

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

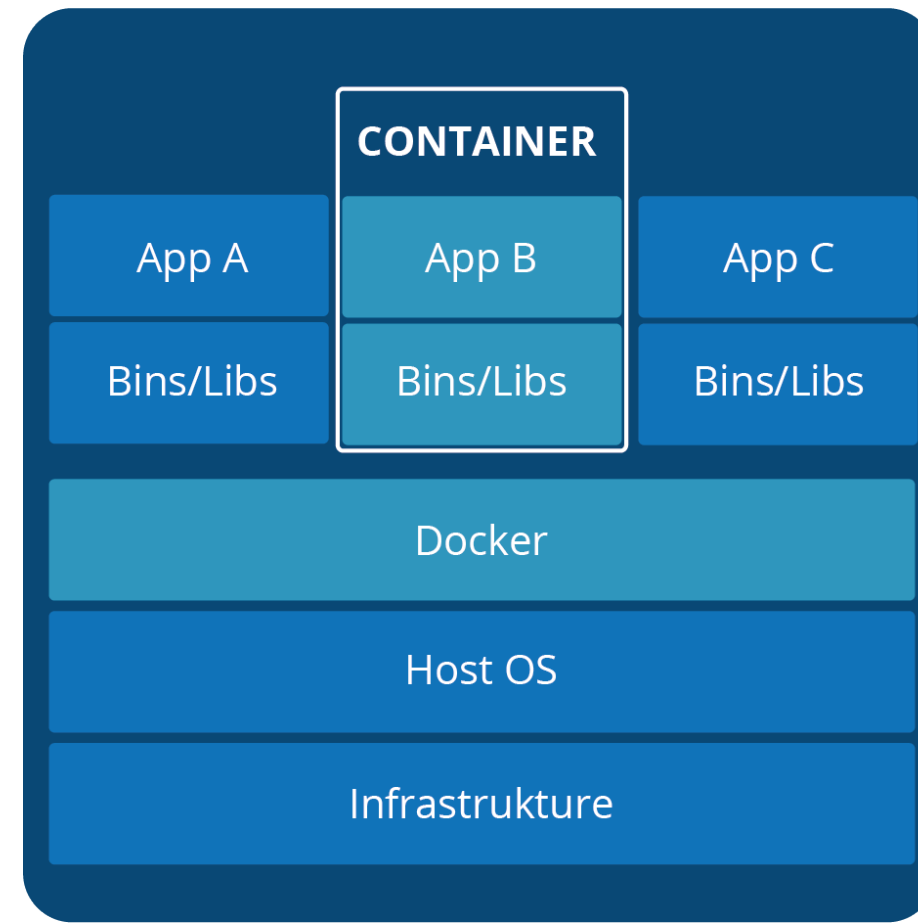
Cluster Updates

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

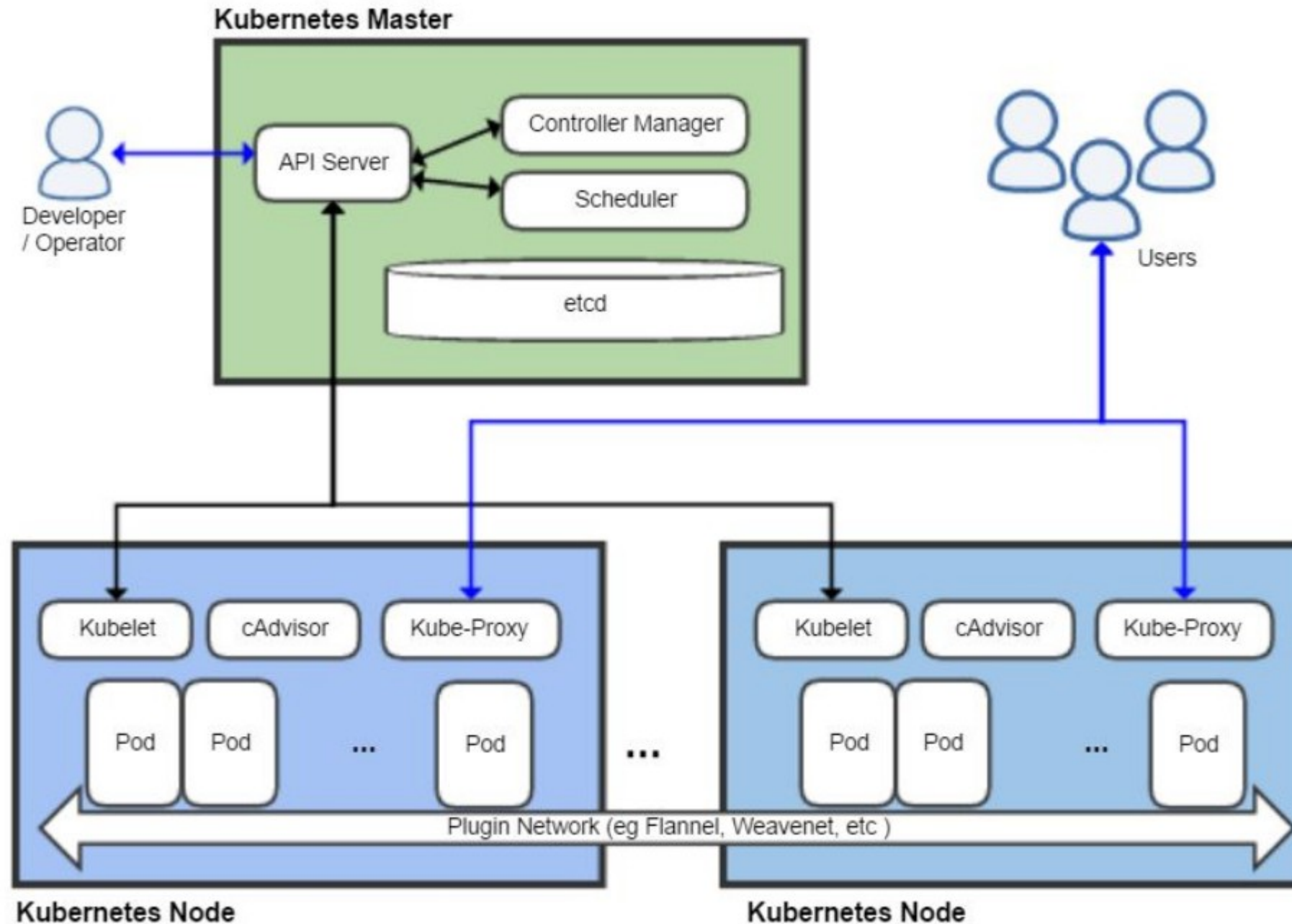
Container Runtime - Container Engine

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

Container Orchestrierung

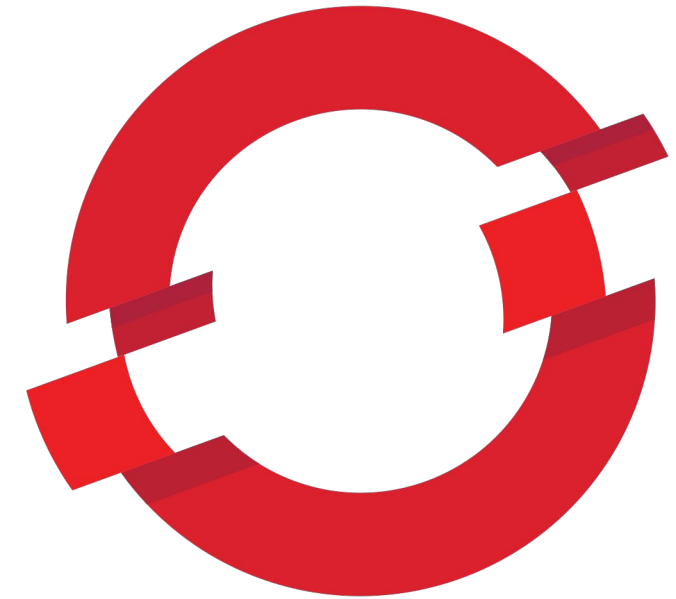
Automatisierung und Verwaltung von:

- 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

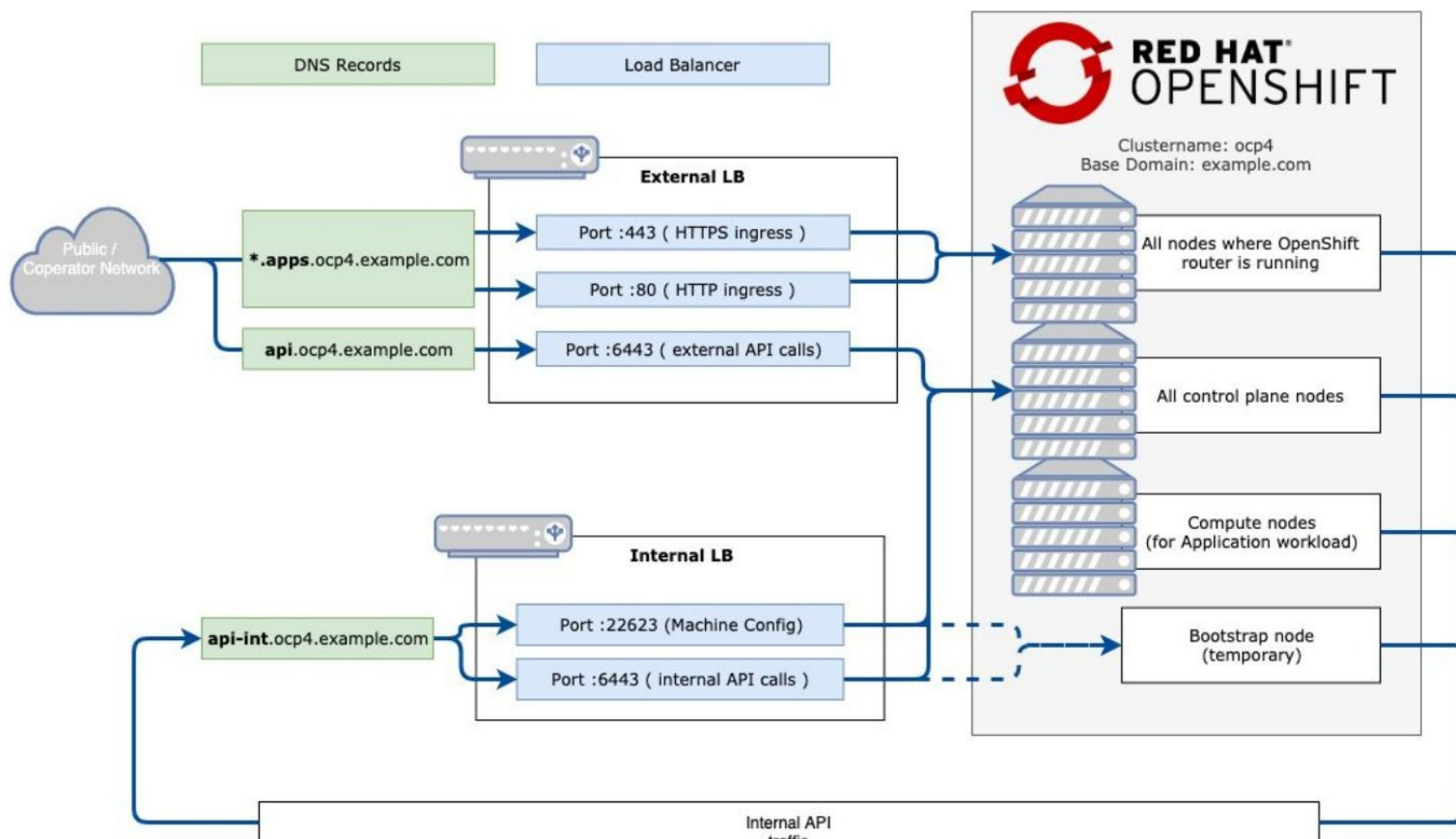
- Base Domain
- Cluster Name
- Pull Secret
- FIPS Mode
- Restricted Network?
- Internet Anbindung über Proxy?
- Company CA certificate
- Mirrored Registry

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

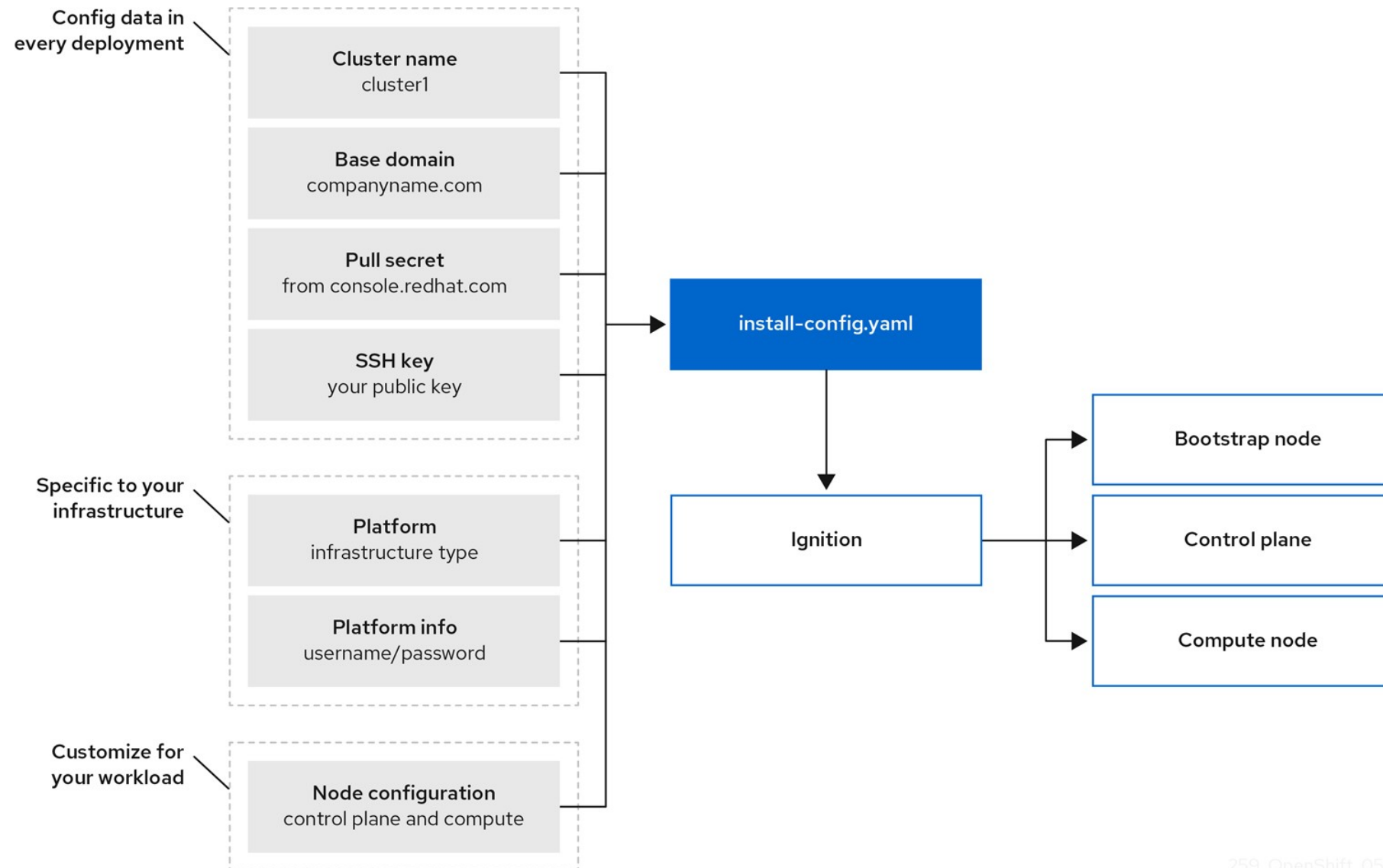
- Node Network

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

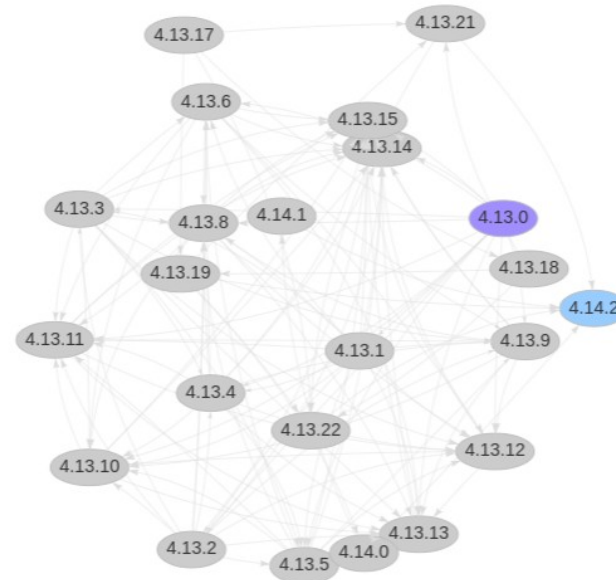
Note

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

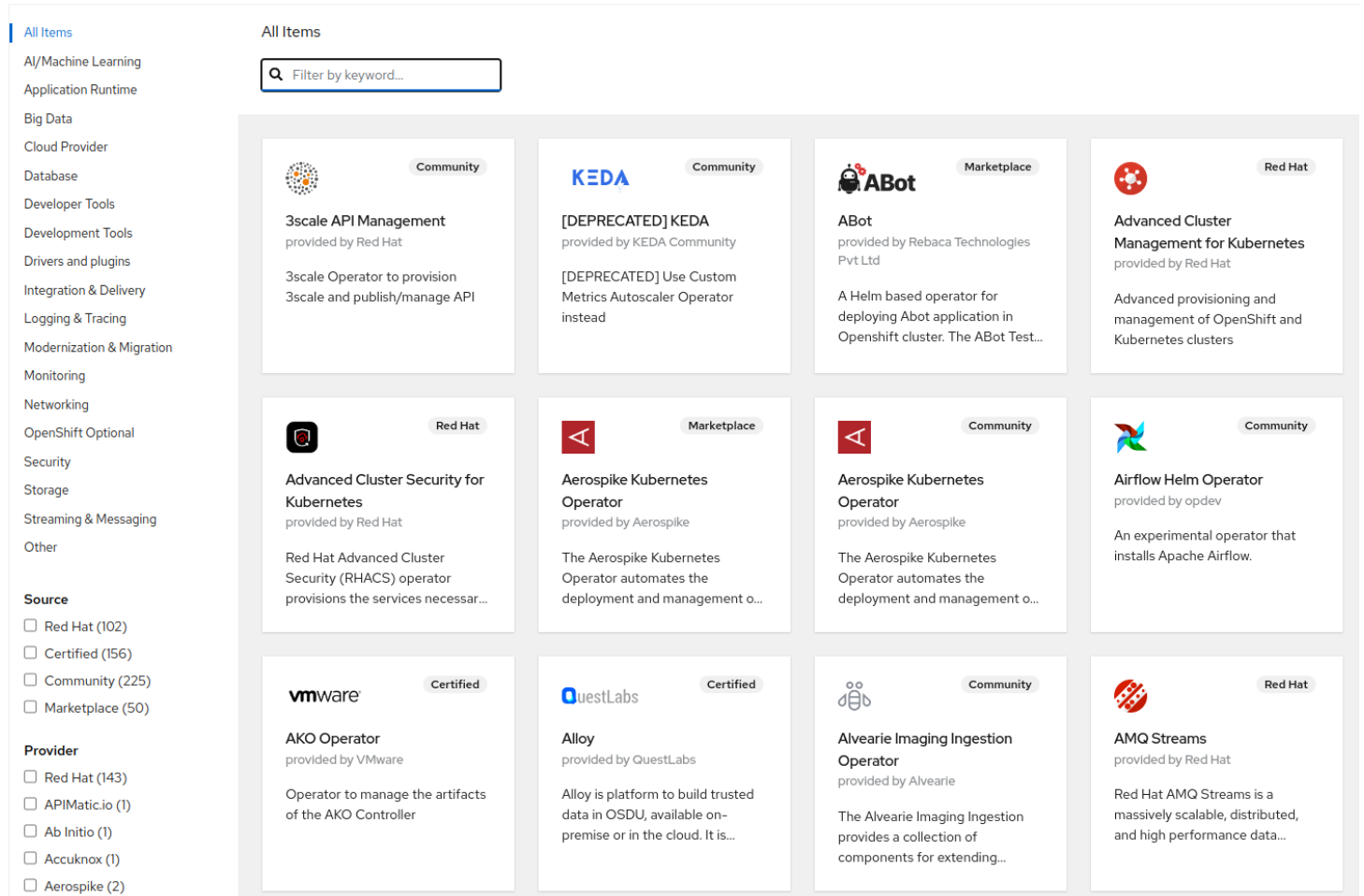
<https://docs.openshift.com/container-platform/4.13/operators/operator-reference.html>

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 displays the OperatorHub interface. On the left is a sidebar with a category list and a 'Source' filter. The main area shows a grid of operator cards. Each card includes an icon, a name, a provider, and a brief description.

Operator Name	Provider	Source
3scale API Management	provided by Red Hat	Community
[DEPRECATED] KEDA	provided by KEDA Community	Community
ABot	provided by Rebeca Technologies Pvt Ltd	Marketplace
Advanced Cluster Management for Kubernetes	provided by Red Hat	Red Hat
Advanced Cluster Security for Kubernetes	provided by Red Hat	Red Hat
Aerospike Kubernetes Operator	provided by Aerospike	Marketplace
Aerospike Kubernetes Operator	provided by Aerospike	Community
Airflow Helm Operator	provided by opdev	Community
AKO Operator	provided by VMware	Certified
Alloy	provided by QuestLabs	Certified
Alvearie Imaging Ingestion Operator	provided by Alvearie	Community
AMQ Streams	provided by Red Hat	Red Hat

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 Devloper 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

Monitoring

ClusterOperator Monitoring

- deployed und managed den lifecycle des Monitoring Stacks
- CRD's
 - Prometheus (Prometheus Instanzen)
 - Servicemonitor (scrape config services)
 - Podmonitor (scrape config Pods) (User Workload)
 - Alertmanager (deploy, manage Alertmanager Instanzen)
 - Prometheusrules (managing Prometheusrules)

Cluster Monitoring Komponenten

Cluster Monitoring Operator - deployed und managed vom ClusterversionOperator

Prometheus Operator - Prometheus und Alertmanager Instanzen

Prometheus - Monitoring System

Prometheus Adapter - exposed Cluster Ressource Metrics „oc adm top nodes/pods“

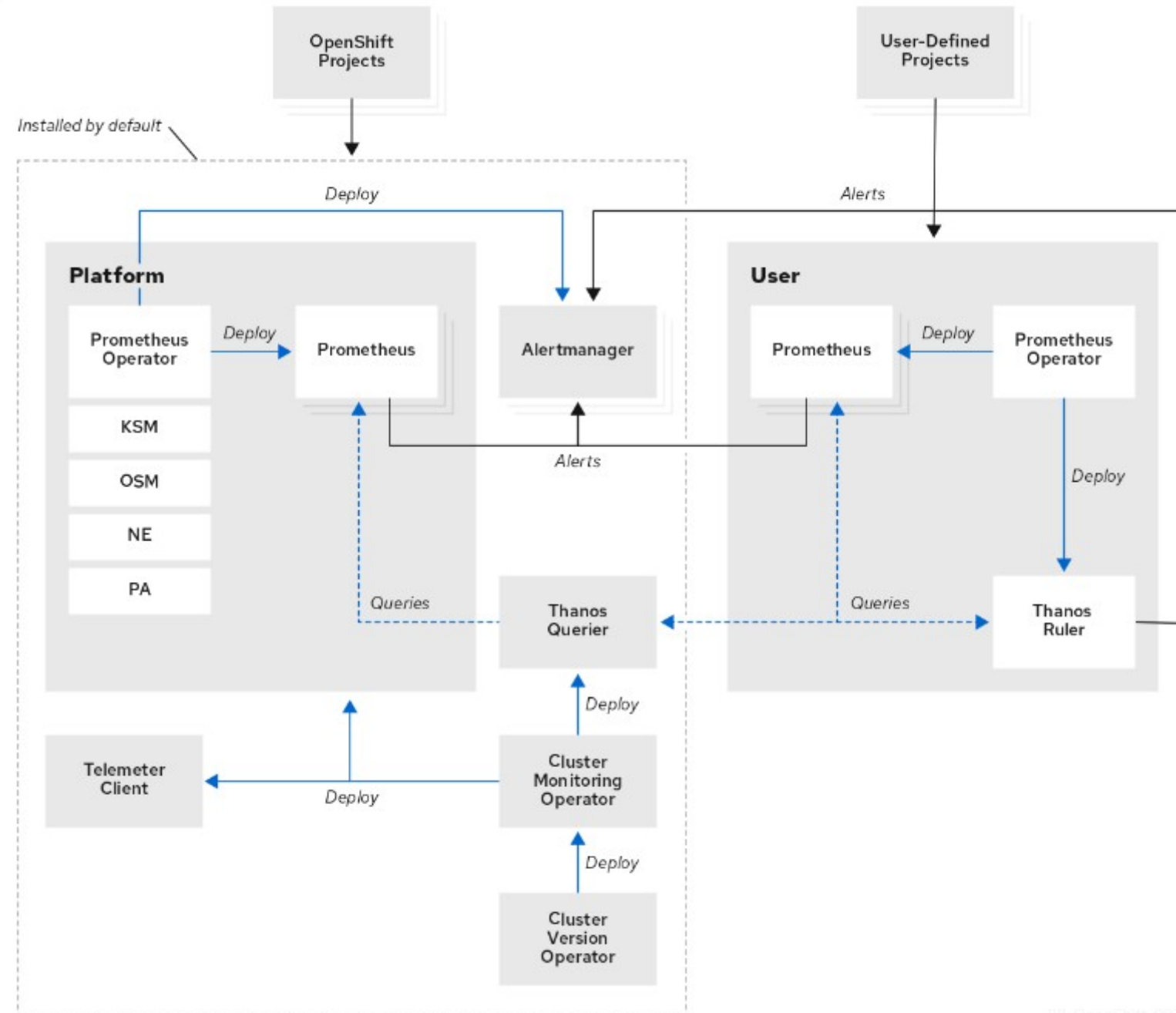
Alertmanager - Alert Handling

kube-state-metrics agent / openshift-state-metrics agent - Exporter

Node Exporter - Exporter

Thanos Querier - Aggregierung, Deduplizierung von Metriken

Telemeter Client - der Draht zu RedHat



100_OpenShift_0920

Default Targets

- CoreDNS
- Elasticsearch (if Logging is installed)
- etcd
- Fluentd (if Logging is installed)
- HAProxy
- Image registry
- Kubelets
- Kubernetes API server
- Kubernetes controller manager
- Kubernetes scheduler
- OpenShift API server
- OpenShift Controller Manager
- Operator Lifecycle Manager (OLM)

Jede Komponente ist selbst für ihr Monitoring verantwortlich.

Cluster Monitoring Config

- ConfigMap
- Name: cluster-monitoring-config
- Namespace: openshift-monitoring

Best Practices:

- Block Storage für Prometheus und Alertmanager
- Ressource Requests
- Ressource Limits
- InfraNodes - Tolerations

```
1 kind: ConfigMap
2 apiVersion: v1
3 metadata:
4   name: cluster-monitoring-config
5   namespace: openshift-monitoring
6 data:
7   config.yaml: |
8     enableUserWorkload: false
9     alertmanagerMain:
10       volumeClaimTemplate:
11         metadata:
12           name: alertmanager
13         spec:
14           storageClassName: block-storage
15           resources:
16             requests:
17               storage: 1Gi
18   prometheusK8s:
19     retention: 21d
20     volumeClaimTemplate:
21       metadata:
22         name: prometheus
23       spec:
24         storageClassName: block-storage
25         resources:
26           requests:
27             storage: 100Gi
```

User Workload Monitoring

Prometheus Operator

Prometheus

Thanos Ruler - Evaluation von Rules und Alerts für User Projekten

Targets:

- bereitgestellte Metriken an Service Endpoints
 - Pods in User Projekten
-
- UserWorkloadMonitoring muss aktiviert sein
 - ServiceMonitor
 - PodMonitor

Alerting

- by default nicht konfiguriert

Receiver:

- Mail
- PagerDuty
- Slack
- Webhook

Konfiguration: Cluster Settings - Configuration - Alertmanager

Eigene Alerts

```
1  apiVersion: monitoring.coreos.com/v1
2  kind: PrometheusRule
3  metadata:
4    name: cluster-update-available
5    namespace: openshift-cluster-version
6  spec:
7    groups:
8      - name: cluster-updates
9        rules:
10         - alert: ClusterUpdateAvailable
11           annotations:
12             message: >-
13               A new cluster update is available on your selected channel
14           expr: |
15             cluster_version_available_updates > 0
16           for: 5m
17           labels:
18             severity: info
```


Übung

- Konfiguriere den Monitoring Stack
- 1 Gi Storage für den Prometheus und Alertmanager
- Requests und Limits
- Retention

https://docs.openshift.com/container-platform/4.13/monitoring/configuring-the-monitoring-stack.html#modifying-retention-time-and-size-for-prometheus-metrics-data_configuring-the-monitoring-stack

Cluster Security

Security Context Constraints (SCC)

- kontrolliert Rechte für Pods
- ohne SCC keine erweiterten Rechte - kein Scheduling
- erlaubt Pods:
 - Zugriff auf Host Dateisystem
 - Zugriff auf Host Netzwerk
 - Starten als spezifischer User, bzw Root
 - Setzen von SELinux context
 - Erweiterte Möglichkeiten mit Linux-Gruppen
 - Erlauben bestimmter Linux Capabilities

Im Prinzip nichts was man möchten - Prüfen!

- oc get scc
- oc describe scc

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

Etcd Encryption

- default: nicht verschlüsselt
- verschlüsselt Secrets, ConfigMaps, Routes, OAuth Access Tokens, OAuth Authorize Tokens
- verschlüsselt nur die Values und keine Keys

```
1  apiVersion: config.openshift.io/v1
2  kind: APIServer
3  metadata:
4    name: cluster
5  spec:
6    audit:
7      profile: Default
8    encryption:
9      type: aescbc
```

Übung

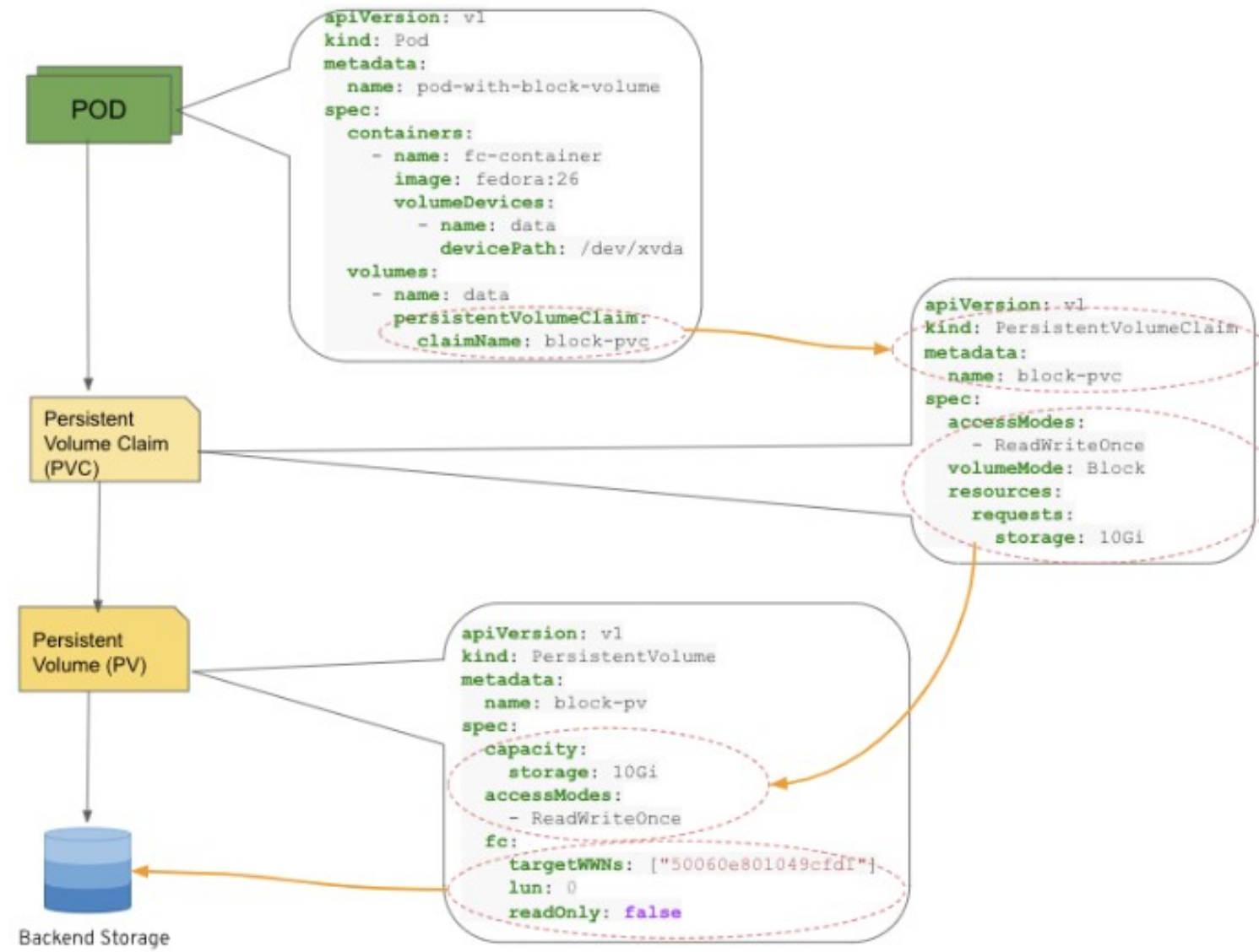
- etcd Encryption einrichten

Doku: <https://docs.openshift.com/container-platform/4.13/security/encrypting-etcd.html>

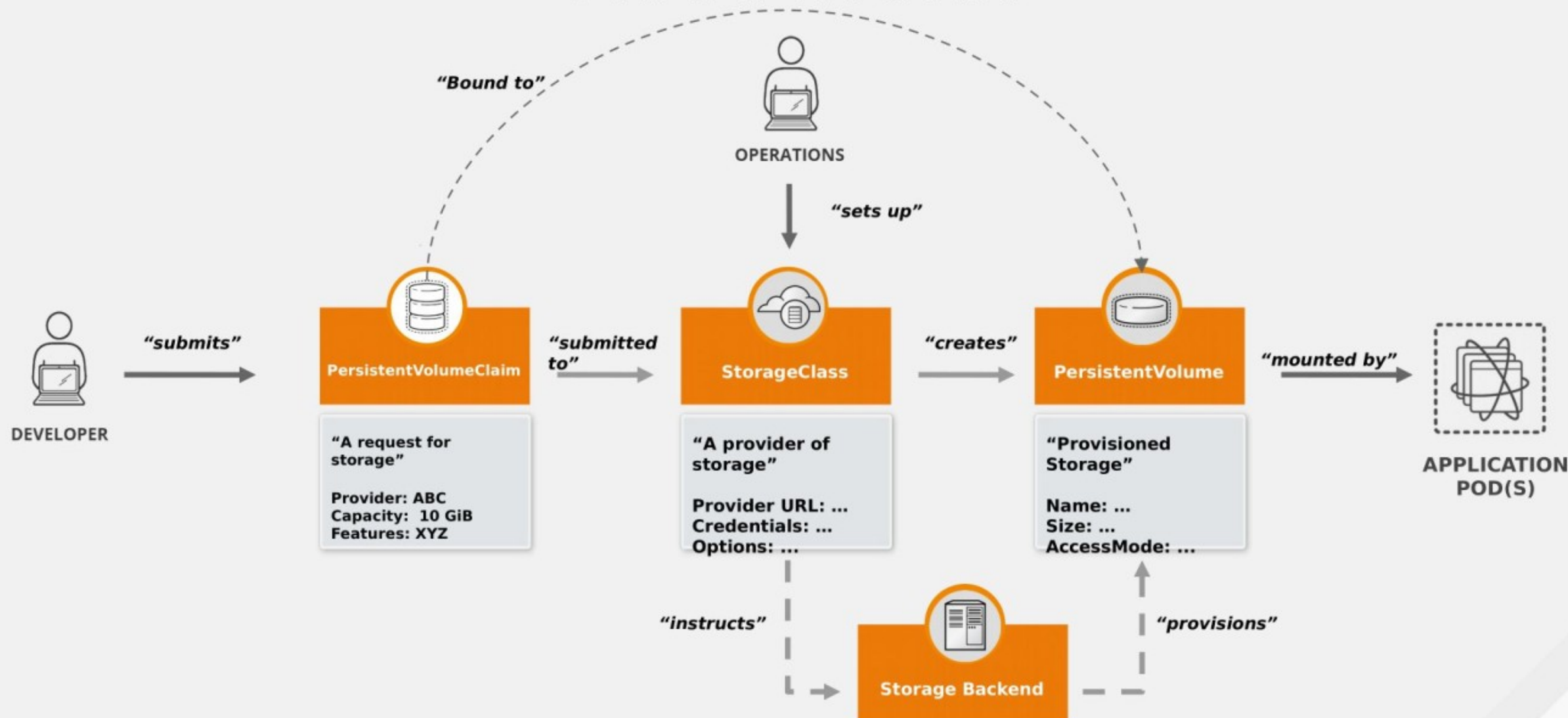
Persistent Storage

Storage Architektur Komponenten

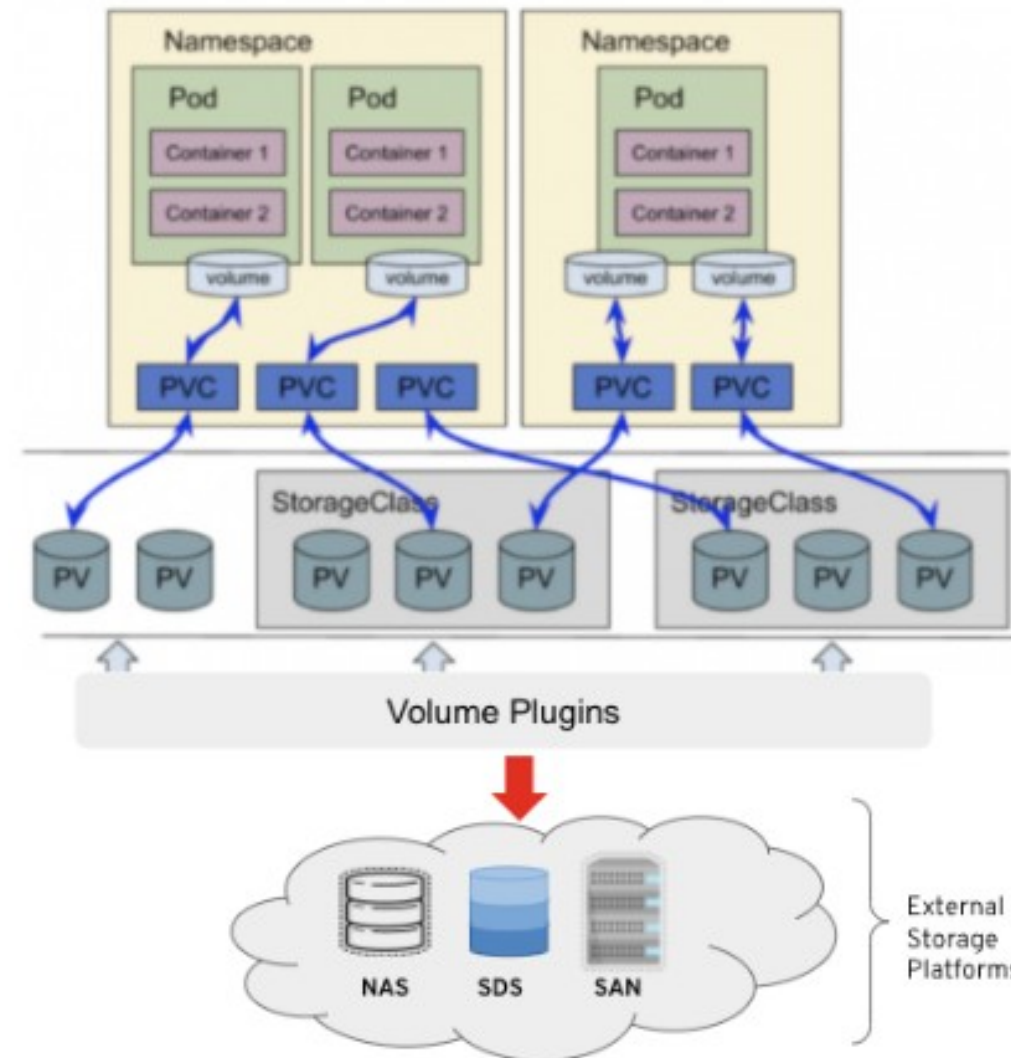
- **Storage Class**
 - HighLevel Beschreibung der Storage Parameter
 - verschiedene Arten von Storage
 - Provisioner Attribut
 - kein Built-In Provisioner für alle Storage Technologien
- **Persistent Volume**
 - zuweisbares Storage Volume
 - Mount auf dem Node → mount in Pods
- **Persistent Volume Claim**
 - User erstellt mit High Level Kriterien:
 - Storage Size
 - Storage Class
 - Access Mode
 - bei Übereinstimmung → Bind



OPENSIFT PERSISTENT STORAGE FRAMEWORK



Volume Plugins



Persistent Storage Provider

- Local
 - HostPath
 - EmptyDir
 - LVM
- NFS
- iSCSI
- Red Hat OpenShift Data Foundation
- und weitere -> Übersicht in der Doku

https://docs.openshift.com/container-platform/4.13/storage/persistent_storage/persistent-storage-ocs.html

Access Modes

Access Mode	CLI abbreviation	Description
ReadWriteOnce	RWO	The volume can be mounted as read-write by a single node.
ReadOnlyMany	ROX	The volume can be mounted as read-only by many nodes.
ReadWriteMany	RWX	The volume can be mounted as read-write by many nodes.

Volume plugin	ReadWriteOnce ^[1]	ReadOnlyMany	ReadWriteMany
AliCloud Disk	✓	-	-
AWS EBS ^[2]	✓	-	-
AWS EFS	✓	✓	✓
Azure File	✓	✓	✓
Azure Disk	✓	-	-
Cinder	✓	-	-
Fibre Channel	✓	✓	-
GCP Persistent Disk	✓	-	-
GCP Filestore	✓	✓	✓
HostPath	✓	-	-
IBM Power Virtual Server Disk	✓	✓	✓
IBM VPC Disk	✓	-	-
iSCSI	✓	✓	-
Local volume	✓	-	-
NFS	✓	✓	✓
OpenStack Manila	-	-	✓
Red Hat OpenShift Data Foundation	✓	-	✓
VMware vSphere	✓	-	✓ ^[3]

Übung

- erstelle einen neuen PVC (1Gi) - wie ist der Status und warum?
- nutze diesen Claim in einem Pod
- deploye eine neue StorageClass mit dem BindingMode Immediate
- teste mit einem PVC

Networking

- alle Features und Capabilities um Netzwerk Traffic zu managen im Stack
- Network Plugins: OVN-Kubernetes (default), OpenShift SDN
- zertifizierte Vendor Plugins: Cisco, Isovalent, Juniper, Tigera, VMware

OVN-Kubernetes - Networking Plugins

Feature	OVN-Kubernetes	OpenShift SDN
Egress IPs	Supported	Supported
Egress firewall ^[1]	Supported	Supported
Egress router	Supported ^[2]	Supported
Hybrid networking	Supported	Not supported
IPsec encryption for intra-cluster communication	Supported	Not supported
IPv6	Supported ^[3] ^[4]	Not supported
Kubernetes network policy	Supported	Supported
Kubernetes network policy logs	Supported	Not supported
Hardware offloading	Supported	Not supported
Multicast	Supported	Supported

Network Policies

- Default: Offen!
- Pod-Level Firewall
- Projekt Admin
- Default: Allow all
- Sobald eine NetworkPolicy zutrifft, ändert sich der Default
- host network nicht betroffen
- additiv

Beispiele

```
1  apiVersion: networking.k8s.io/v1
2  kind: NetworkPolicy
3  metadata:
4    name: web-allow-production
5    namespace: default
6  spec:
7    podSelector:
8      matchLabels:
9        app: web
10   ingress:
11     - from:
12         - namespaceSelector:
13             matchLabels:
14               env: production
15
```

```
1  apiVersion: networking.k8s.io/v1
2  kind: NetworkPolicy
3  metadata:
4    name: api-allow-http-and-https
5    namespace: default
6  spec:
7    podSelector:
8      matchLabels:
9        app: api
10   ingress:
11     - from:
12         - podSelector:
13             matchLabels:
14               role: monitoring
15         ports:
16           - protocol: TCP
17             port: 80
18           - protocol: TCP
19             port: 443
20
```

oc describe networkpolicies

```
Name:          testpol
Namespace:     test
Created on:    2023-12-06 07:19:14 +0000 UTC
Labels:       <none>
Annotations:  <none>
Spec:
  PodSelector:  <none> (Allowing the specific traffic to all pods in this namespace)
  Allowing ingress traffic:
    To Port: <any> (traffic allowed to all ports)
    From:
      PodSelector: <none>
  Not affecting egress traffic
  Policy Types: Ingress
```

Egress Firewall

- ausgehender Traffic, muss den Cluster verlassen
- eine Firewall pro Projekt (default)
- bis zu 8000 Regeln
- DNS oder CIDR Selector
- ingress Replies betroffen
- host networking nicht betroffen
- first Match

```
apiVersion: k8s.ovn.org/v1
kind: EgressFirewall
metadata:
  name: default
  namespace: <namespace> ❶
spec:
  egress:
    - to:
        cidrSelector: <api_server_address_range> ❷
        type: Allow
# ...
    - to:
        cidrSelector: 0.0.0.0/0 ❸
        type: Deny
```

Egress IP

- Maskiert Traffic eines Namespaces mit einer oder mehrerer IPs
- darf keine primäre IP eines Nodes sein
- Nützlich für Firewall Freigaben
- IPs werden (zufällig) auf Nodes verteilt
- der Traffic wird dann über den Node geroutet der die IP assigned hat

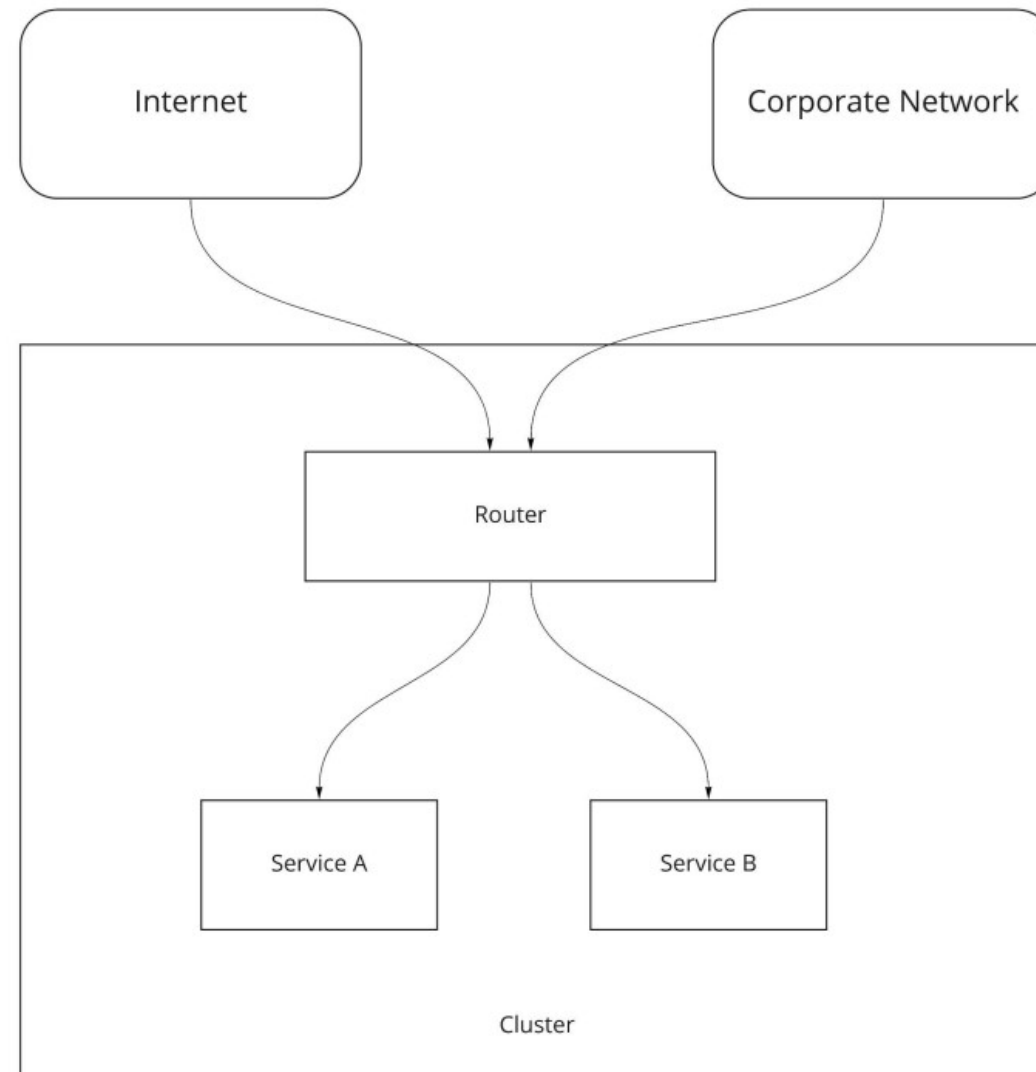
```
apiVersion: k8s.ovn.org/v1
kind: EgressIP
metadata:
  name: egressips-prod
spec:
  egressIPs:
    - 192.168.126.10
    - 192.168.126.102
  namespaceSelector:
    matchLabels:
      env: prod
status:
  items:
    - node: node1
      egressIP: 192.168.126.10
    - node: node3
      egressIP: 192.168.126.102
```

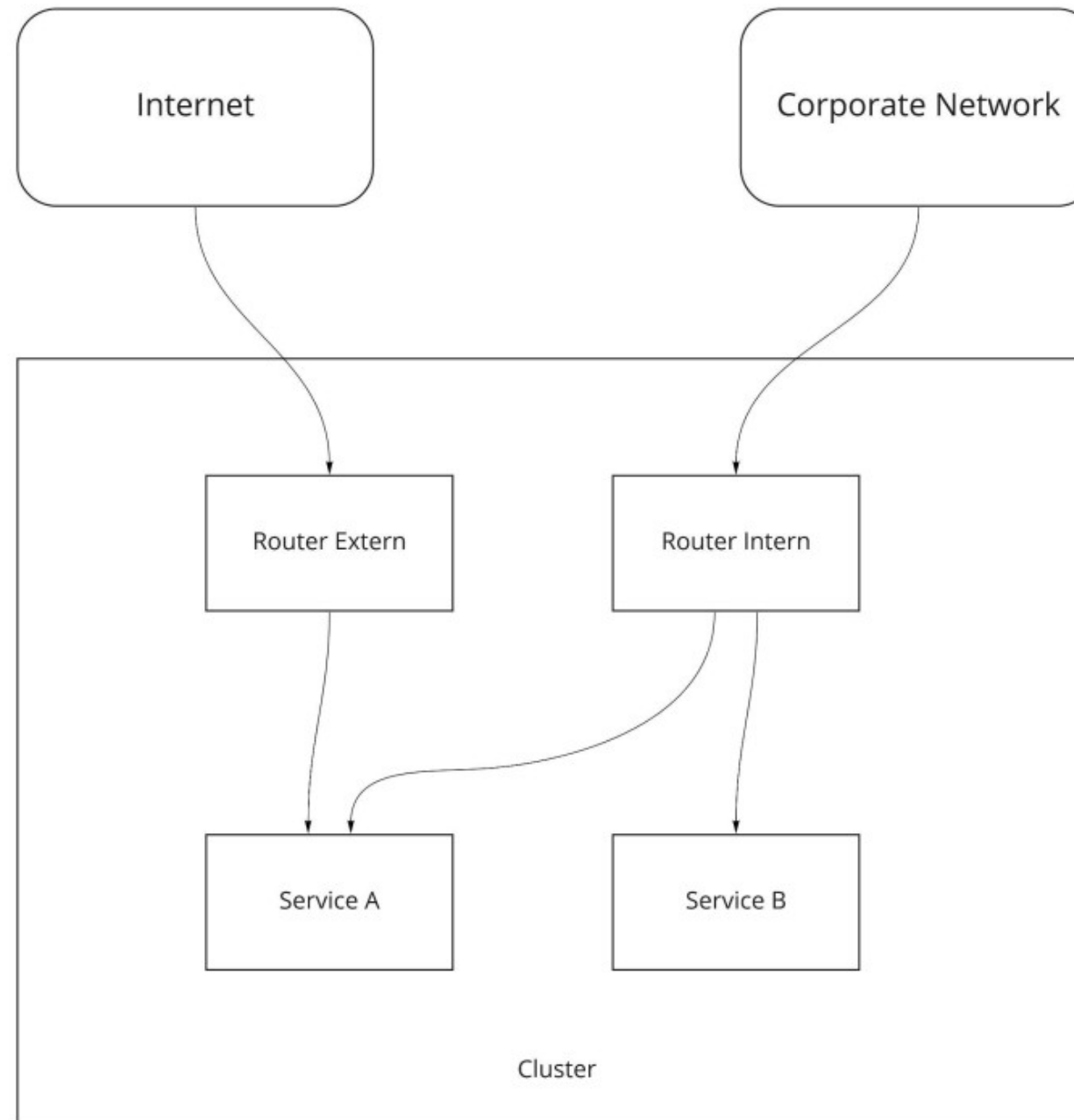
Übung

- Erstelle ein Set von NetworkPolicies die folgenden Ingress Traffic erlaubt:
 - default: deny
 - jeder Traffic innerhalb des Namespaces
 - eingehender Traffic vom Ingress Router
 - eingehender Traffic vom Monitoring
- Erstelle eine EgressFirewall
 - default: deny
 - allow: 10.0.0.0/8

https://docs.openshift.com/container-platform/4.13/networking/network_policy/about-network-policy.html

Ingress





Router Sharding

- hohe Last auf dem Ingress Router (single Instance)
- Routen über verschiedene Router / RouterSets
- verschiedene Routen Konfigurationen / Features in verschiedenen Routern
- zusätzliche IngressController
- Ingress Label: namespace selector, route selector oder beides

Beispiel

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: external-traffic
  namespace: openshift-ingress-operator
spec:
  domain: domain.example.com
  endpointPublishingStrategy:
    type: NodePortService
  nodePlacement:
    nodeSelector:
      matchLabels:
        node-role.kubernetes.io/infra: ""
  tolerations:
    - key: node-role.kubernetes.io/infra
      value: reserved
      effect: NoSchedule
    - key: node-role.kubernetes.io/infra
      value: reserved
      effect: NoExecute
  routeSelector:
    matchLabels:
      type: external
  namespaceSelector:
    matchLabels:
      traffic: external
```

Übung

- Deploye einen weiteren Router der nur Routen mit dem Label „extern=true“ annimmt
- optional zusätzlich nur aus Namespaces mit dem Label „traffic=extern“

Backup und Restore

Backup etcd

- kompletter Cluster State ist im etcd
- etcd Backup Script auf Master Nodes (etcdctl snapshot wrapper)
- Man braucht nur ein etcd node Backup
- nicht in den ersten 24h (Certificate Rotation)
- Cluster Usage beachten (snapshot I/O)
- nach einem Upgrade (Version Match)

Documentation: https://docs.openshift.com/container-platform/4.13/backup_and_restore/control_plane_backup_and_restore/backing-up-etcd.html

Restore

Risikobehaftet

- setzt den Cluster in der Zeit zurück - Konflikte verschiedenster Komponenten
- nicht wenn der etcd erreichbar ist (API Requests)
- Cluster kann sogar Zustände von Workloads, Volumes etc. verlieren
- etcd Backup niemals in neuen Cluster zurückspielen

Documentation: https://docs.openshift.com/container-platform/4.13/backup_and_restore/control_plane_backup_and_restore/disaster_recovery/scenario-2-restoring-cluster-state.html#dr-restoring-cluster-state

Alternative Backup Methoden

- Snapshots der Master Nodes
- Snapshot aller Nodes
- Automatisierung + CI/CD Pipeline
- kommerzielle Lösungen (z.B. Velero)

Best Practices

Noch Fakten?

- Traffic zwischen Cluster Komponenten und der API ist verschlüsselt
- Traffic zwischen Application Pods ist nicht verschlüsselt
- Der Cluster läuft in UTC
- Container laufen by default in UTC
- Container laufen normalerweise mit User ID > 1M

Cluster Konfiguration

- self-provisioner Rechte wegnehmen
- Infrastructure Nodes
- nicht mit cluster-admin Rechten übertreiben (mal schnell)
- Node Capacity Reservation (KubletConfig)
- etcd encryption

Und weiter?

- Deutscher OpenShift Slack Channel: openshift-de.slack.com
- OpenShift Anwender: openshift-anwender.de
- OpenShift Blog: blog.openshift.com
- RedHat Developer Sandbox: developers.redhat.com/developer-sandbox
- learning: learn.openshift.com