiselemente von OpenShift Applikationen sind Docker Container. Mit Docker Container können Prozesse auf einem Linuxsystem so isoliert werden, dass sie nur mit den definierten Ressourcen interagieren können. So können viele unterschiedliche Container auf dem gleichen System laufen, ohne dass sie einander “sehen” (Files, Prozesse, Netzwerk). Typischerweise beinhaltet ein Container einen einzelnen Service (Webserver, Datenbank, Mailservice, Cache). Innerhalb eines Docker Containers können beliebige Prozesse ausgeführt werden.

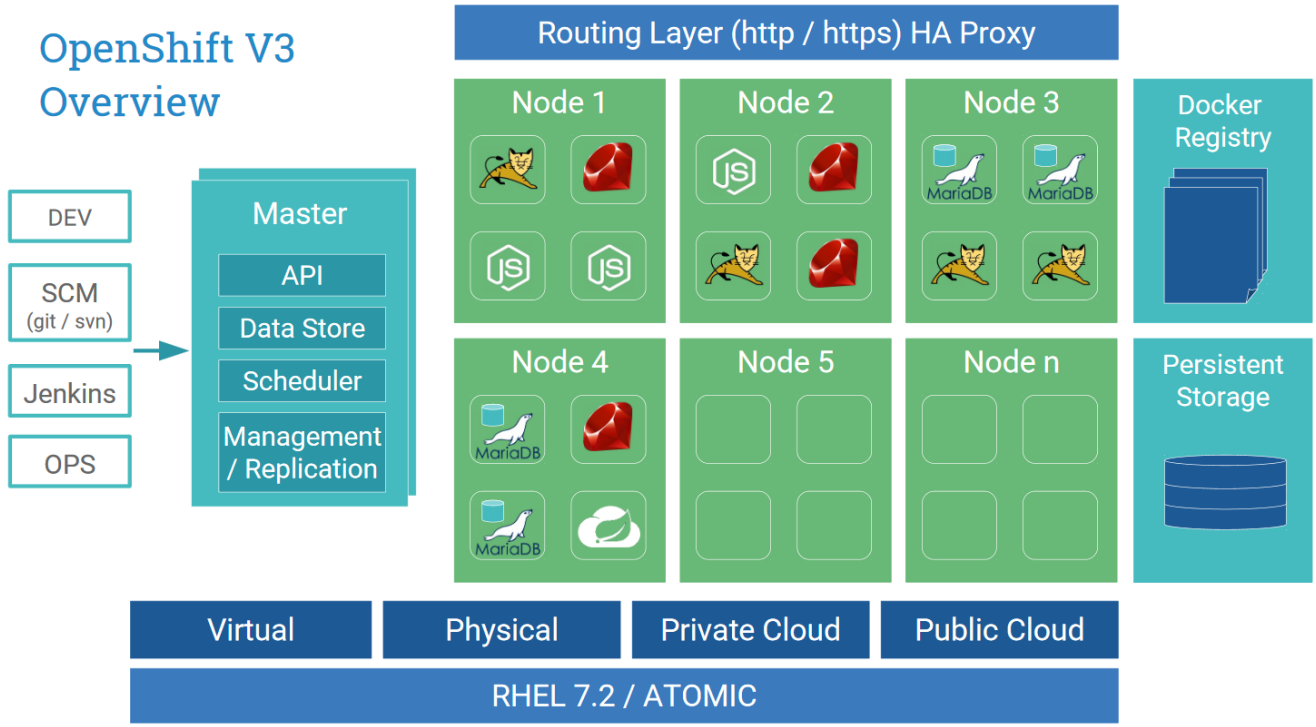
Docker Container basieren auf Docker Images. Ein Docker Image ist eine binary Datei, die alle nötigen Komponenten beinhaltet, damit ein einzelner Container ausgeführt werden kann.

Docker Images werden anhand von DockerFiles (textueller Beschrieb wie das Docker Image Schritt für Schritt aufgebaut ist) gebuildet. Grundsätzlich sind Docker Images hierarchisch angewendete Filesystem Snapshots.

Beispiel Tomcat

- Basis Image (CentOs 7)
  - Install Java
  - Install Tomcat
  - Install App

Die gebuildeten Docker Images werden in der OpenShift internen Docker Registry versioniert abgelegt und stehen der Plattform nach dem Build zum Deployment auf den Nodes zur Verfügung.



Weitere Details zur Architektur: Hier

Projekte

In OpenShift V3 werden Ressourcen (*Container, Docker Images, Pods, Services, Routen, Konfiguration, Quotas und Limiten etc.*) in Projekten strukturiert. Aus technischer Sicht entspricht ein Projekt einem Kubernetes Namespace und erweitert diesen um gewisse Konzepte.

Innerhalb eines Projekts können berechtigte User ihre Ressourcen selber verwalten und organisieren.

Die Ressourcen innerhalb eines Projektes sind über ein transparentes SDN verbunden. So können die einzelnen Komponenten eines Projektes in einem Multi-Node Setup auf verschiedene Nodes deployed werden. Dabei sind sie über das SDN untereinander sicht- und zugreifbar.

Pods

OpenShift übernimmt das Konzept der Pods von Kubernetes.

Ein Pod ist ein oder mehrere Container, die zusammen auf den gleichen Host deployed werden. Ein Pod ist die kleinste zu deployende Einheit auf OpenShift.

Ein Pod ist innerhalb eines OpenShift Projektes über den entsprechenden Service verfügbar.

Services

Ein Service repräsentiert einen internen Loadbalancer auf die dahinterliegenden Pods (Replicas vom gleichen Typ). Der Service dient als Proxy zu den Pods und leitet Anfragen an diese weiter. So können Pods willkürlich einem Service hinzugefügt und entfernt werden, während der Service verfügbar bleibt.

Einem Service ist innerhalb eines Projektes eine IP und ein Port zugewiesen und verteilt Requests entsprechend auf die Pod Replicas.

Routen

Mit einer Route definiert man in OpenShift, wie ein Service von ausserhalb von OpenShift von externen Clients erreicht werden kann.

Diese Routen werden im integrierten Routing Layer eingetragen und erlauben dann der Plattform über ein Hostname-Mapping die Requests an den entsprechenden Service weiterzuleiten.

Sind mehr als ein Pod für einen Service deployt, verteilt der Routing Layer die Requests auf die deployten Pods

Aktuell werden folgende Protokolle unterstützt:

- HTTP
- HTTPS (SNI)
- WebSockets
- TLS mit SNI

Templates

Ein Template beschreibt textuell eine Liste von Ressourcen, die auf OpenShift ausgeführt und entsprechend in OpenShift erstellt werden können.

So hat man die Möglichkeit ganze Infrastrukturen zu beschreiben:

- Java Applikation Service (3 Replicas, rolling Upgrade)
- Datenbank Service
- über Route https://java.app.appuio-beta.ch im Internet verfügbar

Basic OC-Commands

User verwalten

- OpenShift Cluster User anzeigen:

```
[michael@osec01 ~]$ oc get user
NAME      UID                                    FULL NAME   IDENTITIES
michael   5674baec-4b31-11e9-b5a4-000c29054c0c
htpasswd_auth:michael
```

- OpenShift Cluster User anlegen:

```
[michael@osec01 ~]$ oc get user
NAME      UID                                    FULL NAME   IDENTITIES
michael   5674baec-4b31-11e9-b5a4-000c29054c0c
htpasswd_auth:michael
```

- OpenShift Cluster User anzeigen:

```
[michael@osec01 ~]$ oc get user
NAME      UID                                    FULL NAME   IDENTITIES
michael   5674baec-4b31-11e9-b5a4-000c29054c0c
htpasswd_auth:michael
```

OpenShift Lab für einen schnellen Einstieg

Die nachfolgenden Labs, setzen eine bereis vorhandene Openshift Container Plattform voraus.

Falls aktuell noch keine zur Verfügung steht, kann **HIER** nachgelesen werden, wie privat ein kleines Test-Cluster eingerichtet werden kann. In den Labs werden anschliessend Grundkompetenzen zum täglichen Einsatz sowie der Betreuung von OpenShift erarbeitet.

OpenShift Labs:

1. OpenShift Client CLI Installieren
2. Erste Schritte auf der Lab Plattform
3. Ein Docker Image deployen
4. Routen erstellen
5. Skalieren
6. Troubleshooting
7. Datenbank deployen und anbinden
8. Code Änderungen via Webhook direkt integrieren
9. Persistent Storage anbinden und verwenden für Datenbank
10. Applikationstemplates
11. Eigene Templates erstellen

- OpenShift_Techlab_Einfuehrung_1.0.pdf
  - Appuio LAB OpenShift Grundlagen

Last update: **2019/04/04 10:59**