

Openshift Installation und Administration





Openshift bei Heinlein

- Openshift Origin 3 (OKD)
- OKD 4
- RHOSCP - Openshift 4 (4.13)
- 4 Cluster

Tag 1

Einführung

Cluster Konzeption und Anforderungen

Installation

CLI und Web-Console

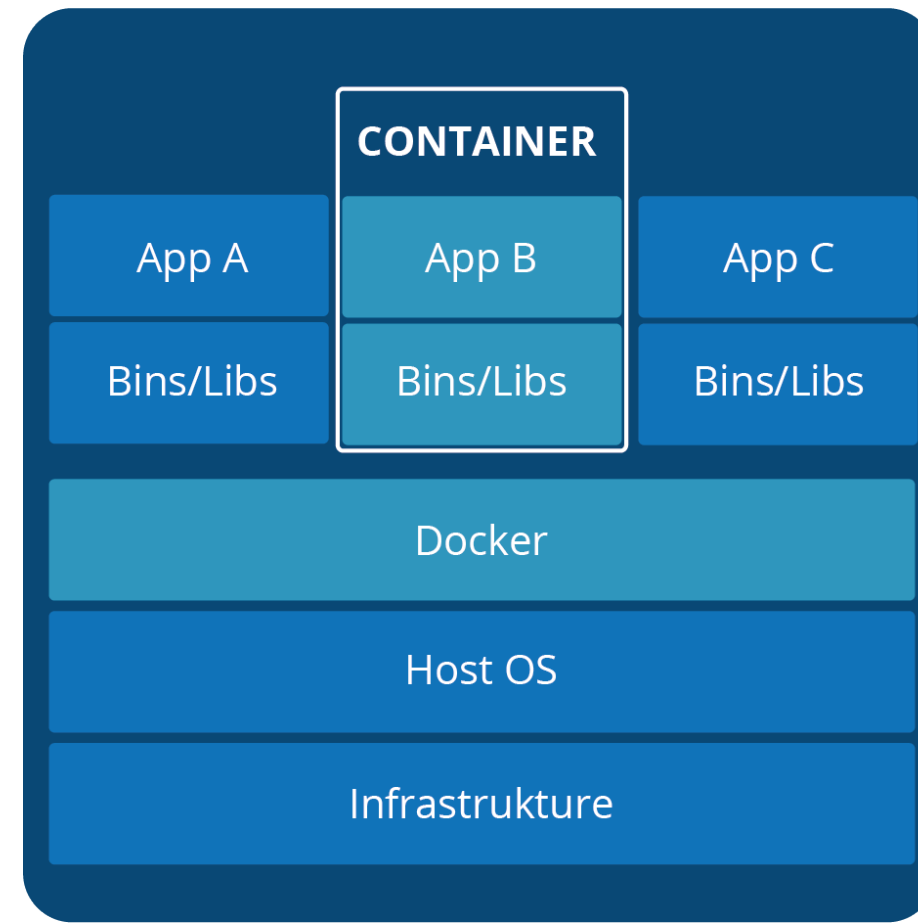
Cluster Updates

Eine kurze Geschichte der Container

- 1982 chroot
- 2000 FreeBSD Jails
- 2007 Linux Kernel cgroups
- 2008 Linux Containers LXC
- 2013 Docker
- 2018 Podman / Containerd

Was ist ein Container?

- Container Image
- **ein** Prozess
- cgroup Isolation
- bestimmter User



Und warum Container?

- Portabilität
- Isolation
- Skalierbarkeit
- schnelles Deployment
- Konsistenz
- Deklarativ
- DevOps

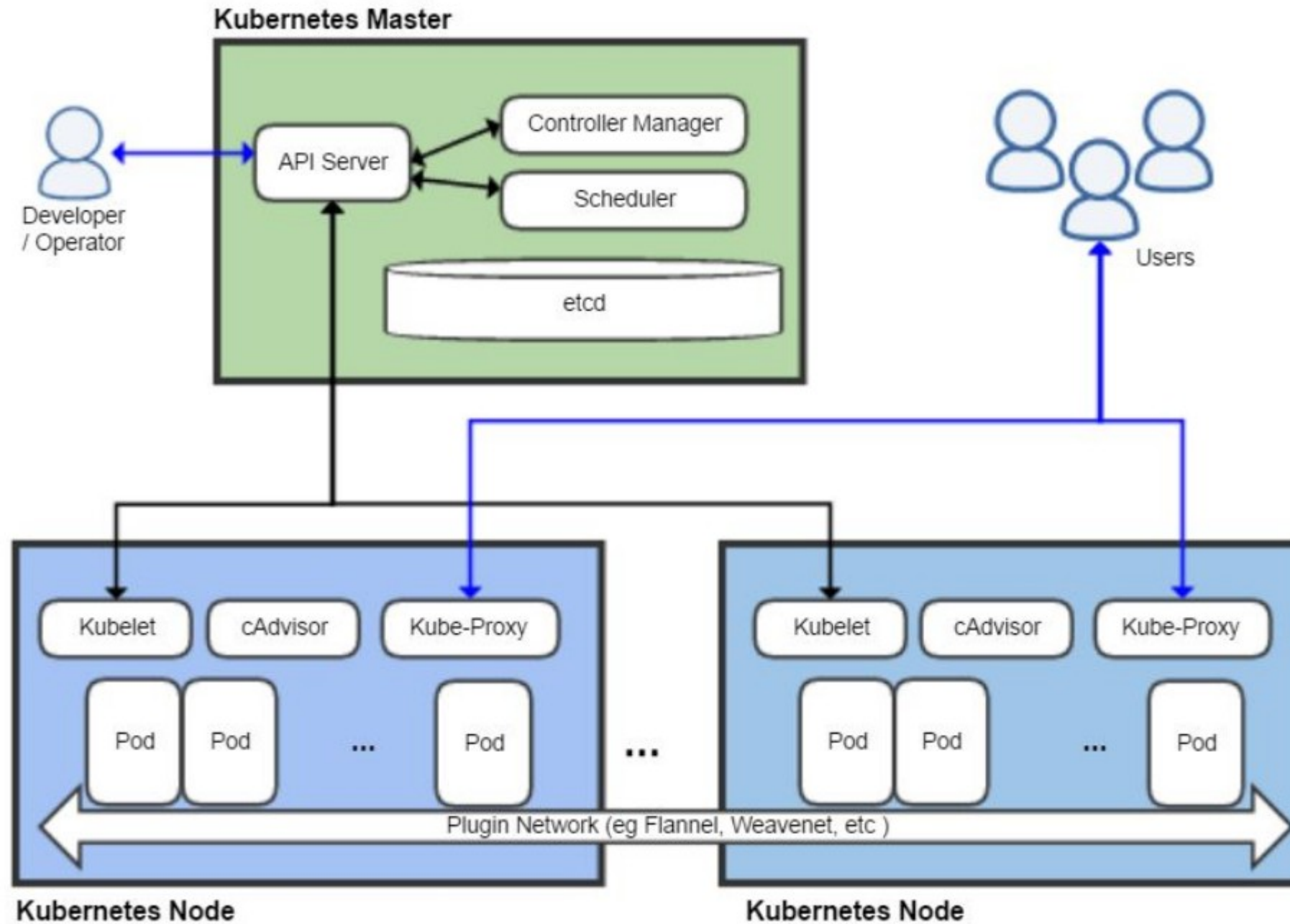
Container Runtime - Container Engine

- Schnittstelle zum OS-Kernel
- **runc** - spezifiziert nach OCI (Open Container Initiative) Standard
- **CRI-O** / Containerd

Container Orchestrierung

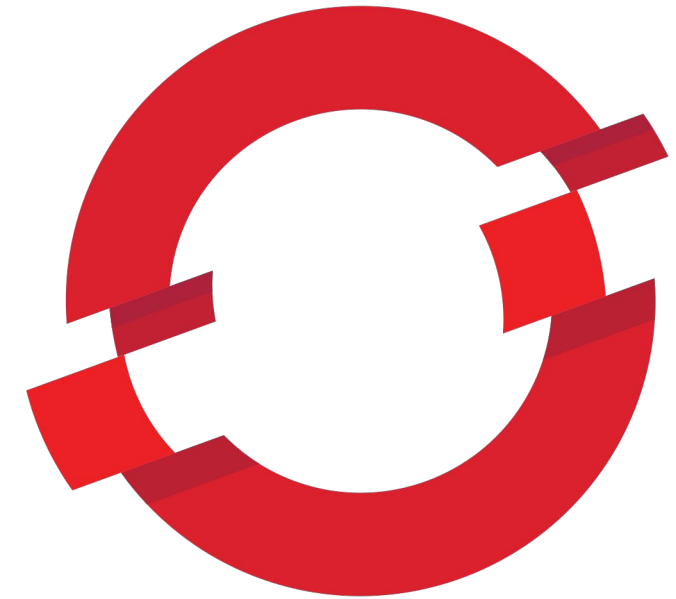
Automatisierung und Verwaltung von:

- Provisionierung und Deployment
- Konfiguration und Planung
- Ressourcenzuweisung
- Container-Verfügbarkeit
- Skalieren von Containern
- Load Balancing und Traffic Routing
- Überwachen des Containerzustands
- Sichern von Interaktionen zwischen Containern





redhat



OPENSIFT

Openshift - Kubernetes plus

- Routing
- Monitoring
- Logging
- Web Console (GUI)
- Image Builds
- Image Registry
- Built-in security
- Networking

Konzeption

Node Roles

Controlplane Nodes

- Kubernetes API
- Controller
- etcd
- Kubernetes Scheduler
- Openshift Authentication

Worker Nodes

- Compute Nodes
- User Workload
- Ingress Controller ?

Infra Nodes (optional)

- extra Label
- nicht in Subscriptions
- Cluster-Monitoring, Router, Registry

Number of worker nodes	Cluster-density (namespaces)	CPU cores	Memory (GB)
24	500	4	16
120	1000	8	32
252	4000	16, but 24 if using the OVN-Kubernetes network plug-in	64, but 128 if using the OVN-Kubernetes network plug-in
501, but untested with the OVN-Kubernetes network plug-in	4000	16	96

Empfohlene Anforderungen

Bootstrap Node:

- 4 vCPU
- 16 GB Memory
- 120 GB Storage

Master Nodes:

- 3 Nodes
- 4 vCPU
- 16 GB Memory
- 120 GB Storage

Worker Nodes:

- mind. 2 Nodes
- 4 vCPU
- 16 GB Memory
- 120 GB Storage

Und wirklich?

- erwarteter Workload / Anforderung der Applikationen
- Hochverfügbarkeit
- automatische Skalierung
- Cluster Updates
- Capacity Reserven

IPI Installation

Installer-provisioned Infrastructure (IPI)

- Der Installer erstellt und startet die Maschinen
- Benötigt einen Automatisierung auf Infrastrukturebene (z.B. ein Hypervisor)
- Ggf. müssen vorher manuell Sachen vorbereitet werden (hier genau die Installationsanleitung aus der Doku beachten)
- Der Cluster ist dann in der Lage mit der entsprechenden API zu kommunizieren und diverse Automaten anzubieten (z.B. Storage provisioning, Scaling, Health Checks, etc)

IPI

- Diverse Cloud Anbieter:
AWS, Microsoft Azure, Google Cloud
- Red Hat Virtualization (oVirt)
Red Hat OpenStack Platform (RHOSP)
- VMware vSphere
(Version 6.5 or higher)
- Bare Metal
- IBM Z / LinuxONE / Power Systems

UPI Installation

User-provisioned Infrastructure (UPI)

- Alle Infrastruktur Komponenten müssen manuell provisioniert werden
- Der Installer generiert die notwendigen Konfigurationen (Ignition-Configs)
- Die Nodes müssen mit CoreOS images gestartet werden und Initialisieren sich selbst

UPI

- Nodes
- DNS
- IP Configuration
(statische IPs oder DHCP)
- Load Balancer
- Storage

Installation Type

Interactive

- Assisted Installer
- Webbasiert
- connected

Automated

- Installer provisioned
- connected und disconnected Umgebungen

Local Agent-based

- Agent-based Installer
- ideal für disconnected Umgebungen

Full Control

- User provisioned
- maximale Konfigurierbarkeit

Installation Parameter

worüber man sich Gedanken machen sollte

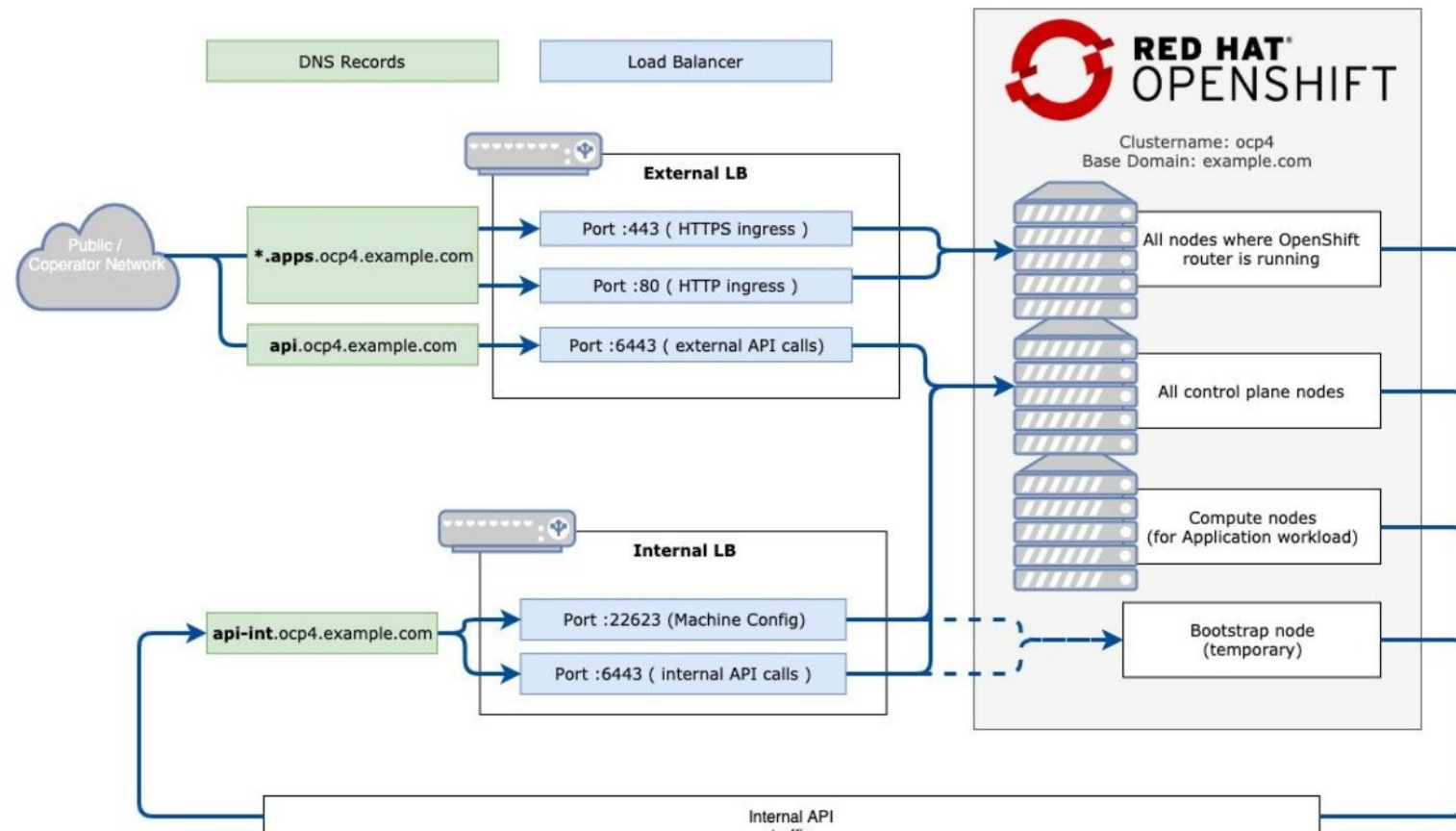
- Base Domain
- Cluster Name
- Pull Secret
- FIPS Mode
- Restricted Network?
- Internet Anbindung über Proxy?
- Company CA certificate
- Mirrored Registry

DNS

api. <cluster-name>.<base-domain>	A or CNAME	Public API
api-int. <cluster-name>.<base-domain>	A or CNAME	API access for nodes
*.apps. <cluster-name>.<base-domain>	A oder CNAME	Ingress wildcard domain

DNS und Loadbalancer

OpenShift 4 DNS & Load Balancer Overview



Netzwerkanforderungen

Machine Network

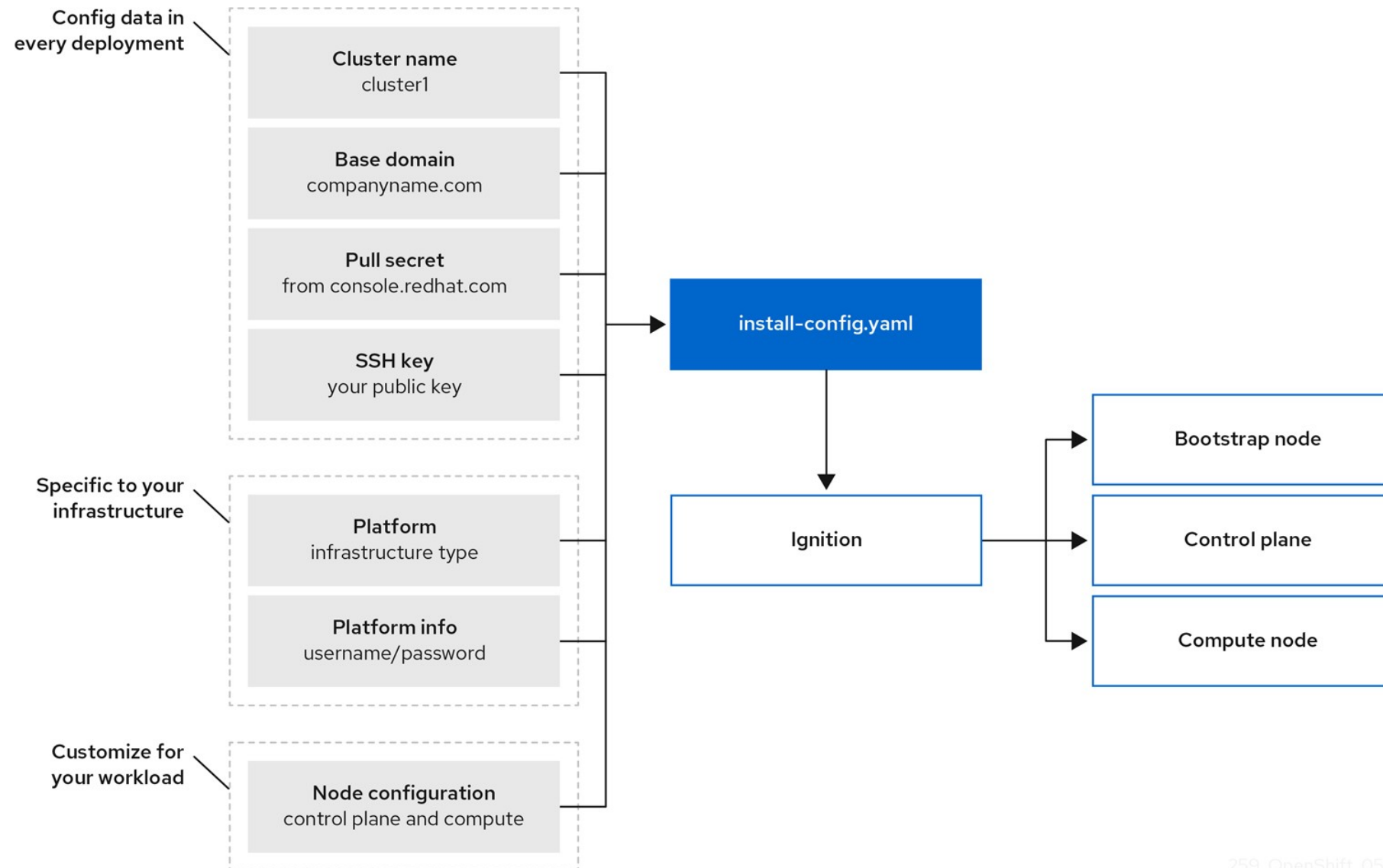
- Node Network

Cluster Network

- SDN Network
- kontrolliert vom Network Plugin
- default 10.128.0.0/14 !

Service Network

- Cluster Services
- default 172.30.0.0/16



- Cloud / Hypervisor Config
- Machine customizations
- Network customizations
- FIPS Mode
- Pull Secret
- SSH Key

```
apiVersion: v1
baseDomain: example.com ①
compute: ②
  name: worker
  replicas: 3
  platform:
    vsphere: ③
      cpus: 2
      coresPerSocket: 2
      memoryMB: 8192
      osDisk:
        diskSizeGB: 120
controlPlane: ②
  name: master
  replicas: 3
  platform:
    vsphere: ③
      cpus: 4
      coresPerSocket: 2
      memoryMB: 16384
      osDisk:
        diskSizeGB: 120
metadata:
  name: cluster ④
platform:
  vsphere:
    vcenter: your.vcenter.server
    username: username
    password: password
    datacenter: datacenter
    defaultDatastore: datastore
    folder: folder
    resourcePool: resource_pool ⑤
    diskType: thin ⑥
    network: VM_Network
    cluster: vsphere_cluster_name ⑦
    apiVIPs:
      - api_vip
    ingressVIPs:
      - ingress_vip
fips: false
pullSecret: '{"auths": ...}'
sshKey: 'ssh-ed25519 AAAA...'
```

Installer Commands

- openshift-install create cluster
- openshift-install create install-config
- openshift-install create manifests
- openshift-install create ignition-configs
- openshift-install wait-for bootstrap-complete
- openshift-install wait-for install-complete
- openshift-install destroy bootstrap
- openshift-install destroy cluster

Pitfalls

- Installation muss innerhalb von 24 Stunden nach Generierung der Ignition Configs erfolgen
- Der Cluster darf die ersten 24 Stunden nicht ausgeschaltet werden
- Installationskonfiguration wird vom Installer geschluckt
- zu kleine IP Range für Node Networks (hostPrefix)
- falsches IP Netz - Routing

Übung

- Cluster Installation

Doku: <https://docs.openshift.com/container-platform/4.13/installing/index.html>

Web Console

CLI

Update

Update Channels

- fast-4.x
- stable-4.x
- eus-4.x
- candidate-4.x

Update Graph

Channel *

fast-4.14

Architecture *

x86_64

☒ Include hotfix versions

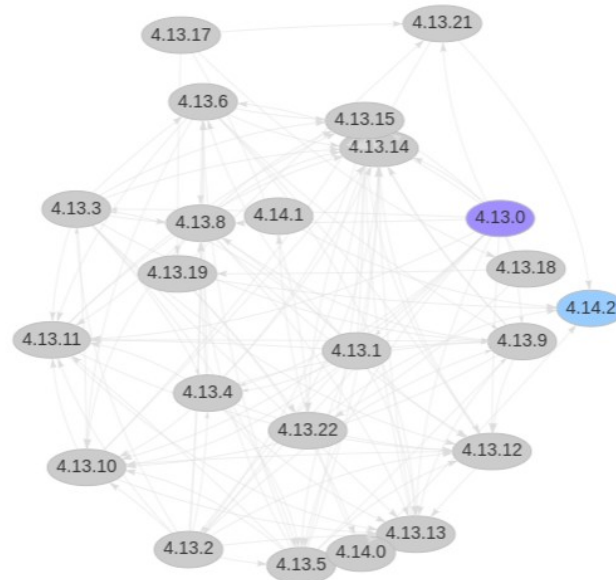
Note

Your cluster will warn you of [any matched known issues](#).

fast-4.14

Legend

- Lowest version in the channel
- Highest version in the channel



Over-the-air Updates

- etwa alle 2 Wochen kommt ein Minor Update
- Rolling Update -> bedingt reboot aller Nodes
- cluster-version-operator aktualisiert alle Cluster Operatoren
- Die Operatoren aktualisieren dann die Komponenten
- CoreOS Update

oc update

- Clusterzustand checken: Nodes, Clusteroperatoren, Clusterversion
- oc adm upgrade plan
- oc adm upgrade --to-latest
- oc adm upgrade --to="4.x."

Übung

- Cluster Update

Doku: <https://docs.openshift.com/container-platform/4.13>

Tag 2

Operatoren

User und Rechte-Management

HA

Machines / Nodes Machine Management

Alles nur Ressourcen

- Der komplette Zustand des Clusters wird durch verschiedene Ressourcen abgebildet
- Cluster Ressourcen
- Namespace Ressourcen
- Die Ressourcen werden im etcd gespeichert
- Die API selbst ist stateless
- Custom Resource Definitions (CRD)

Wichtige Ressource Typen

- Namespace
- Project
- ClusterRole
- User
- PersistentVolume
- Node
- Pod
- Deployment
- ConfigMap
- Secret
- Service
- Route

Operatoren

- Operatoren automatisieren die Erstellung, Konfiguration und Verwaltung
- überwachen Anwendungen während ihrer Ausführung
- können automatisch Daten sichern
- das System nach Fehlern wiederherstellen
- Anwendungen kontinuierlich upgraden

observe → diff → act

Cluster Operatoren

Wichtige Operatoren

- apiserver-operator
- etcd-operator
- network-operator
- machine-config-operator

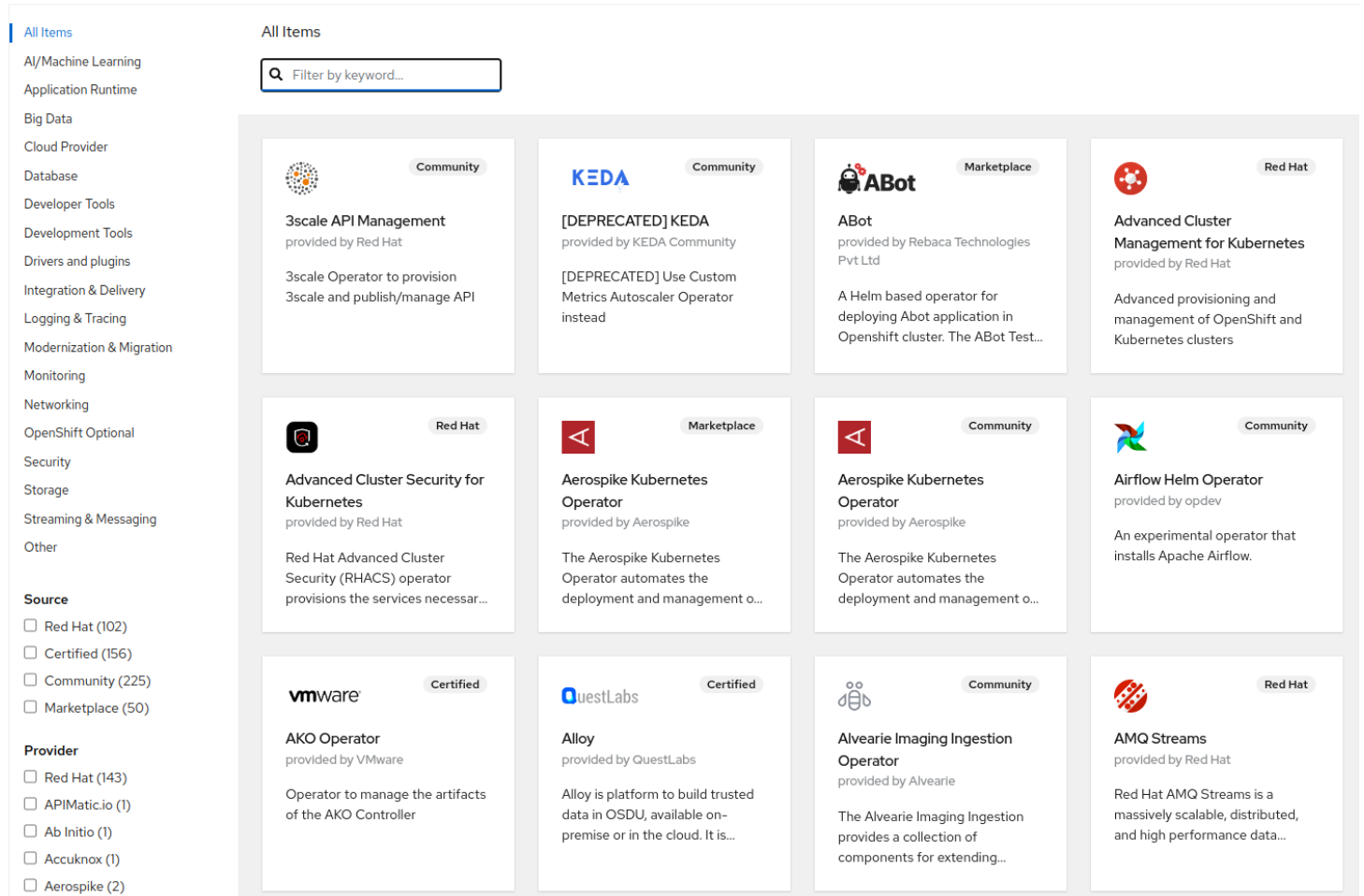
<https://docs.openshift.com/container-platform/4.13/operators/operator-reference.html>

User Operatoren

- Nachträglich installierte Operatoren
- Marketplace (<https://operatorhub.io>)
- OLM Operator Lifecycle Manager
- Cluster-Admin Rechte

OperatorHub

Discover Operators from the Kubernetes community and Red Hat partners, curated by Red Hat. You can purchase commercial software through [Red Hat Marketplace](#). You can install Operators on your clusters to provide services to your developers. After installation, the Operator capabilities will appear in the [Developer Catalog](#) providing a self-service experience.



The screenshot shows the OperatorHub interface with a sidebar on the left and a main grid of operator cards. The sidebar includes a search bar and a list of categories: AI/Machine Learning, Application Runtime, Big Data, Cloud Provider, Database, Developer Tools, Development Tools, Drivers and plugins, Integration & Delivery, Logging & Tracing, Modernization & Migration, Monitoring, Networking, OpenShift Optional, Security, Storage, Streaming & Messaging, and Other. Below the categories, there are filters for Source (Red Hat (102), Certified (156), Community (225), Marketplace (50)) and Provider (Red Hat (143), APIMatic.io (1), Ab Initio (1), Accuknox (1), Aerospike (2)).

The main grid displays 12 operator cards, each with a logo, name, provider, and description:

- 3scale API Management** (Community, provided by Red Hat): 3scale Operator to provision 3scale and publish/manage API
- [DEPRECATED] KEDA** (Community, provided by KEDA Community): [DEPRECATED] Use Custom Metrics Autoscaler Operator instead
- ABot** (Marketplace, provided by Rebaca Technologies Pvt Ltd): A Helm based operator for deploying Abot application in Openshift cluster. The ABot Test...
- Advanced Cluster Management for Kubernetes** (Red Hat, provided by Red Hat): Advanced provisioning and management of OpenShift and Kubernetes clusters
- Advanced Cluster Security for Kubernetes** (Red Hat, provided by Red Hat): Red Hat Advanced Cluster Security (RHACS) operator provisions the services necessar...
- Aerospike Kubernetes Operator** (Marketplace, provided by Aerospike): The Aerospike Kubernetes Operator automates the deployment and management o...
- Aerospike Kubernetes Operator** (Community, provided by Aerospike): The Aerospike Kubernetes Operator automates the deployment and management o...
- Airflow Helm Operator** (Community, provided by opdev): An experimental operator that installs Apache Airflow.
- AKO Operator** (Certified, provided by VMware): Operator to manage the artifacts of the AKO Controller
- Alloy** (Certified, provided by QuestLabs): Alloy is platform to build trusted data in OSDU, available on-premise or in the cloud. It is...
- Alvearie Imaging Ingestion Operator** (Community, provided by Alvearie): The Alvearie Imaging Ingestion provides a collection of components for extending...
- AMQ Streams** (Red Hat, provided by Red Hat): Red Hat AMQ Streams is a massively scalable, distributed, and high performance data...

User Management

- OpenShift hat keine vollständige Userverwaltung
- User müssen durch externen Identity Provider authentifiziert werden
- Gruppen -> Liste von Usern
- Rechte können direkt an User oder an Gruppen vergeben werden

OpenShift Identity Provider

Statisch

- kubeadmin
- htpasswd

Directory Service

- LDAP
- Basic Auth
- Request Header

Oauth

- Github / Gitlab
- OpenID Connect
- Keystone
- Google

Beispiel LDAP

```
apiVersion: config.openshift.io/v1
kind: OAuth
metadata:
  name: cluster
spec:
  identityProviders:
  - name: ldapidp ①
    mappingMethod: claim ②
    type: LDAP
    ldap:
      attributes:
        id: ③
        - dn
        email: ④
        - mail
        name: ⑤
        - cn
        preferredUsername: ⑥
        - uid
      bindDN: "" ⑦
      bindPassword: ⑧
        name: ldap-secret
      ca: ⑨
        name: ca-config-map
      insecure: false ⑩
      url: "ldaps://ldaps.example.com/ou=users,dc=acme,dc=com?uid" ⑪
```

Mapping

Claim

- default, User wird mit Identity angelegt
- schlägt fehl wenn User mit anderer Identität existiert

Lookup

- Identität muss existieren, manuelle Provisionierung von Usern

Generate

- wie claim aber User wird ggfs mit „unique name“ angelegt

Add

- Identity wird zum User hinzugefügt

Emergency User

- kubeadmin
(wird bei der Installation generiert)
- eigene Client CA
(was im CN steht wird als Username genommen)
- system:admin kubeconfig
(wird vom Installer generiert)
- eigener Service Account
(kubeconfig generieren und abspeichern)

Übung

- htpasswd Identity Provider konfigurieren
- einen User anlegen
- Updaten und zweiten User hinzufügen - oc
- Updaten und ersten User wieder löschen - yamI file

Doku: https://docs.openshift.com/container-platform/4.13/authentication/identity_providers/configuring-htpasswd-identity-provider.html

RBAC (role based access control)

- Zugriffsrechte im Cluster werden über Roles / Clusterroles gesteuert

Default Roles:

- cluster-admin
- cluster-reader
- self-provisioner
- admin
- edit
- view

Roles und Clusterroles

- Rollen bestehen aus einem spezifischen Set von Rechten
- Rechte bestehen aus einem Verb und ein Ressourcetyp
(zum Beispiel: get pods)
- **kein explizites Deny**
- Principle of the least Privilege: nur die Rechte geben, die absolut notwendig sind!

Verbs

- GET
- CREATE
- APPLY
- UPDATE
- PATCH
- DELETE
- PROXY
- LIST
- WATCH
- DELETEDCOLLECTION

```
1  apiVersion: rbac.authorization.k8s.io/v1
2  kind: ClusterRole
3  metadata:
4    name: pod-shell
5  rules:
6    - apiGroups:
7      - ""
8      resources:
9        - pods
10       - pods/exec
11     verbs:
12       - create
13       - get
14
15
```

Roles und RoleBindings

- **Role** -> in einzelnen Namespaces
- **ClusterRole** -> im ganzen Cluster
- **RoleBinding** -> bindet Role an Subject (namespace bezogen)
- **ClusterRoleBinding** -> bindet ClusterRole an Subject
- ClusterRoles können auch in RoleBindings genutzt werden
- Zulässige Subjects: User, Group, ServiceAccount

Achtung: OpenShift akzeptiert RoleBindings auch wenn die Role und / oder das Subject nicht existiert

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: <rolebinding_name>
  namespace: <current_project_name>
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: <role_name>
subjects:
- kind: ServiceAccount
  name: <service_account_name>
  namespace: <current_project_name>
```

```
❯ oc describe roles.rbac.authorization.k8s.io prometheus-k8s
Name:          prometheus-k8s
Labels:        app.kubernetes.io/component=prometheus
               app.kubernetes.io/instance=k8s
               app.kubernetes.io/name=prometheus
               app.kubernetes.io/part-of=openshift-monitoring
               app.kubernetes.io/version=2.39.1
Annotations:   <none>
PolicyRule:
  Resources                                Non-Resource URLs  Resource Names      Verbs
  -----                                -
  endpoints                               []                 []                  [get list watch]
  pods                                    []                 []                  [get list watch]
  services                                []                 []                  [get list watch]
  ingresses.extensions                    []                 []                  [get list watch]
  ingresses.networking.k8s.io            []                 []                  [get list watch]
```


ServiceAccounts

- technische User des Clusters
- Automatisierte Tasks

```
oc create sa my-bot
```

```
oc create token my-bot
```

```
oc get pods -token
```

```
oc get pods -A
```

```
- create clusterrolebinding sa-view
```

Übung

- welche Clusterrolebindings referenzieren auf die self-provisioner Clusterrole?
- was erlaubt die Clusterrole self-provisioner
- entferne die Clusterrole vom entsprechenden Subject

Gruppen:

- erstelle 2 Gruppen (admin und dev) und füge jeweils einen User zu einer Gruppe hinzu
- die Admin Gruppe kriegt Cluster-Admin Rechte
- die Developer Gruppe kriegt nur die Rechte im Cluster zu lesen
- erstelle ein neues Projekt und gebe dem Dev-User die Rechte hier Pods zu erstellen
- wie kann ich das testen

HA

- **Zwingend erforderlich:**
 - 3 Master
 - 2 Worker
- etcd nutzt Raft Algorithmus zur Vermeidung von Split-Brain
- es darf maximal 1 Master ausfallen!
- Reserve Capacity überwachen

Hochverfügbarkeit im Cluster

- Replicas der Pods
- Resource Allocation / Quality of Service
- PodDisruptionBudget
- Health Checks
- IPI - machine health checks

Machine Health Check

- prüft den Zustand von Nodes
- kann Nodes automatisch löschen und neu provisionieren

```
1  apiVersion: machine.openshift.io/v1beta1
2  kind: MachineHealthCheck
3  metadata:
4    name: example
5    namespace: openshift-machine-api
6  spec:
7    selector:
8      matchLabels:
9        machine.openshift.io/cluster-api-cluster: my-cluster
10       machine.openshift.io/cluster-api-machine-role: worker
11       machine.openshift.io/cluster-api-machine-type: worker
12       machine.openshift.io/cluster-api-machineset: my-machine-set
13    unhealthyConditions:
14      - type: Ready
15        status: Unknown
16        timeout: 300s
17      - type: Ready
18        status: 'False'
19        timeout: 300s
20    maxUnhealthy: 40%
21
```

Health Checks

Liveness Probe

- lebt der Container?
- fail → **Restart**

HTTP-GET-Request, TCP-Port Check, Shell Command

Readiness Probe

- kann der Container Traffic annehmen?
- fail → kein Traffic zum Container - kein Restart

Failing ist okay

- jeder Pod kann zu jeder Zeit gekillt werden

Warum?

- manuell (Admin, Developer, etc)
- Node Failure / Maintenance
- Pod / Container out-of-memory
- Node out-of-memory

Machine Management 1

Node

- Entitäten auf denen der Workload läuft
- Scheduler
- kann Bedingungen für Workload haben

Machine

- 1 - 1 Mapping zu Node
- repräsentiert eine Maschine / VM etc
- enthält die Parameter dieser Instanz

MachineSet

- Menge an gleichen Maschinen (worker, infra)
- Wird zum Skalieren benutzt

Machine Mangement 2

Machine Health Check

- Erkennt kaputte Maschinen
- provisioniert neue Maschinen wenn Check fails

Machine Config

- einzelne Ignition Configs
- eigene Konfigurationen

Machine Config Pool

- mehrere Machine Configs
- Konfiguration der Nodes
- Änderung - **Reboot**

Kubelet Config

- Ermöglicht die Konfiguration von kubelet auf den Nodes zu verändern
- Warum sollte man das tun?
- kubelet reserviert Ressourcen für System Prozesse
- Default: 0,5CPU / 0,5GB Memory

```
apiVersion: machineconfiguration.openshift.io/v1
kind: KubeletConfig
metadata:
  name: dynamic-node 1
spec:
  autoSizingReserved: true 2
  machineConfigPoolSelector:
    matchLabels:
      pools.operator.machineconfiguration.openshift.io/worker: "" 3
#...
```

Übung

- Kubelet Config
- eigene Machine Config
- erstelle ein neues Machine Set für kleinere Worker Nodes
- ersetze den aktuellen Worker Node durch einen kleineren Worker Node


Logging?

- Container loggen nach stdout / stderr
- Logs hängen am Pod. Pod weg = Logs weg
- Cluster Logs
- Audit Log

Openshift Logging Stack

- Fluentd
- ElasticSearch
- Kibana

- OpenShift Logging Operator (nur mit Subscription)
- Komponenten inzwischen deprecated: Fluentd, Kibana, Elasticsearch Operator

 **Red Hat OpenShift Logging**
5.8.0 provided by Red Hat, Inc. ✕

[Install](#)

Latest version
5.8.0

Capability level

- ☒ Basic Install
- ☒ Seamless Upgrades
- ☐ Full Lifecycle
- ☐ Deep Insights
- ☐ Auto Pilot

Source
Red Hat

Provider
Red Hat, Inc

Infrastructure features
Disconnected
Proxy-aware

Red Hat OpenShift Logging

The Red Hat OpenShift Logging Operator orchestrates and manages the aggregated logging stack as a cluster-wide service.

Features

- **Create/Destroy:** Launch and create an aggregated logging stack to support the entire OCP cluster.
- **Simplified Configuration:** Configure your aggregated logging cluster's structure like components and end points easily.

Prerequisites and Requirements

Red Hat OpenShift Logging Namespace

Cluster logging and the Red Hat OpenShift Logging Operator is only deployable to the **openshift-logging** namespace. This namespace must be explicitly created by a cluster administrator (e.g. `oc create ns openshift-logging`). To enable metrics service discovery add namespace label `openshift.io/cluster-monitoring: "true"`.

For additional installation documentation see [Deploying cluster logging](#) in the OpenShift product documentation.

Alternativen

- Grafana Loki
- ELK - ElasticSearch / Logstash / Kibana / Beats

Komponenten im Cluster oder Logs ableiten

- z.B. filebeat → externer Elasticsearch / Logstash

Filebeat Deployment

- namespace
- serviceaccount
- clusterrole
- clusterrolebinding
- daemonset - tolerations!
- secret - log forward host
- filebeat config