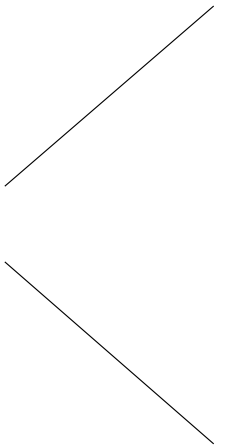


- Openshift
Orchestrierungsservice zur Bereitstellung, Verwaltung und Skalierung von Container-Anwendungen
- Deklaratives System
Status wird in Ressourcen (YAML/JSON) definiert und durch Controller hergestellt
IaC – Infrastructure as Code (<https://blog.nelhage.com/post/declarative-configuration-management>)

```
$ oc api-resources -o name --sort-by=name
alertmanagers.monitoring.coreos.com
apiservers.config.openshift.io
apiservices.apiregistration.k8s.io
appliedclusterresourcequotas.quota.openshift.io
authentications.config.openshift.io
authentications.operator.openshift.io
baremetalhosts.metal3.io
bindings
brokertemplateinstances.template.openshift.io
buildconfigs.build.openshift.io
builds.build.openshift.io
builds.config.openshift.io
catalogsources.operators.coreos.com
certificatesigningrequests.certificates.k8s.io
cloudcredentials.operator.openshift.io
clusterautoscalers.autoscaling.openshift.io
clusternetworks.network.openshift.io
clusteroperators.config.openshift.io
...
```



Pod
Replicaset
Deployment
Service (svc)
Route
PersistentVolumeClaim
Secrets
Configmaps

Imagestream
BuildConfig

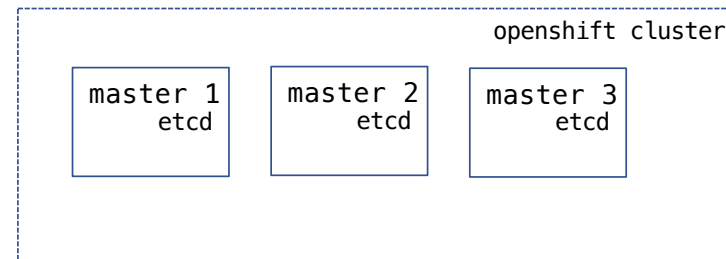
Node
PersistentVolume

Operator
CustomResourceDefinition

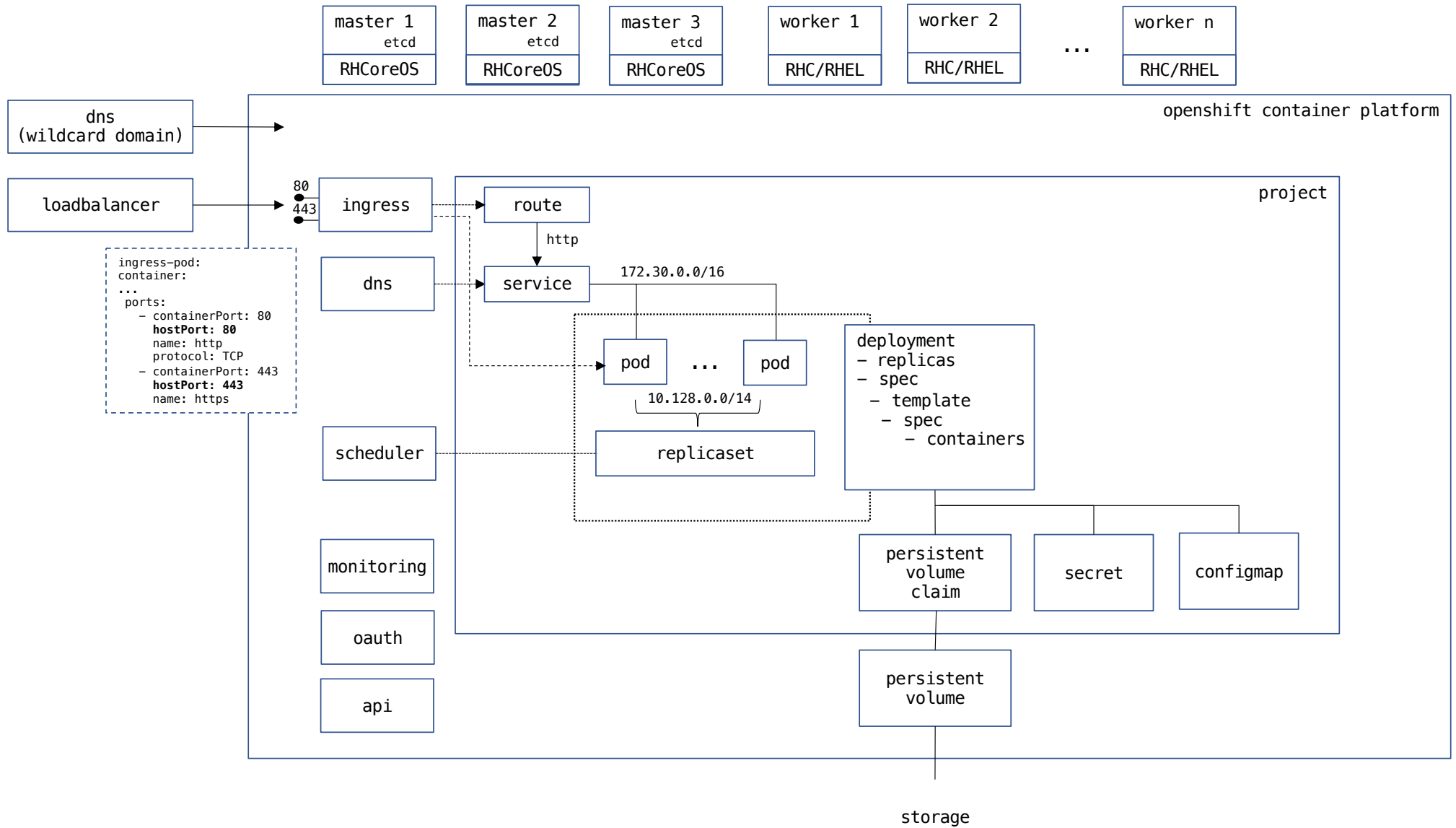
Openshift Resources (Manifest)

```
apiVersion: v1
kind: < Resource Type >
metadata:
  name: <name>
  namespace: <namespace>
  annotations:
    ...
  labels:
    app: <application-name>
    ...
spec:
  ...
  selector:
    <key>: <value>
    ...
status:
  ...
```

oc create



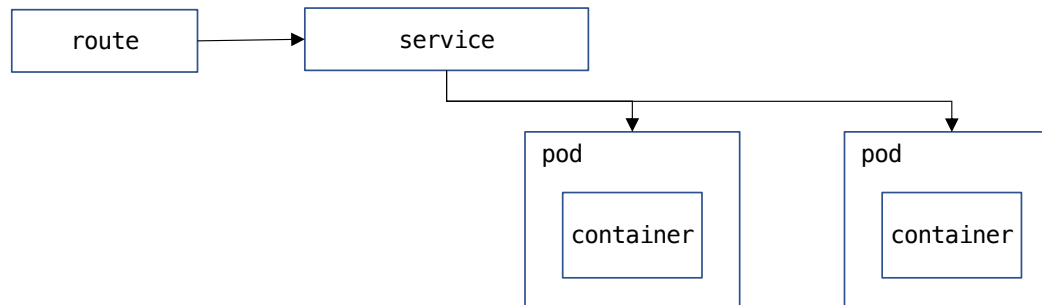
```
apiVersion: v1
kind: Pod
metadata:
  name: webserver
  namespace: do180
  labels:
    app: webserver
spec:
  containers:
    - image: quay.io/danielstraub/webserver:do180
      imagePullPolicy: Always
      ports:
        - containerPort: 8080
          protocol: TCP
    ...
```



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wildfly
  labels:
    app: wildfly
spec:
  replicas: 2
  selector:
    matchLabels:
      app: wildfly
  template:
    metadata:
      labels:
        app: wildfly
    spec:
      containers:
        - name: wildfly
          image: quay.io/do288/wildfly:latest
          ports:
            - containerPort: 8080
              protocol: TCP
```

```
apiVersion: v1
kind: Service
metadata:
  name: wildfly
  labels:
    app: wildfly
spec:
  type: ClusterIP
  selector:
    app: wildfly
  ports:
    - name: http
      protocol: TCP
      port: 8080
      targetPort: 8080
```

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: wildfly
  labels:
    app: wildfly
spec:
  host: sample.apps.eu46.prod.nextcle.com
  to:
    kind: Service
    name: wildfly
  tls:
    termination: edge
```



Declarative :

```
$ ls
deployment.yml route.yml service.yml

$ oc apply -f .
deployment.apps/wildfly created
route.route.openshift.io/wildfly created
service/wildfly created
```

Imperative :

```
$ oc new-app <container-image | git-repository>
--> Found container image 9a9e908 (9 days old) from quay.io for "quay.io/do288/wildfly"

    * An image stream tag will be created as "wildfly:latest" that will track this image

--> Creating resources ...
    imagestream.image.openshift.io "wildfly" created
    deployment.apps "wildfly" created
    service "wildfly" created
--> Success
```

- `oc login -u <user> -p <password> <api-server-url>`
- `oc new-project <name>`
- `oc create -f <resource-yml>`
`oc apply -f <resource-yml>`
- `oc status`
- `oc get <resource-type> [<resource-name>]`
 - `oc get pods`
 - `oc get deployment`
 - `oc get svc <service>`
 - `oc get events`
- `oc describe <resource-type> <resource-name>`
- `oc expose svc <service-name>`
- `oc logs <podname>`
- `oc exec -it <podname> -- <program>`
- `oc rsh <podname>`
- `oc cp <pod>:<location> <location>`
- `oc port-forward <podname> <local-port>:<remote-port>`
- `oc new-app <☺anything☺>`
- `oc delete <resource-type> <resource-name>`
- `oc rollout latest deployment <deployment-name>`

https://docs.openshift.com/container-platform/4.15/cli_reference/openshift_cli/developer-cli-commands.html

```
$ oc new-app --help
Create a new application by specifying source code, templates, and/or images
```

...

Usage:

```
oc new-app (IMAGE | CONTAINERFILE | SOURCE | TEMPLATE | ...) [flags]
```

Beispiele:

```
$ oc new-app quay.io/do288/nginx --name nginx
```

└──────────┘

Container-Image

```
$ oc new-app php:7.3~https://github.com/.../php-hello
```

└──┘ └──────────────────────────┘

Builder-Image
(s2i)

Git-Projekt (Source)

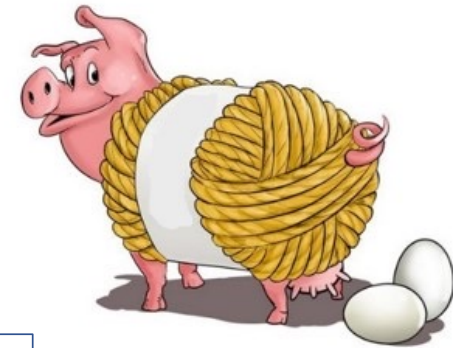


Deployment

Service

Imagestream

BuildConfig



```
$ oc create deployment --image=quay.io/danielstraub/toolbox -o yaml toolbox -- bash -c 'sleep infitity'
```

```
apiVersion: apps/v1
```

```
kind: Deployment
```

```
metadata:
```

```
  name: toolbox
```

```
  labels:
```

```
    app: toolbox
```

```
spec:
```

```
  replicas: 1
```

```
  selector:
```

```
    matchLabels:
```

```
      app: toolbox
```

```
  template:
```

```
    metadata:
```

```
    labels:
```

```
      app: toolbox
```

```
    spec:
```

```
      containers:
```

```
        - command:
```

```
          - bash
```

```
          - -c
```

```
          - sleep infitity
```

```
          image: quay.io/danielstraub/toolbox
```

```
          name: toolbox
```



```
$ oc create service clusterip webserver --tcp=80:8080 -o yaml
```

```
apiVersion: v1
kind: Service
metadata:
  name: webserver
  labels:
    app: webserver
spec:
  ports:
    - name: 80-8080
      port: 80
      protocol: TCP
      targetPort: 8080
  selector:
    app: webserver
  type: ClusterIP
```

```
$ oc create route edge --hostname do180.<wildcard-domain> --service webserver --insecure-policy=Redirect webserver -o yaml
```

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: webserver
  labels:
    app: webserver
spec:
  host: do180.apps.eu410.prod.nextcle.com
  port:
    targetPort: http
  tls:
    insecureEdgeTerminationPolicy: Redirect
    termination: edge
  to:
    name: webserver
```

```
oc create route -help
```

Available Commands:

edge	Create a route that uses edge TLS termination
passthrough	Create a route that uses passthrough TLS termination
reencrypt	Create a route that uses reencrypt TLS termination

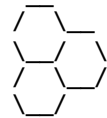
odo innerLoop:

```
odo dev → build run  
odo dev -debug → build debug
```

verwenden eines 'nicht-Default' Commando:

```
odo dev --buildCommand=... --runCommand=
```

```
workstation:demo $ odo dev [ --no-watch ] [ --platform podman ]
```



Developing using the "java-springboot" Devfile
Namespace: architecture-setup
odo version: v3.15.0 (10b5e8a8f)

↳ Running on the cluster in Dev mode

- Waiting for Kubernetes resources ...
- ✓ Added storage m2 to component

=====

△ Pod is Pending

=====

- ✓ Pod is Running
- ✓ Syncing files into the container [801ms]
- ✓ Building your application in container (command: build) [1m]
- Executing the application (command: run) ...

↳ Dev mode

Web console accessible at <http://localhost:20000/>

Keyboard Commands:

- [Ctrl+c] - Exit and delete resources from the cluster
- [p] - Manually apply local changes to the application on the cluster
- ✓ Finished executing the application (command: run) [7s]

odo outerLoop:

```
odo deploy → deploy (composite-Commando)
```

devfile.yaml

```
metaData:
  ...

components:
- container:
  image: ...
  command: ...
  env: ...

- kubernetes
  uri: deployment.yaml

- image
  uri: Dockerfile

- volume

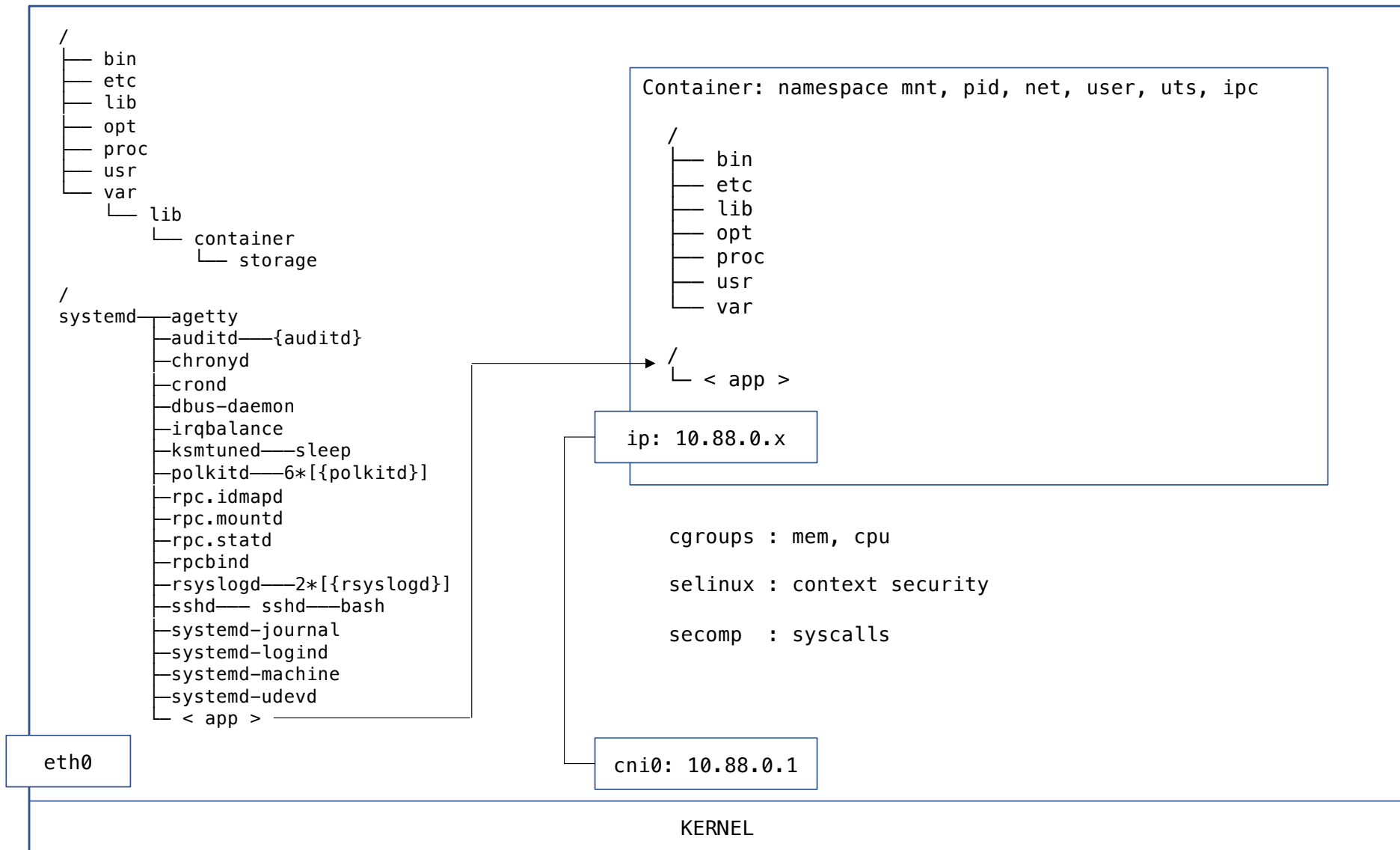
commands:
- exec
  component: ...
  commandLine: ..

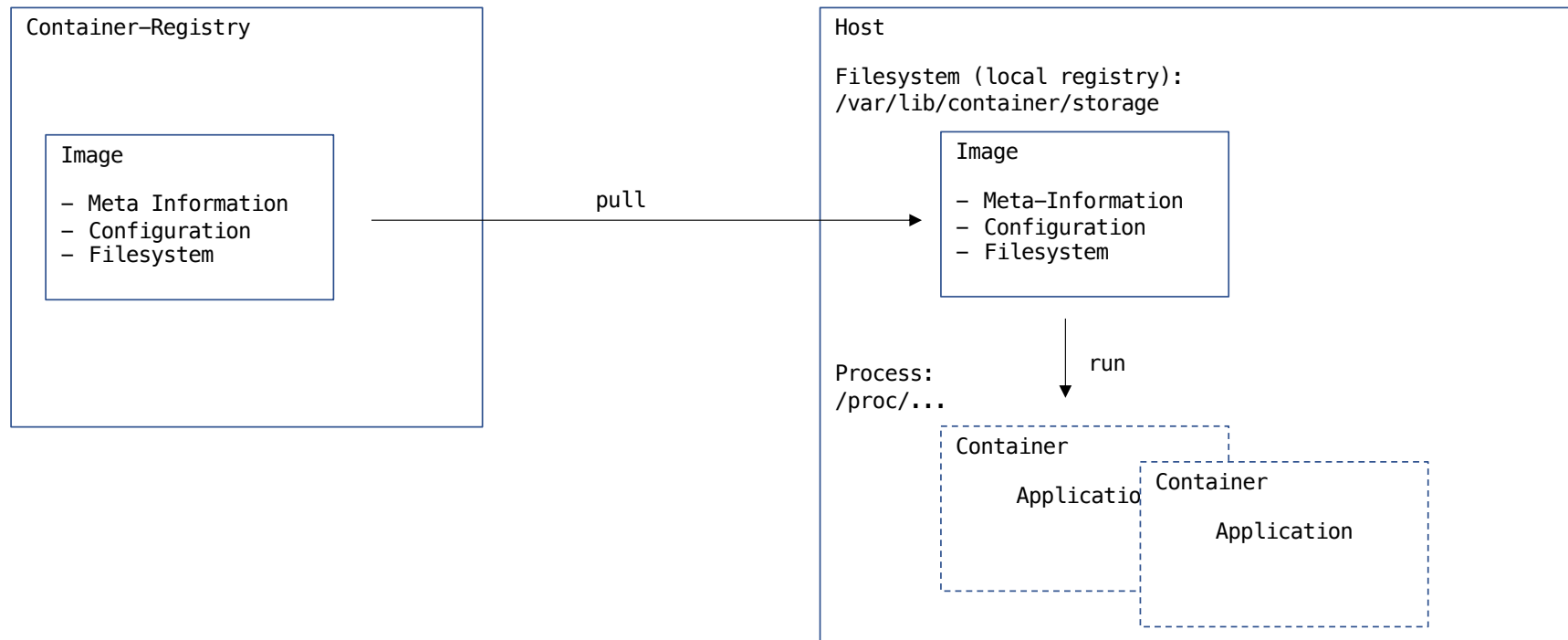
- apply
  component: ...

- composite
  commands:
  - ...
```

Kommando-Gruppen/Lifecycle

```
group:
  kind: build|run|test|debug|deploy
  isDefault: true|false
```





podman build - Containerfile

```
FROM ubi9/ubi
LABEL version=.... maintainer=
MAINTAINER daniel
ENV key=value
ARG version
```

```
ADD http://repos/app-$version.tar /opt/app/
COPY webapp.war /opt/tomcat/webapps
RUN yum install -y tomcat && \
    useradd tomcat && \
    chown -R 1000:1000 /opt/tomcat && \
    yum cleanup && rm /tmp/*
```

```
USER 1000
WORKDIR /opt/tomcat
VOLUMES /opt/tomcat/logs
EXPOSE [ 8080, 8001 ]
ENTRYPOINT [ "bin/sh -c" ]
CMD [ "bin/catalina.sh", "start" ]
```

```
podman build -t my-tomcat . <- Build-Dir
```

```
podman push <registry>/<name>:<tag>
```

Layer

Config

Layer

...

Layer

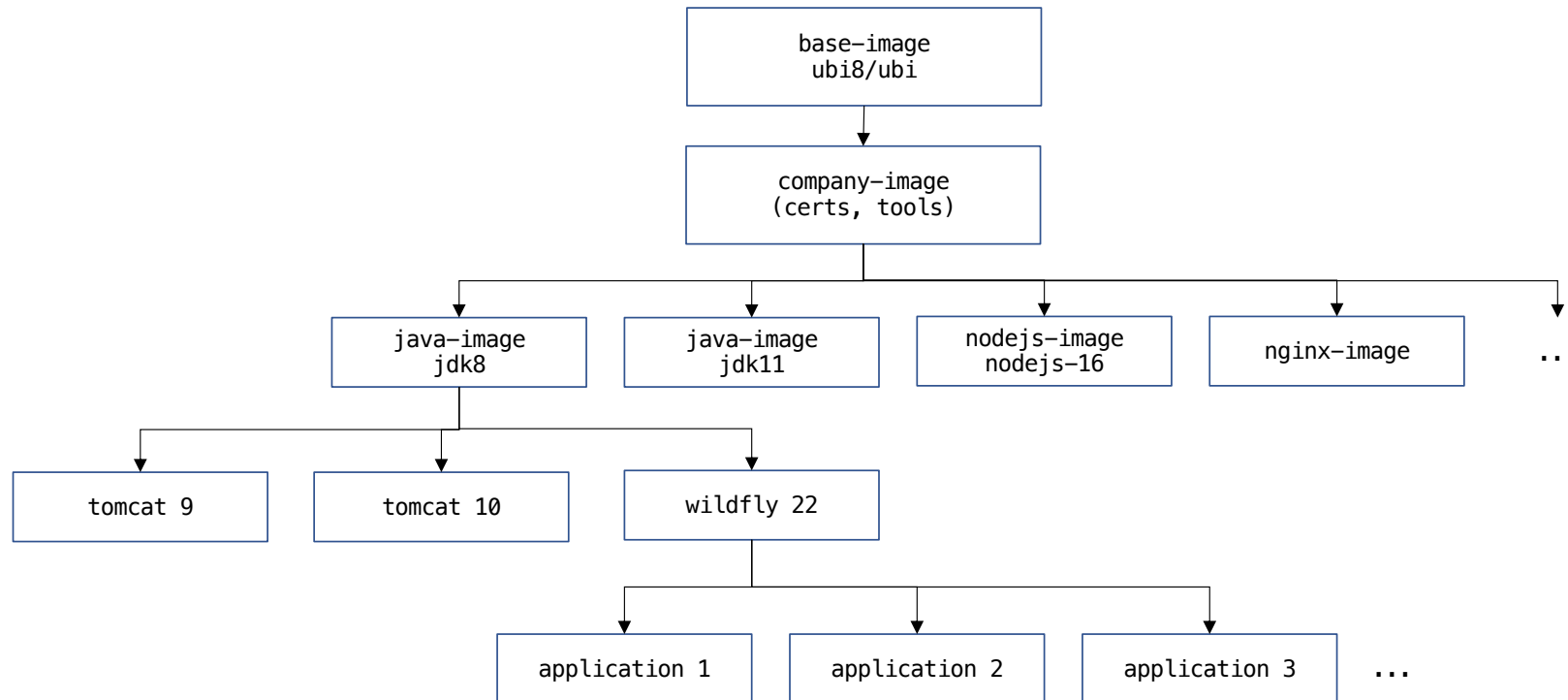
Config

Manifest

local container storage

remote registry

Beispiel: Image – Vererbung



Änderungen an einem Basis-Image erfordern Rebuild der davon abhängigen Images !

Container in Openshift:

- beliebige User-Id RUN chmod - R 0770
- Group-Id 0 (root) RUN chgrp -R 0
- Ports > 1024

```
apiVersion: project.openshift.io/v1
kind: Project
metadata:
  annotations:
    openshift.io/sa.scc.mcs: s0:c26,c15
    openshift.io/sa.scc.supplemental-groups: 1000680000/10000
    openshift.io/sa.scc.uid-range: 1000680000/10000
```

```
# oc exec pgadmin-778c479f79-tfbqn -- id
uid=1000680000(1000680000) gid=0(root) groups=0(root),1000680000
```

NFS-Mount →

```
# ls -al /mnt/nfs/apps/pgadmin
-rw-r--r-- 1 1000680000 root 124K Nov 27 01:03 access_log
-rw-r--r-- 1 1000680000 root 853 Nov 27 00:44 config_local.py
-rw-r--r-- 1 1000680000 root 1.2K Nov 27 00:46 error_log
```

<https://cloud.redhat.com/blog/a-guide-to-openshift-and-uids>

Abweichende User-Id : Serviceaccount mit Security Context Constraint 'anyuid' notwendig :

```
apiVersion:
rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: scc-anyuid
rules:
- apiGroups:
  - security.openshift.io
  resourceNames:
  - anyuid
  resources:
  - securitycontextconstraints
  verbs:
  - use
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: gitea:anyuid
  namespace: apps
roleRef:
  kind: ClusterRole
  name: scc-anyuid
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: ServiceAccount
  name: gitea
  namespace: apps
```

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: gitea
  namespace: apps
```

erstellt von Cluster-Administrator !

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gitea
  namespace: apps
...
spec:
  template:
    spec:
      serviceAccountName: gitea
  ...
```

```
# oc exec gitea-7dcdc5c445-w9qmv -- id
uid=65534(nobody) gid=65534(nobody) groups=65534(nobody),0(root)

# ll /mnt/nfs/repos/ds
drwxr-xr-x 7 nobody nobody 119 Nov 26 16:57 admin.git/
drwxr-xr-x 7 nobody nobody 119 Nov 26 16:12 calibre.git/
drwxr-xr-x 7 nobody nobody 119 Nov 17 16:02 gitea.git/
...
```

UserId aus Container-Config !

Container Registry:

Red Hat → <https://access.redhat.com/RegistryAuthentication>

```
# podman login quay.io
Username: ...
Password: ...
Login Succeeded!    -> /run/user/<user-id>/containers/auth.json
```

```
# podman push --creds <username>:<password> ...
```

```
# skopeo --help
Various operations with container images and container image registries
```

Usage:
 skopeo [command]

Available Commands:

copy	Copy an IMAGE-NAME from one location to another
delete	Delete image IMAGE-NAME
help	Help about any command
inspect	Inspect image IMAGE-NAME
list-tags	List tags in the transport/repository specified by the REPOSITORY-NAME
login	Login to a container registry
logout	Logout of a container registry
manifest-digest	Compute a manifest digest of a file
standalone-sign	Create a signature using local files
standalone-verify	Verify a signature using local files
sync	Synchronize one or more images from one location to another

```
skopeo copy --format ... --dest-creds <user>:<password> containers-storage:localhost/webserver docker://quay.io/do288/webserver
```

Verwenden einer externen Container Registry - Authentifizierung

```
$ oc get serviceaccounts
NAME          SECRETS
default       2
...
```

```
$ oc create secret docker-registry <secret-name> --docker-server <registry> --docker-username <user> --docker-password <password>
```

```
$ oc secrets link <serviceaccount-name> <secret-name> --for pull
```

Verwenden einer externen Container Registry - Secret von auth.json

```
$ oc create secret generic quayio --from-file .dockerconfigjson=/run/user/1000/containers/auth.json --type kubernetes.io/dockerconfigjson
```

```
apiVersion: v1
kind: Secret
metadata:
  name: quayio
type: kubernetes.io/dockerconfigjson
data:
  .dockerconfigjson: ewogICJhdXRocyI6IHsKICAgICJyZWdp3 ...
```

Serviceaccount 'imagePullSecrets' :

```
$ oc secrets link <serviceaccount-name> <secret-name> --for pull
```

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: default
imagePullSecrets:
- name: default-dockercfg-4sdrk
- name: quayio
...
```

oder im Deployment verwenden:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: pgadmin
spec:
  replicas: 1
  template:
    spec:
      imagePullSecrets:
      - name: quayio
      containers:
      - name: pgadmin
        image: registry.connect.redhat.com/crunchydata/crunchy-pgadmin4
```

Imagestream:

- enthält Verweise (Zeiger) auf Images und deren Tags (keine Images)
- Verwendung in Deployment als Image und Trigger
- Import aus externer Registry oder Ergebnis eines Build

```
$ oc import-image webserver --from=quay.io/do288/webserver --confirm ( --scheduled)
$ oc describe is webserver
Name:                               nginx
Unique Images:                       4
Tags:                                4

latest
  updates automatically from registry quay.io/do288/webserver:latest
  * quay.io/do288/nginx@sha256:c34f57431167fca470730b67a1a8636126d2464eee619ec8d0b577c8e63bffe0
1.2
  updates automatically from registry quay.io/do288/webserver:1.2
  * quay.io/do288/nginx@sha256:ee508edacfe0bc1e6af43a15348b400a7d97121507348bd5fb5effb6b9f8d84e
1.1
  updates automatically from registry quay.io/do288/webserver:1.1
  * quay.io/do288/nginx@sha256:674ab485f6e83f162eb4bdaf12986469c7b4f484f65fbb18f3b03218fd5f36e4
1.0
  updates automatically from registry quay.io/do288/webserver:1.0
  * quay.io/do288/nginx@sha256:693b30b107da2661a8636126d2464eee619ec8d0b577c8e63bffe0
```

TAG	LAST MODIFIED ↓	SECURITY SCAN	SIZE	MANIFEST
1.2	40 minutes ago	8 Medium	91.9 MB	SHA256 ee508edacfe0
latest	14 hours ago	8 Medium	91.9 MB	SHA256 c34f57431167
1.1	a day ago	8 Medium	90.6 MB	SHA256 674ab485f6e8
1.0	a day ago	8 Medium	90.6 MB	SHA256 693b30b107da

Verwenden von Imagestreams:

```
$ oc set image-lookup webserver
$ oc describe is webserver | grep Lookup
Image Lookup:          local=true

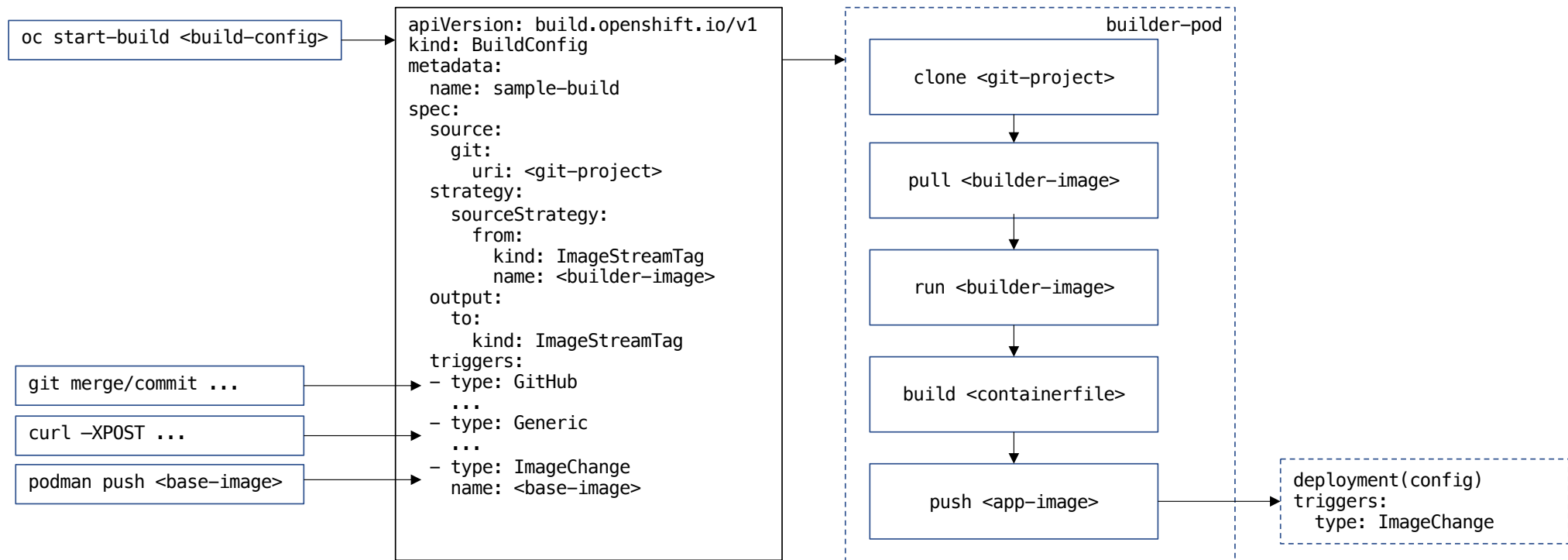
$ oc create deployment webserver --image webserver ← wird ersetzt mit Image-ID

$ oc describe deployments webserver | grep Image:
Image:   quay.io/do288/webserver@sha256:8c5458214997261f09bd0ffdec0552b57c8c0a8737decf07b51b89997a43c3f4
```

Aktualisierung des Deployments bei Änderungen im ImageStreams:

```
$ oc set triggers deployment webserver --from-image webserver:latest --container webserver

$ oc get deployments.apps webserver -o jsonpath='{.metadata.annotations.image\.openshift\.io/triggers}' | jq
[
  {
    "from": {
      "kind": "ImageStreamTag",
      "name": "webserver:latest"
    },
    "fieldPath": "spec.template.spec.containers[?(@.name==\"webserver\")].image"
  }
]
```



Source: binary | dockerfile | git | images

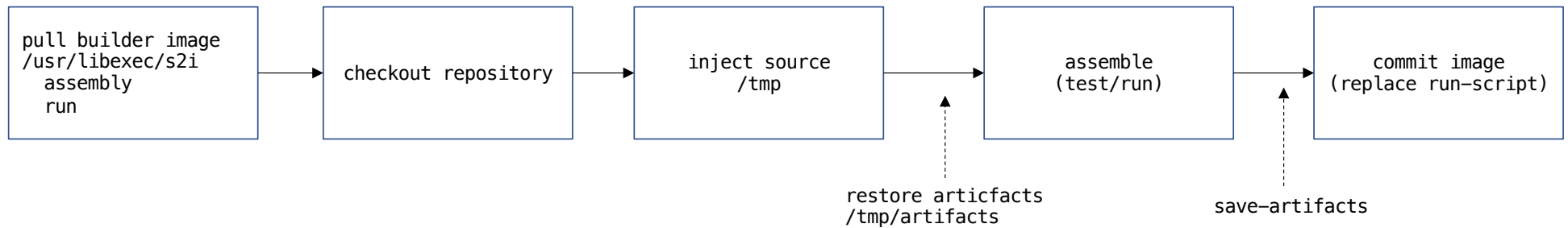
Strategy:

- source : Builder-Image enthält Tools und Logik zum Erstellen einer Anwendung (Source2Image)
- docker : Git-Repository mit Dockerfile

```

strategy:
  dockerStrategy:
    dockerfilePath: Containerfile
    ....

```



Build-Scripte:

- default in /usr/libexec/s2i
- assemble und run sind mandatory
- save-artifacts, usage, test/run sind optional
- können überschrieben werden im Git-Repo .s2i/bin
(Wrapper um Original-Script oder komplett neues Script)

Incremental Builds:

- save-artifacts erstellt TAR
- wird vor dem Ausführen von assembly injected in /tmp/artifacts

Deployment-Strategien

- Rolling Updates : Pods werden der Reihe nach aktualisiert
- Recreate: existierende Pods werden beendet und neue gestartet

DeploymentConfig: DEPRECATED ab 4.15 !

- Pre/Mid/Post – Lifecycle Hooks

Beenden eines Pods:

- SIGTERM: Pod soll keine neuen Verbindungen annehmen und bestehenden Aktionen beenden
- SIGKILL: nach `terminationGracePeriodSeconds` (30s) wird der Pod beendet

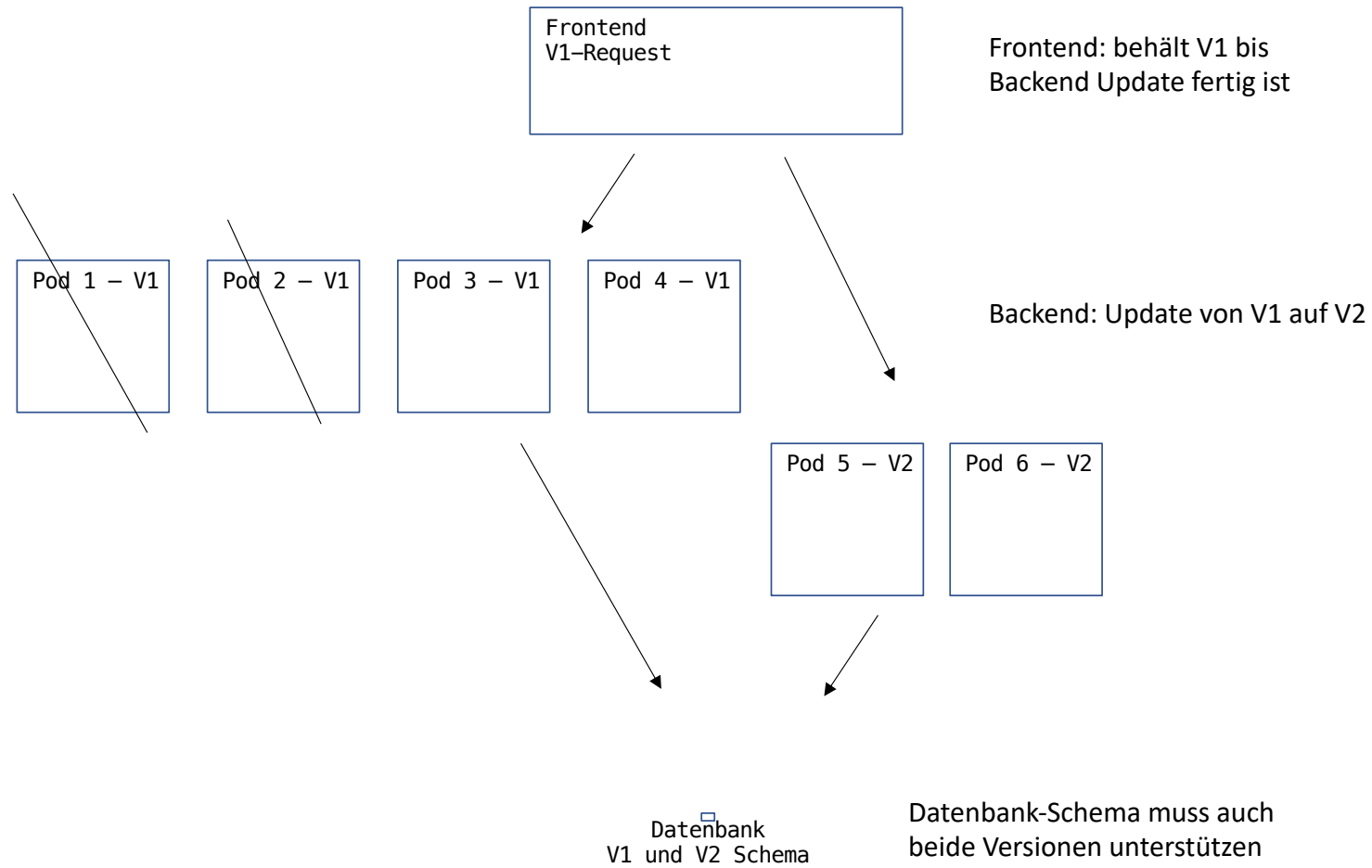
```
kind: Deployment
metadata:
  name: ...
spec:
  revisionHistoryLimit: 3   (default: 10)
  replicas: 4
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1           ← max. 5 Pods aktiv
      maxUnavailable: 1
  ...
  template:
    spec:
      containers:
      - ...
      terminationGracePeriodSeconds: 30
```

```
oc rollout SUBCOMMAND deployment <name>
```

cancel	Cancel the in-progress deployment
history	View rollout history
latest	Start a new rollout for deployment config with latest state
pause	Mark the provided resource as paused
restart	Restart a resource
resume	Resume a paused resource
retry	Retry the latest failed rollout
status	Show the status of the rollout
undo	Undo a previous rollout

```
oc rollback deployment <name> [--to-version=]
```

N-1 Abwärtskompatibilität bei Rolling-Update:



A/B Deployment Strategy:

```
apiVersion: v1
kind: Service
metadata:
  name: service-a
spec:
  ports:
    - name: http
      port: 80
      protocol: TCP
      targetPort: http
  selector:
    app.kubernetes.io/instance: deployment-a
```

```
apiVersion: v1
kind: Service
metadata:
  name: service-b
spec:
  ports:
    - name: http
      port: 80
      protocol: TCP
      targetPort: http
  selector:
    app.kubernetes.io/instance: deployment-b
```

```
kind: Route
metadata:
  name: <name>
spec:
  host: <host>
  to:
    kind: Service
    name: service-a
    weight: 50
  alternateBackends:
    - kind: Service
      name: service-b
      weight: 200
```

Secrets:

- Passwörter, Token, Zertifikate ...
- typisiert: basic-auth, dockerfg, tls, opaque
- Inhalte sind base64-decodiert, nicht verschlüsselt

→ max. Größe 1 MB

→ nur innerhalb eines Project (NS) sichtbar

```
apiVersion: v1
kind: Secret
metadata:
  name: ...
  namespace: ...
data:
  password: MTIzNDU2
type: Opaque
```

```
# echo MTIzNDU2 | base64 -d
123456
```

ConfigMap:

- generische Key-Value Daten

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: ...
  namespace: ...
binaryData:
  keystore: |
    7oAMCAQICCF7Dt6ZDf6TgMA0GCSqGSIb3DQEBAQEI1ZSQUla
    MTEQMA4GA1UECwwHU ...
data:
  HOME: /usr/share/nginx
  default.conf: |
    server {
      listen 8181 default_server;
      server_name _;
      location / {
        root /usr/share/nginx/html;
        index index.html index.htm;
      }
    }
```

```
$ oc create configmap <cm-name> --from-literal F00=BAR
```

```
$ oc create configmap <cm-name> --from-file <path>
```

```
$ oc create secret docker-registry quayio --docker-server quay.io --docker-username <user> --docker-password <password>
```

Secrets: Verwendung als Umgebungs-Variablen

```
apiVersion: v1
kind: Pod
metadata:
  name: secret-env-pod
spec:
  containers:
  - name: mycontainer
    image: redis
    env:
    - name: SECRET_USERNAME
      valueFrom:
        secretKeyRef:
          name: mysecret
          key: username
    - name: SECRET_PASSWORD
      valueFrom:
        secretKeyRef:
          name: mysecret
          key: password
```

```
$ oc set env deployment/<deployment-name> --from cm/<cm-name>
```

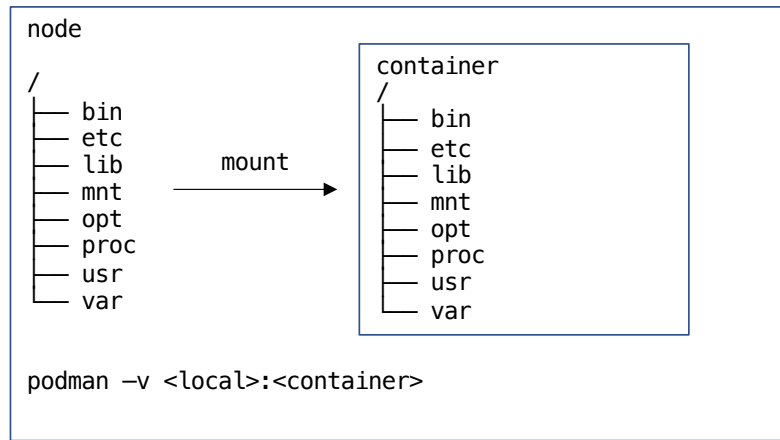
```
$ oc set volume deployment/<deployment-name> --add --type configmap --mount-path /etc/nginx/conf.d --name config --configmap-name <cm-name>
```

ConfigMap: Verwendung als Konfigurations-Dateien

```
apiVersion: apps/v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    container: nginx
    volumeMounts:
    - mountPath: /etc/nginx/conf.d
      name: config
  volumes:
  - name: config
    configMap:
      name: nginx-config
```

```
apiVersion: apps/v1
kind: Pod
metadata:
  name: wildfly-standalone-xml
spec:
  containers:
  - name: wildfly
    container: nginx
    volumeMounts:
    - mountPath: /opt/wildfly/standalone/configuration
      name: standalone-xml
      subPath: standalone.xml
  volumes:
  - name: standalone-xml
    configMap:
      name: standalone-xml
```

Volumes



```
kind: Pod
...
spec:
  containers:
    ...
    volumeMounts:
      - mountPath: <path_container_fs>
        name: <name>
    ...
  volumes:
    - name: <volume>
      <volume-type>:
        <volume-attributes>
```

→ <https://kubernetes.io/docs/concepts/storage/volumes/>

Volume=Types

- emptyDir
- hostPath (system:openshift:scc:hostmount-anyuid !)
- configMap
- secret
- persistentVolumeClaim
- ...

Persistence

Administrator erzeugt PersistentVolume

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfs-data
spec:
  accessModes:
    - ReadWriteMany
  capacity:
    storage: 10Gi
  nfs:
    path: /mnt/nfs/data
    server: 10.0.0.1
  persistentVolumeReclaimPolicy: Retain
```

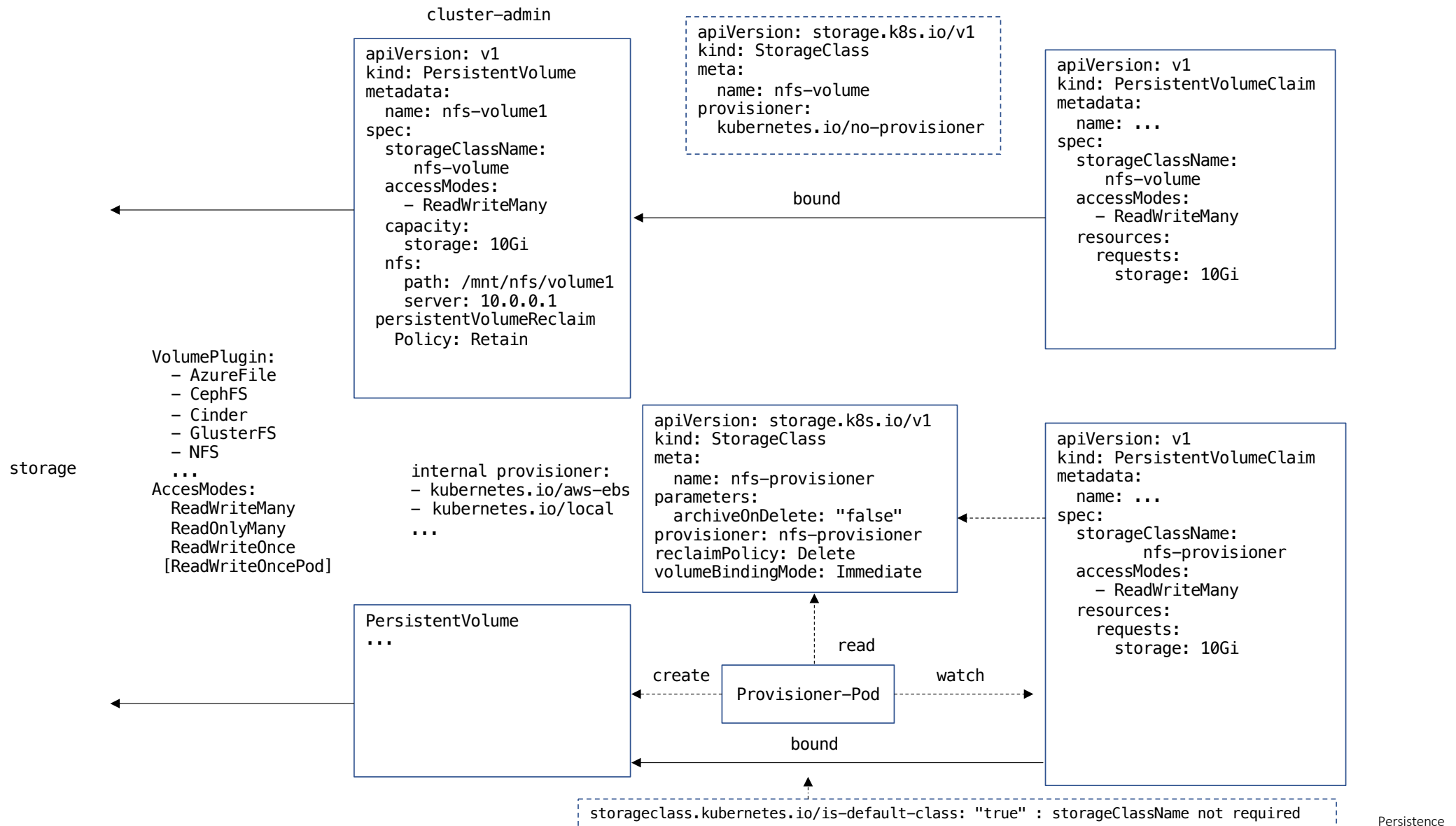
- automatisiertes PV-Management mit storageClass/Provisioner

Anwendung erstellt Anforderung

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: html-data
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
```

und verwendet dieses im Deployment / Pod

```
kind: Deployment
...
  containers:
    - name: webserver
      ...
      volumeMounts:
        - mountPath: /usr/share/nginx/html
          name: html
  volumes:
    - name: html
      persistentVolumeClaim:
        claimName: html-data
```



StatefulSet

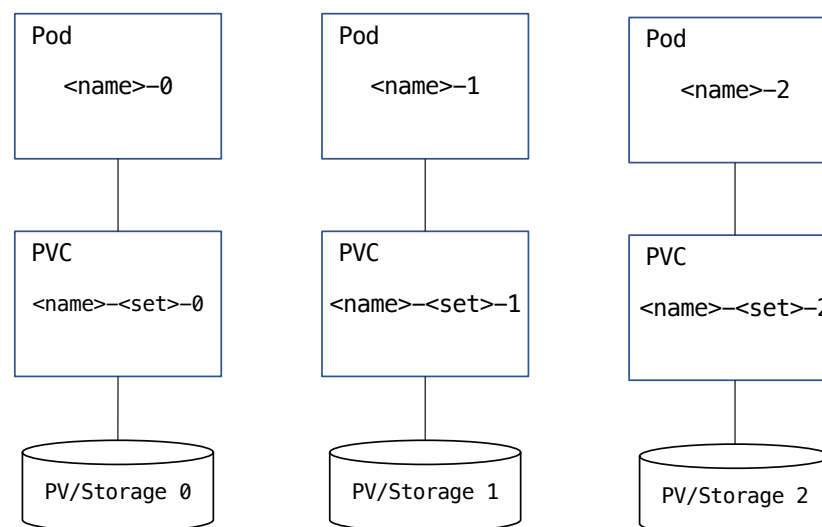
```

apiVersion: v1
kind: Service
metadata:
  name: <svc-name>
spec:
  clusterIP: None
  ports:
  - name:
    ...
  selector:
    app: <name>
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: <name>
spec:
  serviceName: <svc-name>
  replicas: 3
  ...
  template:
    metadata:
      labels:
        app: <name>
    spec:
      containers:
      - name: ...
        ...
        volumeMounts:
        - name: <pvc-name>
          mountPath: ...
  volumeClaimTemplates:
  - metadata:
      name: <pvc-name>
    spec:
      accessModes:
      - ReadWriteMany
      resources:
        requests:
          storage: xxGi

```

Headless-Service (keine Cluster-IP, kein Loadbalancing)

Host-Name/DNS A-Record für jeden Pod: <name-#>.<svc-name>.<project>.svc.cluster.local
CNAME <svc-name>.<project>.svc.cluster.local + SVR-Records für jeden Pod



(compute) Resources :

- Memory: number of bytes (quantity suffixes: E, P, T, G, M, k | Ei, Pi, Ti, Gi, Mi, Ki)
- CPU : millicores (m)

millicores are the fractions of *time* of a single CPU (not the fraction of number of CPUs).

Cgroups, and hence Docker, and hence Kubernetes, doesn't restrict CPU usage by assigning cores to processes (like VMs do), instead it restricts CPU usage by restricting the amount of time (quota over period) the process can run on each CPU (with each CPU taking up to 1000mcpus worth of allowed time).

<https://stackoverflow.com/questions/61851751/multi-threading-with-millicores-in-kubernetes>

```
apiVersion: v1
kind: Pod
metadata:
...
spec:
  containers:
  - name: <name>
    resources:
      requests:
        memory: 64Mi
        cpu: 100m
      limits:
        memory: 128Mi
        cpu: 200m
```

Scheduling

Execution
(cgroups)

```
$ oc describe node master01
...
Allocatable:
  cpu: 3500m
  memory: 15268156Ki
Non-terminated Pods: (60 in total)
  CPU Requests CPU Limits Memory Requests Memory Limits
...
Allocated resources:
Resource      Requests      Limits
-----
cpu           2397m (68%)   0 (0%)
memory        9347Mi (62%)  512Mi (3%)
```

```
$ oc get pods -w
```

NAME	READY	STATUS	RESTARTS	AGE
famousapp-famouschart-65744d4c8b-4zqhn	0/1	Running	0	10s
famousapp-mariadb-0	0/1	ContainerCreating	0	10s
famousapp-mariadb-0	0/1	Running	0	11s
famousapp-famouschart-65744d4c8b-4zqhn	0/1	Running	1	33s
famousapp-famouschart-65744d4c8b-4zqhn	0/1	Error	1	34s
famousapp-famouschart-65744d4c8b-4zqhn	0/1	Running	2	35s
famousapp-famouschart-65744d4c8b-4zqhn	0/1	Error	2	36s
famousapp-famouschart-65744d4c8b-4zqhn	0/1	CrashLoopBackOff	2	37s
famousapp-mariadb-0	1/1	Running	0	48s
famousapp-famouschart-65744d4c8b-4zqhn	0/1	Running	3	56s
famousapp-famouschart-65744d4c8b-4zqhn	1/1	Running	3	62s

```

metadata:
  name: famousapp-mariadb:
...

livenessProbe:
  exec:
    command:
      - /bin/bash
      - -ec
      - |
        password_aux="${MARIADB_ROOT_PASSWORD:-}"
        if [[ -f "${MARIADB_ROOT_PASSWORD_FILE:-}" ]]; then
          password_aux=$(cat "${MARIADB_ROOT_PASSWORD_FILE:-}")
        fi
        mysqladmin status -uroot -p"${password_aux}"

```

```

metadata:
  name: famousapp-famouschart
...

livenessProbe:
  initialDelaySeconds: 30
  httpGet:
    path: /
...
readinessProbe:
  failureThreshold: 3
  httpGet:
    path: /
    port: http
    scheme: HTTP
    periodSeconds: 10
    successThreshold: 1
    timeoutSeconds: 1

```

Liveness / Readiness / Startup Probes

- liveness : Container wird bei negativen Ergebnis neu gestartet `.spec.containers.livenessProbe`
- readiness: Route/Service wird aktiviert/deaktiviert `.spec.containers.readinessProbe`
- startup: liveness/readiness sind deaktiviert bis startup positiv ist
Container wird bei neg. Startup-Probe sofort beendet `.spec.containers.startupProbe`

Probes:

```
exec:
  command:
  - cat
  - /tmp/ready
  initialDelaySeconds: 5
  periodSeconds: 5
```

```
httpGet:
  path: /healthz
  port: healthz-port
  schema: https
  httpHeaders: ...
  failureThreshold: 1
  periodSeconds: 10

200 <= status < 400
```

```
tcpSocket:
  port: 5432
  initialDelaySeconds: 15
  periodSeconds: 20
```

- initialDelaySeconds: Zeitdauer bis zur ersten liveness/readiness Probe
- periodSeconds: Intervall zur Ausführung der Proben (default 10 sec)
- timeoutSeconds: max. Timeout bei einer Probe (default 1 sec)
- successThreshold: Schwellwert ab wann aufeinanderfolgende positive Proben als Erfolg gewertet werden (default 1)
- failureThreshold: Schwellwert ab wann aufeinanderfolgende negative Proben als Ausfall gewertet werden (default 3)

```

kind: Deployment
apiVersion: apps/v1
metadata:
  name: webserver
spec:
  ...
  template:
    spec:
      containers:
      - name: webserver
        image: webserver
        imagePullPolicy: Always
        ports:
        - name: http
          containerPort: 8080
          protocol: TCP
        readinessProbe:
          failureThreshold: 3
          httpGet:
            path: /healthz
            port: http
            scheme: HTTP
          periodSeconds: 10
          successThreshold: 1
          timeoutSeconds: 1
      ...

```

```

server {
    listen 8080 default_server;
    server_name _;
    location / {
        root    /usr/share/nginx/html;
        index   index.html index.htm;
    }

    location /healthz {
        access_log off;
        return 200;
    }
}

```

nginx.conf

Pod – Scheduling

1. Filter

- Node-Selector für Labels
<https://kubernetes.io/docs/reference/labels-annotations-taints>
- Toleration für Taints
<https://kubernetes.io/docs/concepts/scheduling-eviction/taint-and-toleration>

```
apiVersion: v1
kind: Pod
metadata:
  ...
spec:
  containers:
  - name: nginx
    nodeSelector:
      disktype: ssd
  tolerations:
  - key: class
    value: do280
    operator: "Equal"
    effect: "NoSchedule"
```

```
apiVersion: v1
kind: Node
metadata:
  labels:
    disktype: ssd
spec:
  taints:
  - key: class
    value: do280
    effect: NoSchedule
```

Pod – Scheduling

2. Scoring

Affinity/Anti-Affinity-Rules

```
apiVersion: v1
kind: Pod
metadata:
  name: with-node-affinity
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
              - key: kubernetes.io/os
                operator: In
                values:
                  - linux
      preferredDuringSchedulingIgnoredDuringExecution:
        - weight: 1
          preference:
            matchExpressions:
              - key: another-node-label-key
                operator: In
                values:
                  - another-node-label-value
  containers:
    - name: with-node-affinity
      image: ...
```

...DuringScheduling: während des Scheduling

IgnoredDuringExecution: Pod wird weiter ausgeführt,
auch wenn sich nach dem Scheduling Node-Labels ändern

<https://kubernetes.io/docs/concepts/scheduling-eviction/assign-pod-node>

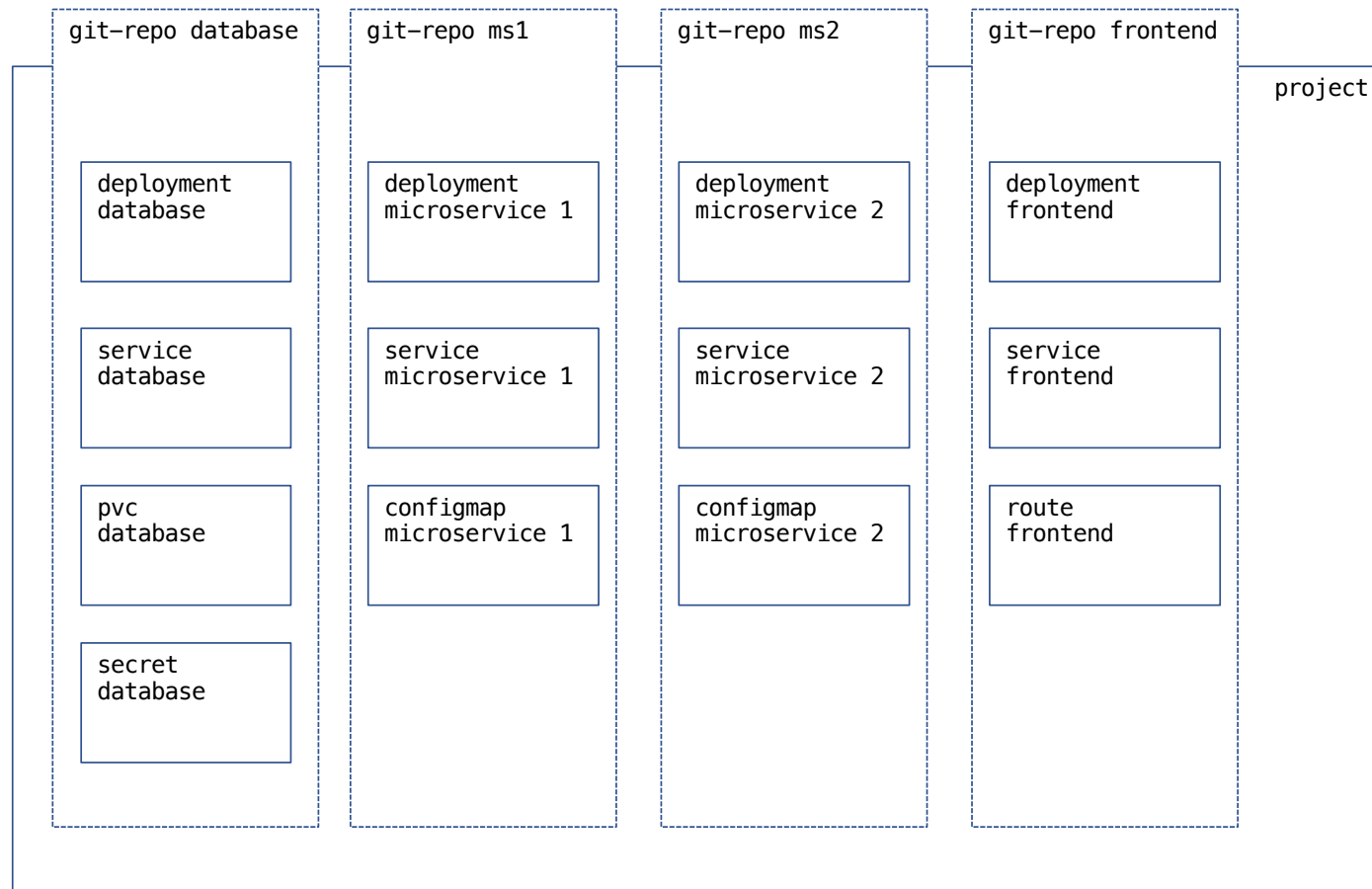
<https://www.cncf.io/blog/2021/07/27/advanced-kubernetes-pod-to-node-scheduling>

Pod Verteilung auf unterschiedliche Nodes:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  ...
spec:
  selector:
    matchLabels:
      app: store
  replicas: 3
  template:
    metadata:
      labels:
        app: store
    spec:
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            - labelSelector:
                matchExpressions:
                  - key: app
                    operator: In
                    values:
                      - store
              topologyKey: "kubernetes.io/hostname"
      containers:
        - ...
```

gleiche Label und gleicher Hostname → AntiAffinity
Pods werden auf unterschiedlichen Nodes verteilt

https://docs.openshift.com/container-platform/4.10/nodes/scheduling/nodes-scheduler-pod-affinity.html#nodes-scheduler-pod-affinity-example-antiaffinity_nodes-scheduler-pod-affinity



Template: parametrisierbare Liste von Resource-Definitionen

```
kind: Template
apiVersion: v1
metadata:
  name: rest-sample
objects:
- apiVersion: v1
  kind: Service
  metadata:
    name: ${APP_NAME}
  spec:
    selector:
      app.kubernetes.io/name: ${APP_NAME}
    ...
- apiVersion: apps/v1
  kind: Deployment
  metadata:
    name: ${APP_NAME}
  spec:
    template:
      spec:
        containers:
          - name: ${APP_NAME}
            image: ${IMAGE_NAME}
    ...
- apiVersion: v1
  kind: Route
  ...
parameters:
- description: Application Name
  name: APP_NAME
  required: true
- description: Image Name
  name: IMAGE_NAME
  required: true
...
```

```
oc process (TEMPLATE | -f FILENAME) -p APP_NAME=... | oc create -f -
oder bei installiertem Template ( oc create -f template.yml ) :
oc new-app <template-name>
```

```

apiVersion: template.openshift.io/v1
kind: Template
labels:
  app: php-sample
metadata:
  name: php-sample
  labels:
    samples.operator.openshift.io/managed: "true"
  app: php-sample
objects:
- apiVersion: v1
  kind: Service
  metadata:
    annotations:
      description: Exposes and load balances the application pods
  ...

```

Labels für alle Objekte

Labels nur für das Template

```

oc process -f template.yml -o yaml
apiVersion: v1
items:
- apiVersion: v1
  kind: Service
  metadata:
    labels:
      app: php-sample
  ...
- apiVersion: route.openshift.io/v1
  kind: Route
  metadata:
    labels:
      app: php-sample
  ...
- apiVersion: image.openshift.io/v1
  kind: ImageStream
  metadata:
    labels:
      app: php-sample
  ...

```

```

$ oc create -f template.yml
template.template.openshift.io/php-sample created

```

```

$ oc get template php-sample -o yaml
apiVersion: template.openshift.io/v1
kind: Template
labels:
  app: php-sample
metadata:
  labels:
    samples.operator.openshift.io/managed: "true"
  ...

```

\$ oc delete all -l app=php → alle vom Template erzeugten Objekte werden gelöscht

Helm-Chart: Paket-Manager (Lifecycle + Template-Engine + Dependencies)

```
$ helm create sample
Creating sample

$ tree sample
sample
├── charts
├── Chart.yaml
├── templates
│   ├── deployment.yaml
│   ├── _helpers.tpl
│   ├── hpa.yaml
│   ├── ingress.yaml
│   ├── NOTES.txt
│   ├── serviceaccount.yaml
│   ├── service.yaml
│   └── tests
│       └── test-connection.yaml
└── values.yaml
```

Helm-Chart: Paket-Manager (Lifecycle + Template-Engine + Dependencies)

```
Chart.yml
apiVersion: v1
name: sample
description: Sample Application
version: 1.0
appVersion: 1.0
dependencies:
- name: dep1
  version: ...
  repository: ...
```

```
values.yml
image:
  repository: quay.io/redhat.io/sample
  tag: '2'

service:
  port: 8080

env:
  ...

dep1.key: value
```

```
helm create
helm dependency update
helm install / upgrade / rollback / uninstall

helm template (lokales processing)
```

Templates:

```
deployment.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ APP_NAME }}
spec:
  template:
    selector:
      matchLabels:
        {{- include "sample.selectorLabels" . | nindent 6 }}
    spec:
      containers:
        - image: {{.Values.image.repository}}: {{.Values.image.tag}}
      ...
```

Go-Templates:

```
_helpers.tpl
{{- define "sample.selectorLabels" -}}
app.kubernetes.io/name: {{ include "sample.name" . }}
app.kubernetes.io/instance: {{ .Release.Name }}
{{- end -}}
...
```

Kustomize: generieren/transformieren von Ressourcen (Manifeste mit minimalen Meta-Daten)

```
kind: Kustomization                                kustomization.yml
apiVersion: kustomize.config.k8s.io/v1beta1

namespace: sample

resources:
- deployment.yml
- service.yml
- route.yml
- http://...    -> kustomize.yml in Git/Web-Repository

images:
- name: sample
  newName: registry/sample
  newTag: '5'

commonLabels:
  app.kubernetes.io/instance: sample

configMapGenerator:
- name: rest-sample
  literals:
  - LAUNCH_JBOSS_IN_BACKGROUND=1
...
```

resources → <https://github.com/hashicorp/go-getter#url-format>

```
apiVersion: apps/v1
metadata:
  name: rest-sample
spec:
  replicas: 1
  template:
    spec:
      containers:
      - name: sample
        image: sample
```

```
$ oc kustomize <kustom-dir>
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app.kubernetes.io/instance: rest-sample
  name: rest-sample
  namespace: sample
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/instance: sample
  template:
    containers:
      image: registry/sample:5
...

$ oc apply -k .
```

Kustomize Overlays : erzeugen unterschiedlicher Varianten von einer Basis-Vorlage

```
                                base/kustomization.yml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

resources:
- deployment.yml
- service.yml
- route.yml
```

```
                                overlays/test/kustomization.yml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

resources:
- ../../base

namespace: test

images:
- name: sample
  newName: registry/sample
  newTag: '3-SNAPSHOT'
```

```
                                overlays/production/kustomization.yml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

resources:
- ../../base

namespace: production

images:
- name: sample
  newName: registry/sample
  newTag: '5'
```

```
$ oc apply -k overlays/test
service/sample configured
deployment.apps/sample configured
route.route.openshift.io/sample configured

$ oc apply -k overlays/production
...
```

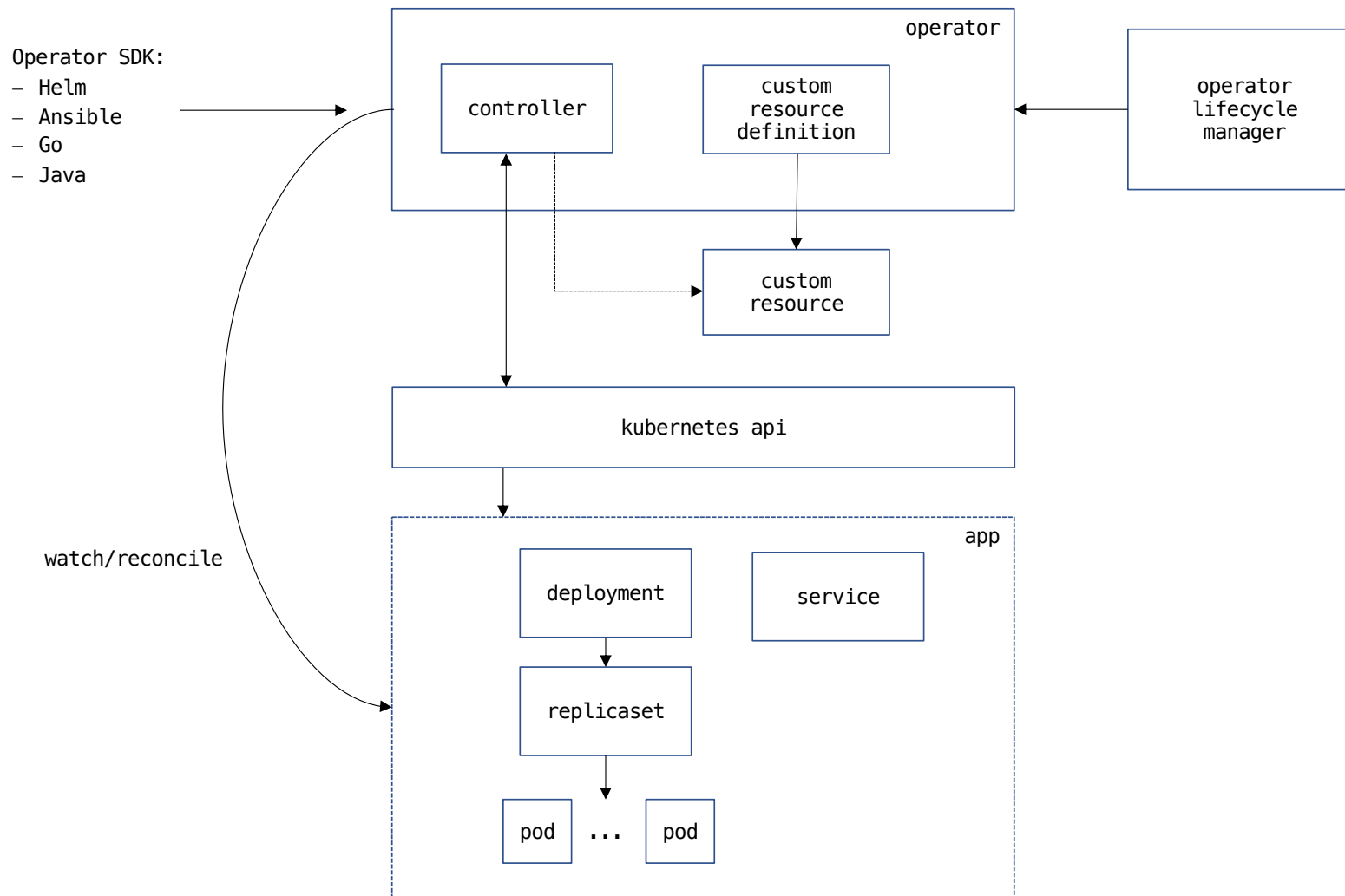
https://kubectldocs.kubernetes.io/guides/extending_kustomize/exec_krm_functions

```
$ grep -A3 images kustomization.yml
images:
- name: webserver
  newName: quay.io/danielstraub/webserver
  newTag: "1.0"

$ kustomize edit set image webserver=quay.io/danielstraub/webserver:2.0
$ grep -A3 images kustomization.yml
images:
- name: webserver
  newName: quay.io/danielstraub/webserver
  newTag: "2.0"

$ oc apply -k .
configmap/webserver-kt5mdg45d2 unchanged
service/webserver unchanged
deployment.apps/webserver configured
route.route.openshift.io/webserver unchanged

$ curl https://stage-prod.apps.eu46a.prod.ole.redhat.com
Hello, D0288
Version 2.0
```

```
$ oc api-resources | grep keycloak
```

OperatorHub

Discover Operators from the Kubernetes community and Red Hat partners, curated by Red Hat. Install Operators on your clusters to provide optional add-ons and shared services to your development and self-service experience.

All Items

AI/Machine Learning

Application Runtime

Big Data

Cloud Provider

Database

Developer Tools

Integration & Delivery

Logging & Tracing


Modernization & Migration

Monitoring

Networking

All Items


Q key



Keycloak Operator

provided by Red Hat

An Operator for installing and managing Keycloak



Keycloak Operator

24.0.5-opr.1 provided by Red Hat

Install

Channel

stable-v24

Version

24.0.5-opr.1

Capability level

☒ Basic Install

☒ Seamless Upgrades

☒ Full Lifecycle

☒ Deep Insights

☐ Auto Pilot

A Kubernetes Operator based on the Operator SDK for installing and managing Keycloak.

Keycloak lets you add authentication to applications and secure services with minimum fuss. No need to deal with storing users or authenticating users. It's all available out of the box.

The operator can deploy and manage Keycloak instances on Kubernetes and OpenShift. The following features are supported:

- Install Keycloak to a namespace
- Import Keycloak Realms

```
oc api-resources | grep keycloak
keycloakrealmimports
keycloaks
```

operator keycloak

```
$ oc project keycloak
$ oc create secret tls keycloak-cert --cert=...fullchain.pem --key=..privkey
```

```
$ oc apply -f - <<EOF
kind: Keycloak
apiVersion: k8s.keycloak.org/v2alpha1
metadata:
  name: keycloak
  labels:
    app: keycloak
  namespace: keycloak
spec:
  instances: 1
  hostname:
    hostname: idp.<wildcard-domain>
  http:
    tlsSecret: keycloak-cert
EOF
```

```
$ oc get all
```

NAME	READY	STATUS	RESTARTS	AGE
pod/keycloak-0	1/1	Running	0	5m28s
pod/rhbk-operator-565b768dcd-qgqss	1/1	Running	0	13m

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/keycloak-discovery	ClusterIP	None	<none>	7800/TCP	5m28s
service/keycloak-service	ClusterIP	172.30.35.121	<none>	8443/TCP	5m28s

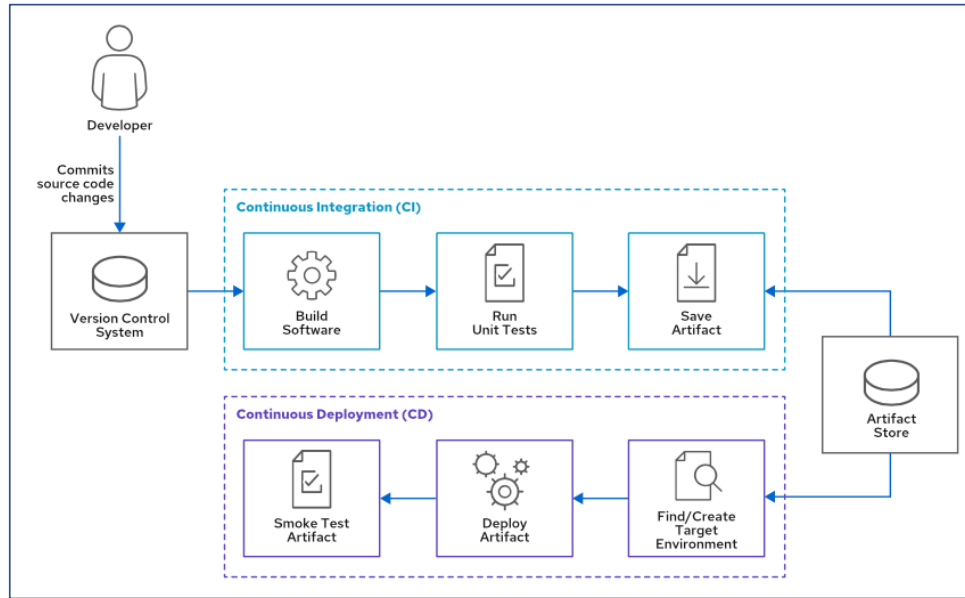
NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/rhbk-operator	1/1	1	1	13m

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/rhbk-operator-565b768dcd	1	1	1	13m

NAME	READY	AGE
statefulset.apps/keycloak	1/1	5m28s

```
NAME
route.route.openshift.io/keycloak-ingress-vvpv8 ...
```

```
$ oc get secret keycloak-initial-admin -o jsonpath='{.data.password}' | base64 -d
xxxxxxxxxx
```



Continuous Integration
Continuous Delivery

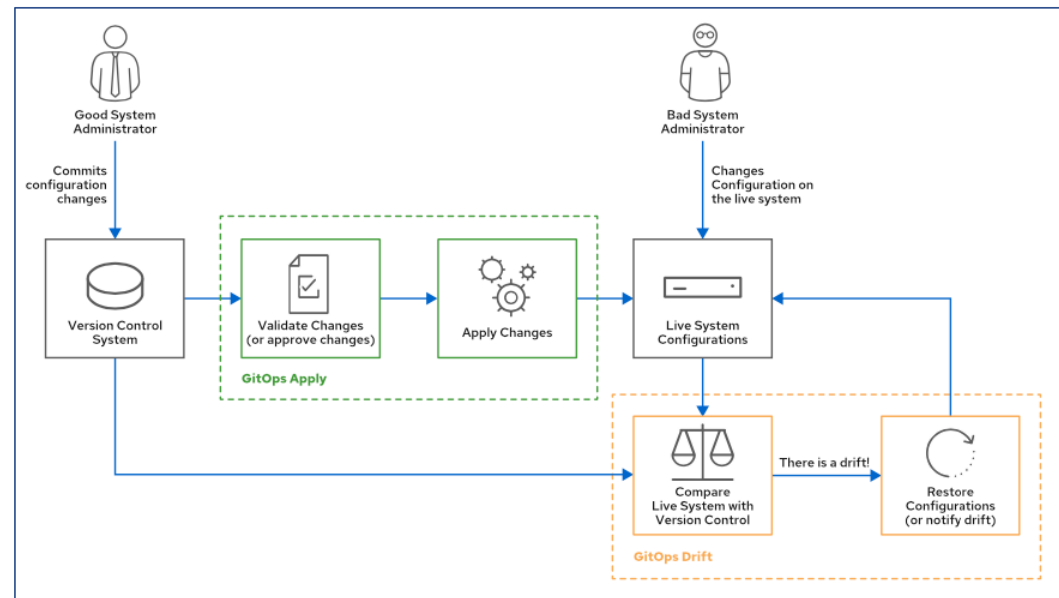
→ Developer
→ running application

Jenkins, CruiseControl, TeamCity, GitLab ...
Kubernetes native (Tekton, Argo CD, ...)

GitOps Workflow

→ Administrators
→ live System

Ansible, Puppet, Terraform ...
ArgoCD, FluxCD, JenkinsX



Tekton - Komponenten

Konfiguration:

- Step
Script/Programm welches in einem Container ausgeführt werden
wird innerhalb eines Tasks definiert
- Task
definieren Ein- und Ausgabeparameter, Umgebung für Steps
enthalten 1..* Steps
- Pipeline
definieren Ein- und Ausgabeparameter, Umgebung für Tasks
enthalten 1..* Tasks
- Eventlistener
reagieren auf HTTP-Events z.B. von VCS

ClusterTasks : global, vom Operator bereitgestellt

```
$ oc get clustertasks
NAME                                AGE
argocd-task-sync-and-wait          175d
buildah                             175d
git-cli                             175d
git-clone                           175d
...
```

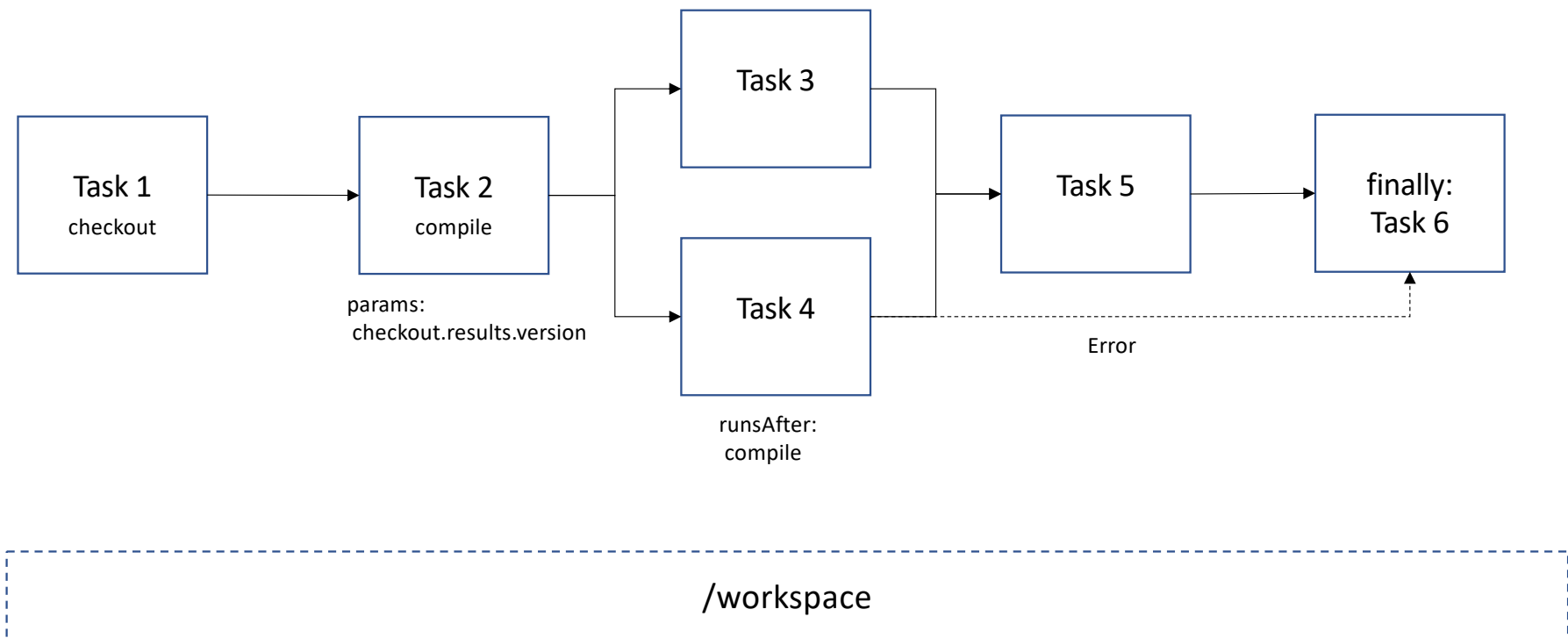
Laufzeit:

- TaskRun
Ausführung eines Task mit konkreten Parameter, Umgebung
(z.B. Workspace)
- PipelineRun
Ausführung einer Pipeline mit konkreten Parameter, Umgebung
(z.B. Workspace)

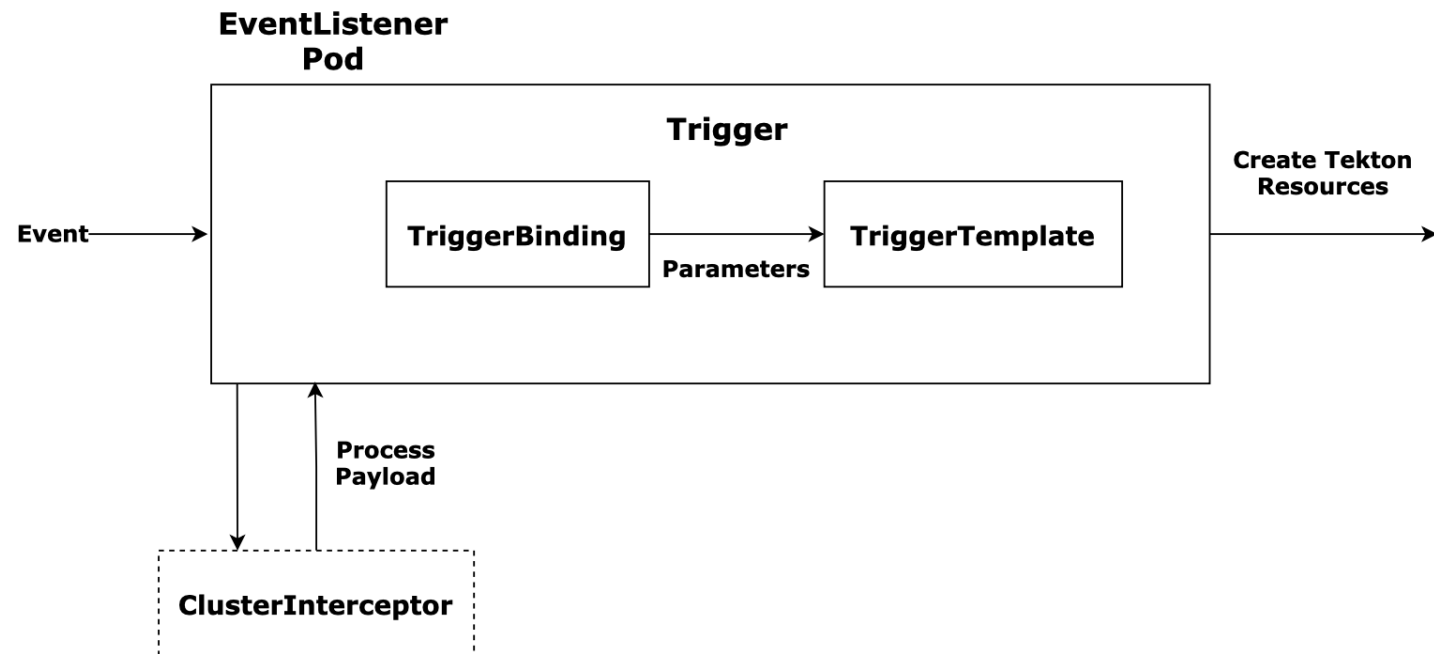
- kein zentraler Server wie z.B. Jenkins !
- Pipeline, Task / PipelineRun, TaskRun sind *namespaced* Ressourcen

Tekton PipelineRun

- Tasks: können über Ein- und Ausgabeparameter/Bedingungen mit einander verknüpft werden
die Tasks werden dann ausgeführt wenn Parameter/Bedingungen erfüllt sind
- Workspace: für alle Tasks sollte ein gemeinsamer Workspace (Persisten-Volume) verwendet werden auf dem gearbeitet wird
(z.B. ausgechecktes Git-Repo)



Tekton Eventlistener



Tekton Eventlistener

```
apiVersion: triggers.tekton.dev/v1beta1
kind: EventListener
metadata:
  name: webserver-build
spec:
  serviceAccountName: pipeline
  triggers:
  - name: github
    interceptors:
    - ref:
        name: cel
      params:
      - name: filter
        value: body.commits[0].author.name != 'pipeline'
      - name: overlays
        value:
          - key: git_short_rev
            expression: body.head_commit.id.truncate(7)
          - key: git_rev
            expression: body.head_commit.id
    bindings:
    - name: git-rev
      value: $(extensions.git_rev)
    - name: git-short-rev
      value: $(extensions.git_short_rev)
  template:
    ref: webserver-build
```

Deployment
el-webserver-build

Service
el-webserver-build

Route
...
(muss selber erstellt werden)

Tekton Ressourcen können normal mit den oc-Client verwendet werden

```
$ oc create -f pr.yml && oc get tr -w
```

webserver-build-run-qdvnd-fetch-repo	Unknown	Pending
webserver-build-run-qdvnd-build-image	True	Succeeded
webserver-build-run-qdvnd-commit-changes	True	Succeeded
webserver-build-run-qdvnd-fetch-repo	True	Succeeded

```
$ oc logs webserver-build-run-qdvnd-fetch-repo-pod
```

```
...
{"level":"info","ts":1719519668.1135974,"caller":"git/git.go:176","msg":"Successfully cloned git@github.com:dstraub/lab-sample.git @ ad23aa3566111e6f588f32256bf5e4fdd77b5199"}
{"level":"info","ts":1719519668.1679106,"caller":"git/git.go:215","msg":"Successfully initialized and updated submodules in path /workspace/output/"}
```

```
$ oc logs el-webserver-build-6d84df79bb-mztzj | tail -n 2 | jq
```

```
{
  "severity": "info",
  "timestamp": "2024-06-27T20:29:08.529Z",
  "logger": "eventlistener.event-broadcaster",
  "caller": "record/event.go:298",
  "message": "Event(v1.ObjectReference{Kind:\"EventListener\", Namespace:\"webserver-build\", Name:\"webserver-build\", UID:\"65993abf-ccf6-4aa4-a9ca-957bbda2e2a7\", APIVersion:\"triggers.tekton.dev/v1beta1\", ResourceVersion:\"22473585\", FieldPath:\"\"}): type: 'Normal' reason: 'dev.tekton.event.triggers.done.v1' ",
  "commit": "f76be74"
}
{
  "severity": "info",
  "timestamp": "2024-06-27T20:29:08.540Z",
  "logger": "eventlistener",
  "caller": "sink/sink.go:420",
  "message": "interceptor stopped trigger processing: rpc error: code = FailedPrecondition desc = expression body.commits[0].author.name != 'pipeline' did not return true",
  "commit": "f76be74",
  "eventlistener": "webserver-build",
  "/triggers-eventid": "2fcf657c-f678-4189-a218-06d45ddfba12",
  "eventlistenerUID": "65993abf-ccf6-4aa4-a9ca-957bbda2e2a7",
  "/trigger": "github"
}
```

GitOps – Workflow mit Pipelines:

- Apply Pipeline:
 - validate : `oc apply --validate --dry-run [folder/files from Git]`
 - apply : `oc apply`
- Drift Pipeline:
 - diff : `oc diff [folder/files from Git]`
 - optional/restore: `oc apply`

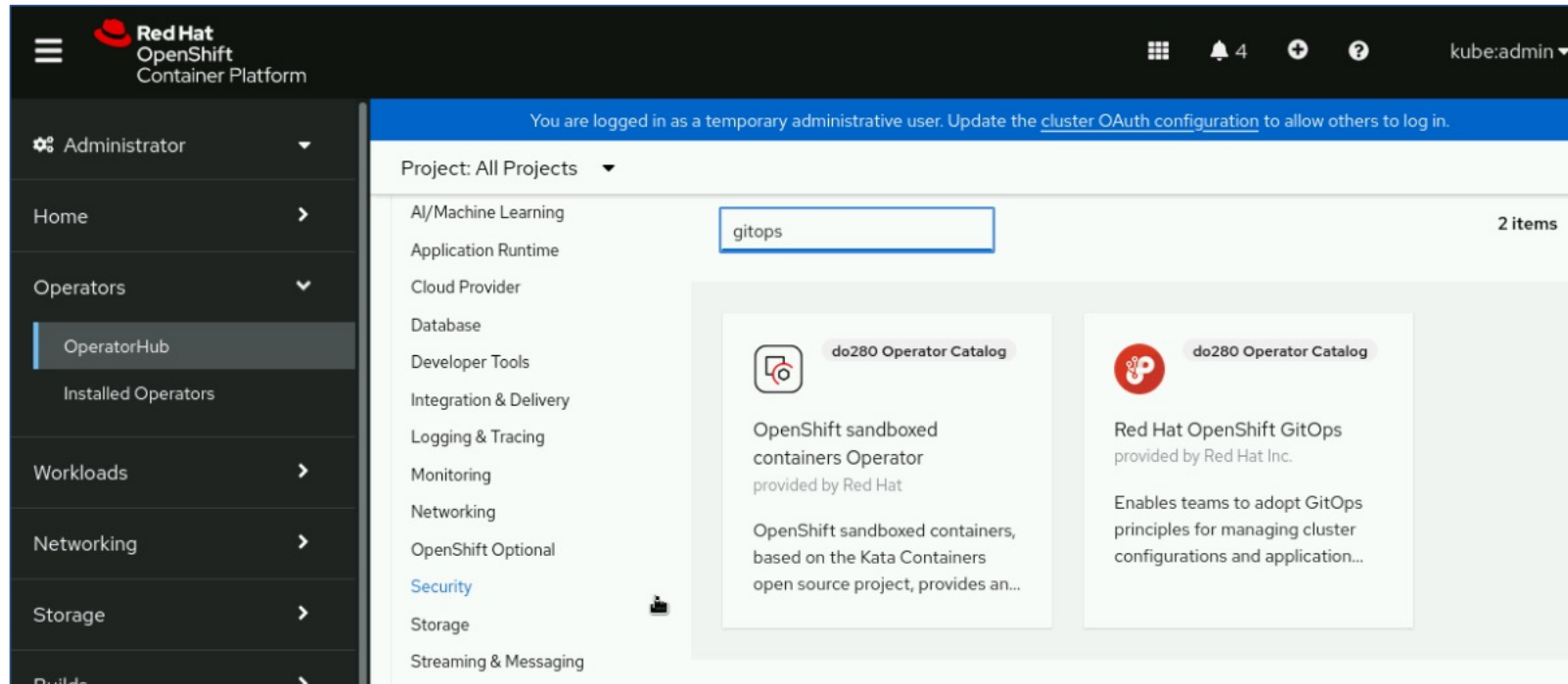
GitOps – Workflow mit ArgoCD / FluxCD:

Ableich Ist-Zustand (Cluster) mit Kustomize/Helm-Definitionen im Git

Benachrichtigungen, manueller/automatische Synchronisation bei Abweichungen

apps calibre	ssh://git@gitea.apps:10022/ds/calibre.git/overlays/production in-cluster/apps	HEAD	♥ Healthy ✓ Synced	⋮
apps pgadmin	ssh://git@gitea.apps:10022/ds/pgadmin.git/overlays/production in-cluster/apps	HEAD	♥ Healthy ✓ Synced	⋮
apps postgres	ssh://git@gitea.apps:10022/ds/postgres.git/overlays/production in-cluster/database	HEAD	♥ Healthy ✓ Synced	⋮
apps rest-sample	ssh://git@gitea.apps:10022/ds/rest-sample.git/overlays/production in-cluster/sample	HEAD	♥ Healthy ⚠ OutOfSync	⋮

Red Hat OpenShift GitOps - Operator



ArgoCD = Openshift GitOps

← → ↻ 🏠 <https://openshift-gitops-server-openshift-gitops.apps.ocp4.example.com/applications/stage-prod?view=tree&re> ⭐

Applications / stage-prod APPLICATION DETAILS

APP DETAILS APP DIFF SYNC SYNC STATUS HISTORY AND ROLLBACK DELETE REFRESH

APP HEALTH **Healthy**

CURRENT SYNC STATUS **OutOfSync** From **HEAD (f46fb2b)** [MORE](#)

LAST SYNC RESULT **Sync OK** To **5852439** [MORE](#)

Succeeded 14 hours ago (Wed Jun 29 2022 13:13:26 GMT-0400)

Author: Daniel Straub <ds@ctrlaltdel.de> - update sync policy

Author: Daniel Straub <ds@ctrlaltdel.de> - Update kustomization.yml

Comment:

Comment:

stage-prod 14 hours

webserver-kt5mdg45d2 14 hours

webserver 14 hours

webserver 14 hours rev:2

webserver 14 hours

webserver 14 hours

webserver-b88c4 endpointslice 14 hours

webserver-567899c8fd 14 hours rev:1

webserver-7676d986b9 14 hours rev:2

webserver