Goldschmiede

anderScore
trust in competence

Eine Tour durch die Kubernetes Welt

1.9.2017 J.Lühr

Inhalt

1. Motivation

- Microservice-Deployment
  - Paketierung .deb / .rpm?
  - Laufzeitumgebung Blech / VM / Container?
  - Wie automatisieren?
  - Skalierbarkeit?
  - Schnittstellen im Team
    - Externe Dienstleister
    - Betrieb ./. Entwicklung

1. Motivation - Ideen

- Paketierung / Laufzeug-Umgebung
  - Docker
- Build
  - Maven
- Automatisierung
  - Jenkins
  - Ansible
- Repository
  - Nexus
- Laufzeitumgebung für die Container
  - Kubernetes

1. Motivation – Warum Kubernetes?

- Paketierung als Docker-Image
  - Unabhängig von:
    - Ziel-Platform
    - Vendor
    - Service-Technologie (Java, Ruby, Python)
  - Self-Contained: Keine Abhängigkeiten
  - Umfangreicher Tool-Support (z.B. Cloud)
  - Einfaches vorgehen
- Betrieb der Docker-Container?
  - **Kubernetes**

Eine kurze Demo

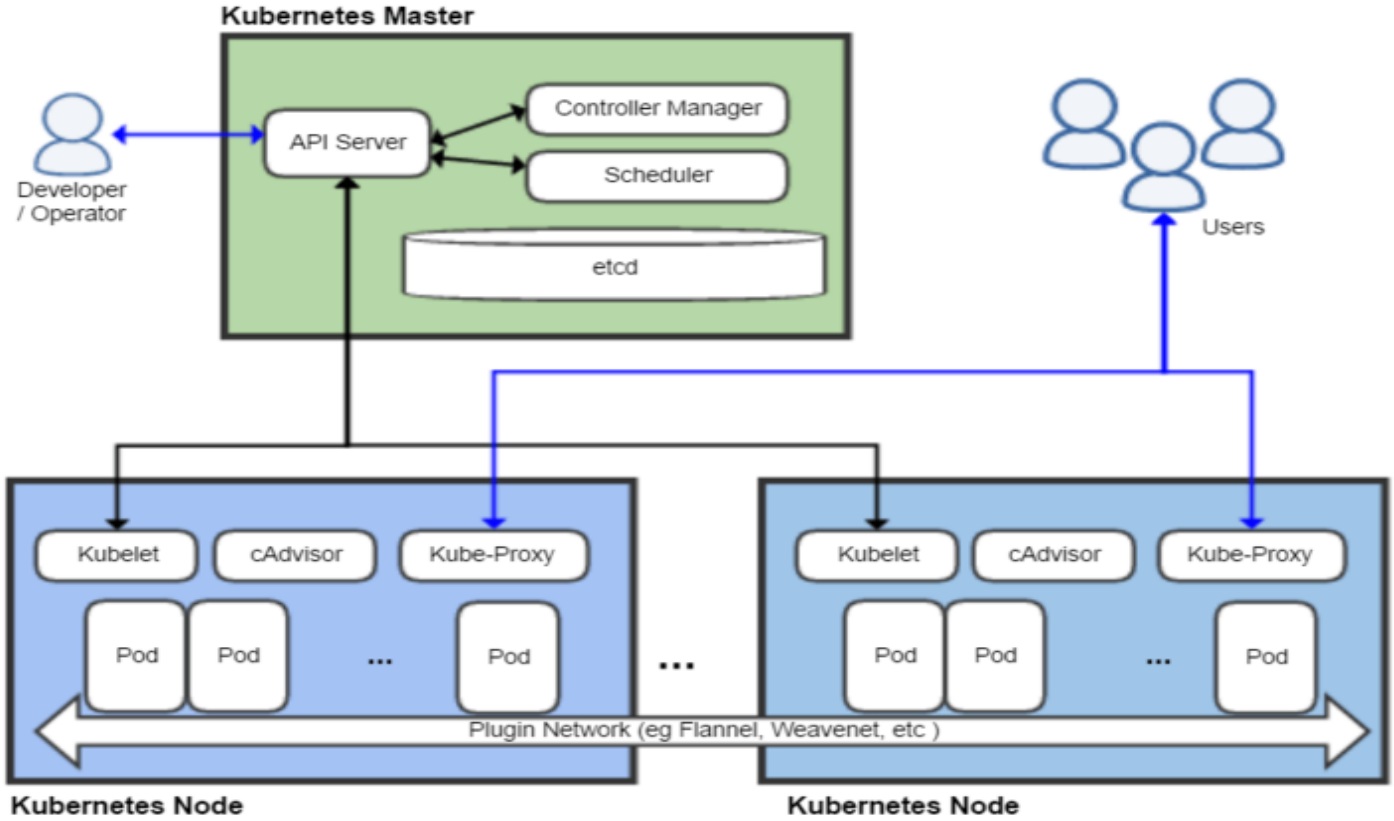
2. WAS IST KUBERNETES?

2. Kubernetes - Zusammenfassung

anderScore trust in competence

- Google-Projekt (auch: k8s, k8),
- Erstes Release 7. Juni 2014, Aktuell 1.7.4 (17. August 2017)
- Cluster für docker und rkt
- Konzepte
  - Node:
    - Maschine im Cluster
  - Pod:
    - Kleinste Einheit, 1-n Container
    - 1 Host, Ressource-Sharing
  - Controller: Mind. Einer pro Cluster
  - Service:
    - Vers. Pods, e.g. ein Tier einer Application
    - Adressierung über DNS über ENV-Variablen

2. Kubernetes



Docker, Maven, Kubernetes, Jenkins

3. ZUSAMMENSPIEL

3. Zusammenspiel: Docker

anderScore trust in competence

Ziel des Docker-Containers – Unterschiede im Detail:

	Service-Paketierung	Development-Environment	Virtueller Server
Ziel	Plattform für Service-Betrieb (PaaS)	Produktionsnahe Umgebung zur Entwicklung	Linux-Server i.a. Bereitstellen / Hosting (IaaS)
Linux-Distribution	Alpine / Snappy Core o.ä. Eigenes Basis-Image	Debian / CentOS / SLES (z.T. Ubuntu)	Alle traditionellen
Installation	Minimal (keine Deamons)	Umfangreich	Distributionsstandard
Bereitstellung / Erzeugung	Continous-Delivery / Built	Images via git o. Filesystem	Distributions-Templates
Updates	Continous-Delivery / Built	Nach Bedarf	Configuration- & Package-Management (ansible, puppet)
Persistence	IdR Keine	Einzelne Ordner	Komplett
Daemons	Nur Service	Graphische Oberfläche, Shell-Zugang	SSH (Wartungskonsole)
Monitoring / Logging	Über Plattform-Konsole – ggf. mit Einbindung	Keine	Innerhalb (Icicinga, logwatch, check_mk)

3. Zusammenspiel: Nexus

- Repository
  - Für Java-Artefakte (.war / .jar)
  - **Und Docker-Images**
    - Private Registry
    - Quelle Cluster Deployment & Builts
- System für Hands-On:
  - https://nexus.jluehr.de (Docker-API https://nexus.jluehr.de:8443/)
  - User: gs
  - Kennwort: goldschmiede

3. Zusammenspiel: maven

- Maven Docker Plugin:
  - docker-maven-plugin (ehem.)
    - Z.T. Generierung `Dockerfile`
  - dockerfile-maven (https://github.com/spotify/dockerfile-maven)
    - `Dockerfile` nicht generiert – in Project-Sources abgelegt
    - `mvn:package` – Image erstellen
    - `mvn:deploy` – Image push in Registry (d.h. nexus repository)

- Maven OWASP Plugin
  - Maven Dependencies auf Sicherheitslücken überprüfen.
  - Z.Z. keine Überprüfung des Docker-Images
    Bei eigenem Basis-Image: Einfacher Pre-Check

3. Zusammenspiel: Jenkins

anderScore trust in competence

- 3 Jobs pro Microservice
  - Build & Test master
  - Dependency Check (OWASP)
  - Deploy master (→ kubernetes)
- Zzgl. Jobs für branches
  - Kein Deploy
- Automatische Generierung Jobs über Seeds

… auch im Kubernetes-Cluster deployed

4. Zusammenfassung

1. Deployment von Microservice: Automatisierung existentiell.

2. Continius Delivery: Toolchain robust & durchdacht

3. Verschiedene Anforderungen an Container (PaaS, IaaS, Dev-Maschine)

4. Security & Patching: Teil des Builds – Monitoring Teil der Plattform

5. Trennung Developers / Operations / Build:
    1. Developers: Microservice Entwicklung
    2. Operations: Plattform Bereitstellen
    3. Build: „Dev-Ops Aufgaben“: Basis-Image, Re-Deployment

4. Hands-On

Aufgaben:

1. Installiere einen kubernets-Cluster via minicube
2. Ergänze die Services um Dockerfiles (ggf. mit Plugin)
3. Lege Deine Images in der Registry ab (Name bitte ändern)
4. Deploy‘ die Images auf Docker
5. Teste die Dependency auf Verwundbarkeiten (NVD).

Material:

- https://github.com/goldschmiede/2017-09-01-Kubernetes
- https://nexus.jluehr.de/ - User: gs Kennwort: goldschmiede