

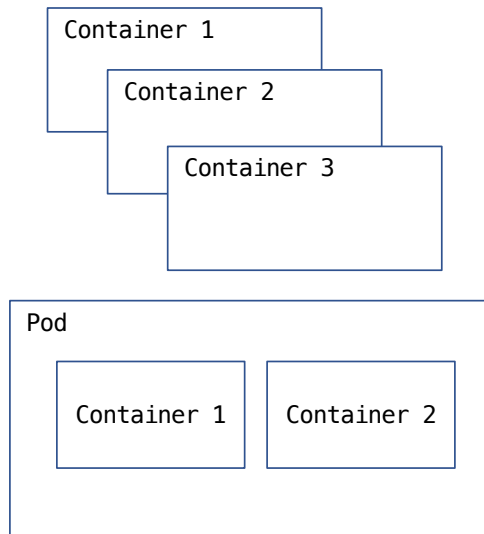
Container versus operating system differences

Container:

- niedriger Hardware-Footprint
- isolierte Umgebung
- schnelle Bereitstellung
- Bereitstellung mit mehreren Umgebungen
- Wiederverwendbar

Podman :

großer Aufwand beim Betrieb mehrerer
Container, Service-Kommunikation, Routing

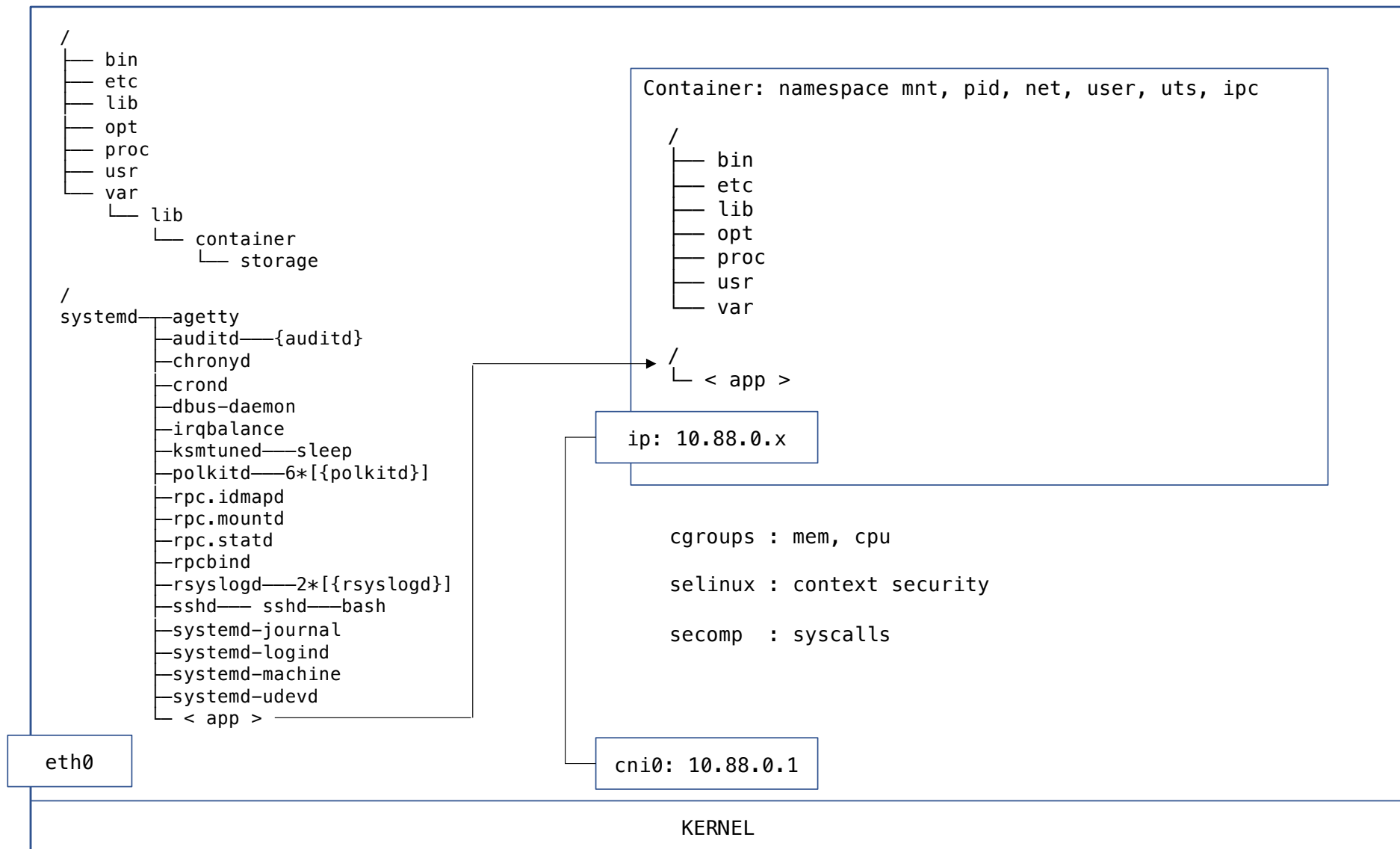


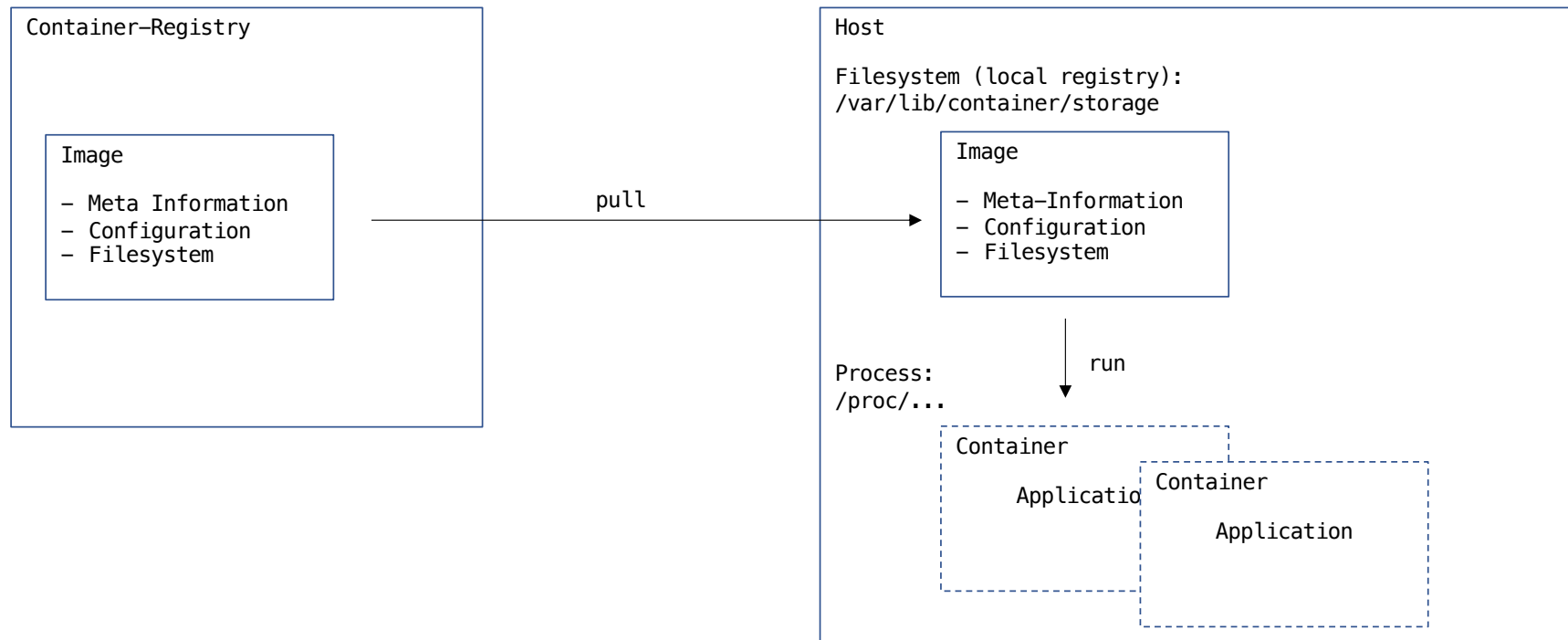
Kubernetes : Orchestrierung von Container-Anwendungen

- Service Discovery, Loadbalancing
- Horizontale Skalierung
- Health Checks
- Rolling Updates
- Secret/Configmanagement
- Operatoren: native Kubernetes Anwendungen zum Cluster- und Anwendungs-Management

Openshift (RHOCP):

- basiert auf Kubernetes
- Entwickler-Workflow (CI/CD)
- Routing
- Metriken und Log-Management
- einheitliche Benutzeroberfläche





<https://podman.io>



- Image- und Containermanagement
- OCI: Open Container Initiative
- keine Client/Serverarchitektur
- gleiche Befehlssyntax wie do...
- Kubernetes kompatibel
- yum install podman

Open Container Initiative

containers/image

runc

containers/storage

cni

<https://buildah.io>



- Erstellen von Images
- yum install buildah

skopeo:

- Kopieren von Images zwischen Registries
- Auskunft über Images

podman run : Environment

```
podman run -e <KEY>=<VALUE>
```

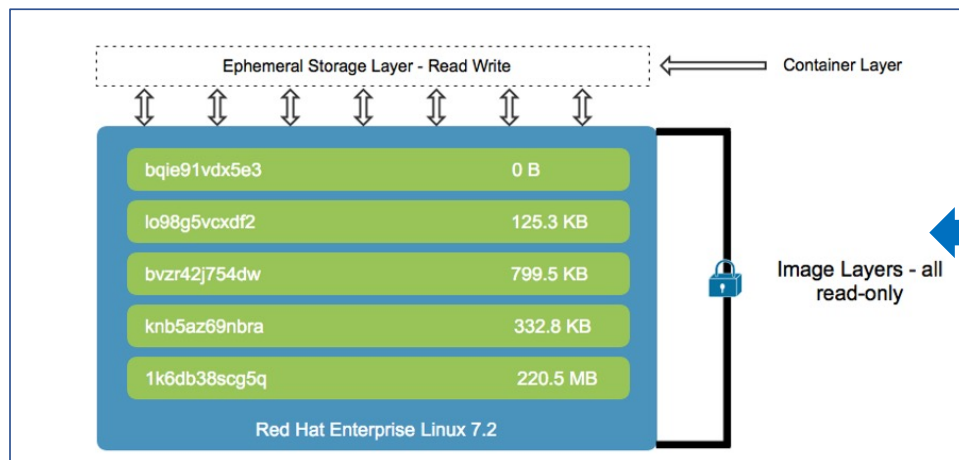
```
podman run --env-file=<host-file>
```

```
podman run --env-host=true|false
```

podman run : Volumes (Files)

```
podman run -v <host-dir>:<container-dir>
```

```
podman run --volumes-from <container-name>
```



Mapping (Mount) des Host-Filesystem in den Container:

– Permissions

```
podman unshare chown -R <container-userid> <host-dir>  
oder chmod 0777 <host-dir> ☺
```

– SELinux

```
sudo semange fcontext -a -t container_file_t '<host-dir>(/.*)?'  
sudo restorecon -Rv <dir>
```

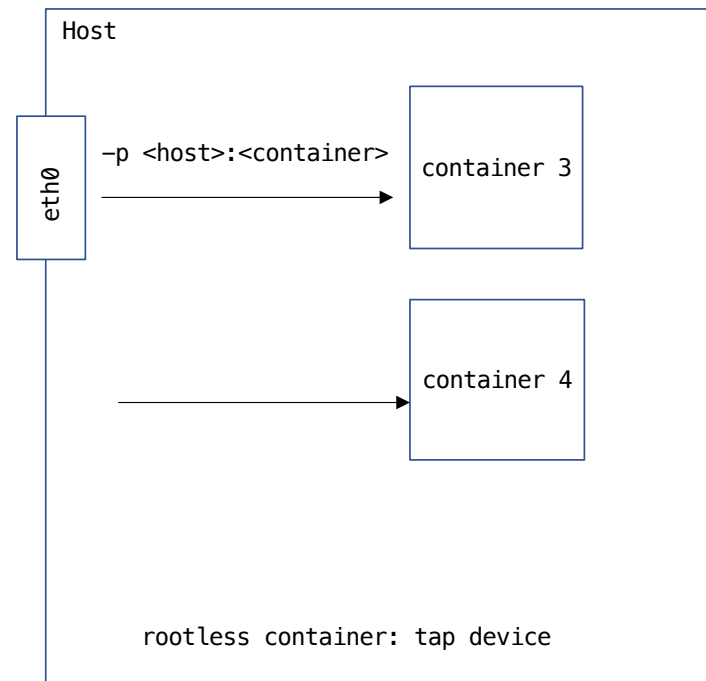
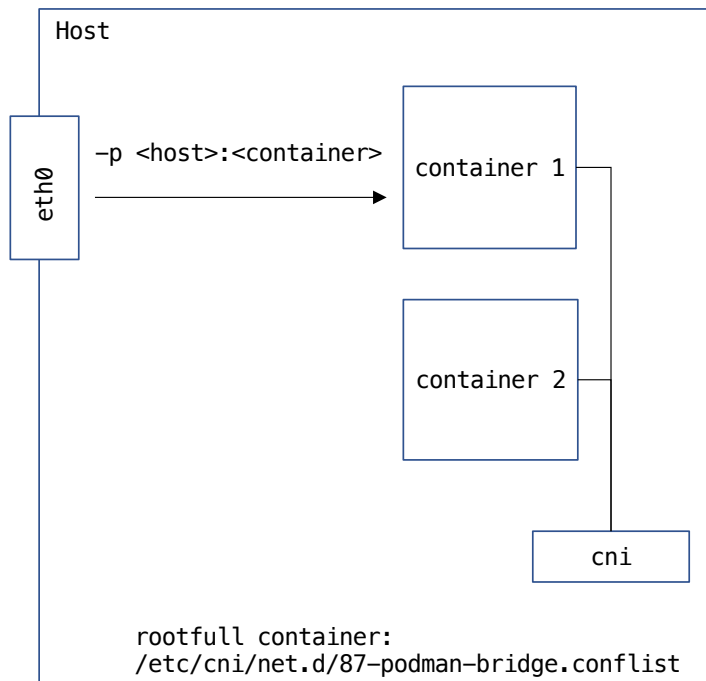
```
podman run -v <host-dir>:<container-dir> <image>
```

podman run - Publishing:

```
podman run -p <host-port>:<container-port> ...
```

```
podman run -P / --publish-all
```

```
podman port -l
```



```
$ podman images
REPOSITORY
localhost/nginx
localhost/nginx
```

TAG	IMAGE ID	CREATED	SIZE
latest	e420c54187d7	14 seconds ago	260 MB
1	2fd45c021c45	9 minutes ago	260 MB

```
$ podman tag nginx:latest nginx:1
```

```
$ podman images
REPOSITORY
localhost/nginx
localhost/nginx
<none>
```

TAG	IMAGE ID	CREATED	SIZE
1	e420c54187d7	27 seconds ago	260 MB
latest	e420c54187d7	27 seconds ago	260 MB
<none>	2fd45c021c45	9 minutes ago	260 MB

```
$ podman image prune
2fd45c021c451352e18ed2383d967fd5d510d1551837446cc0f11202c7bbae05
```

```
$ podman images
REPOSITORY
localhost/nginx
localhost/nginx
```

TAG	IMAGE ID	CREATED	SIZE
latest	e420c54187d7	About a minute ago	260 MB
1	e420c54187d7	About a minute ago	260 MB

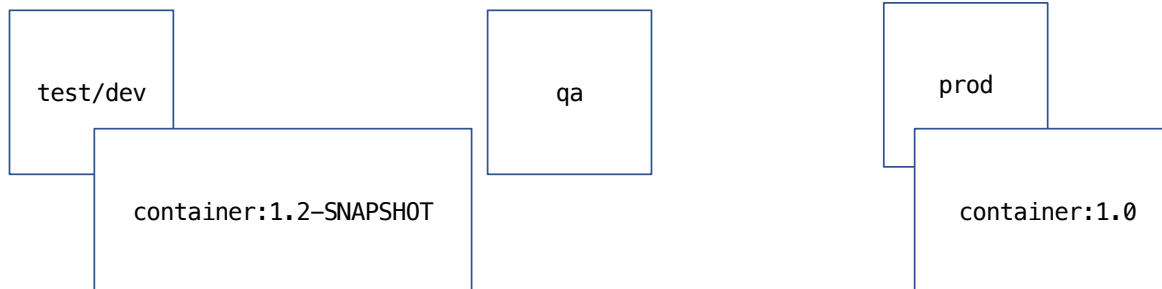


Image – Registry Push

Image-Name: <registry-name>[:<registry-port>]/<user|company|...>/<product>[:<tag>]

Default-Tag → latest

```
$ podman images
REPOSITORY                                TAG      IMAGE ID      CREATED        SIZE
localhost/do180-custom-httpd             latest   dc584a69516a  2 minutes ago  236 MB    → lokal erzeugtes Image
```

```
$ podman tag do180-custom-httpd quay.io/danielstraub/do180-custom-httpd:v1.0
```

```
$ podman images
REPOSITORY                                TAG      IMAGE ID      CREATED        SIZE
quay.io/danielstraub/do180-custom-httpd   v1.0     dc584a69516a  2 minutes ago  236 MB
localhost/do180-custom-httpd              latest   dc584a69516a  2 minutes ago  236 MB
```

```
$ podman push quay.io/danielstraub/do180-custom-httpd:1.0
```

Getting image source signatures

Copying blob cc675081b281 done

Copying blob 7f9108fde4a1 skipped: already exists

...

alternativ ohne 'tagging':

```
$ podman push [--creds <user>:<password>] do180-custom-httpd quay.io/danielstraub/do180-custom-httpd:1.0
```

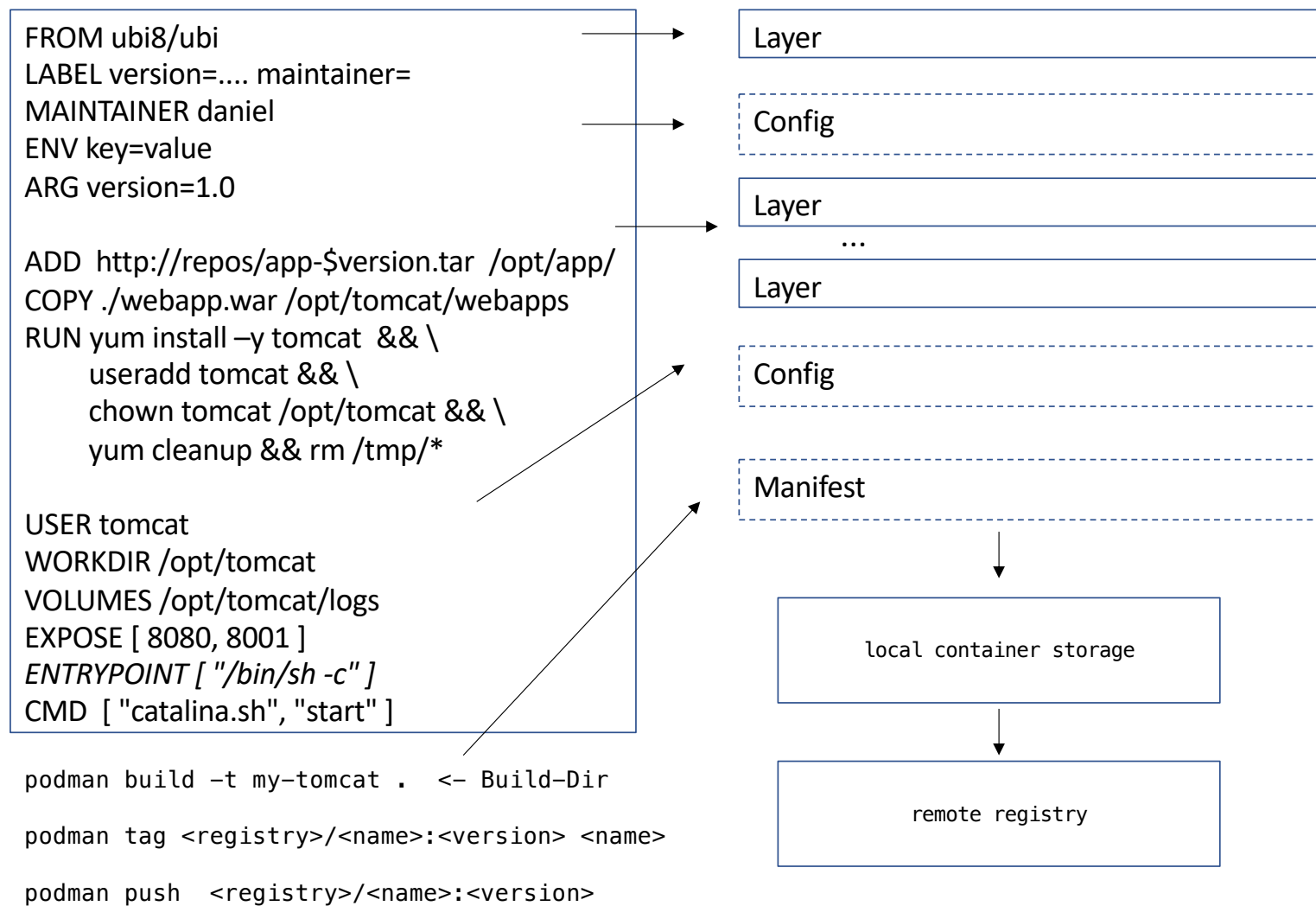
Container – Image

```
$ podman save <image> | tar -xf -  
$ tree -L 1 .
```

```
.  
├── 7076fcda2bf4ccbf058c10666d4c9dc2b4b643d3b6f770ed328c505387d21360  
├── 7076fcda2bf4ccbf058c10666d4c9dc2b4b643d3b6f770ed328c505387d21360.tar  
├── cbadeb4613603e1251cd6a24d6f2aa1d1bcd14a4fd2b85375e97d72a7a22764b.json  
├── manifest.json  
└── repositories
```

```
[  
  {  
    "Config": "cbadeb4613603e1251cd6a24d6f2aa1d1bcd14a4fd2b85375e97d72a7a22764b.json",  
    "RepoTags": [  
      "localhost/nginx:latest"  
    ],  
    "Layers": [  
      "7076fcda2bf4ccbf058c10666d4c9dc2b4b643d3b6f770ed328c505387d21360.tar"  
    ]  
  }  
]
```

podman build - Containerfile



Verwenden von YUM/DNF beim Image-Build

```
$ podman run --rm ubi8/ubi cat /etc/yum.repos.d/ubi.repo
[ubi-8-baseos]
name = Red Hat Universal Base Image 8 (RPMs) – BaseOS
baseurl = https://cdn-ubi.redhat.com/content/public/ubi/dist/ubi8/8/$basearch/baseos/os
enabled = 1
gpgkey = file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
gpgcheck = 1
...
```

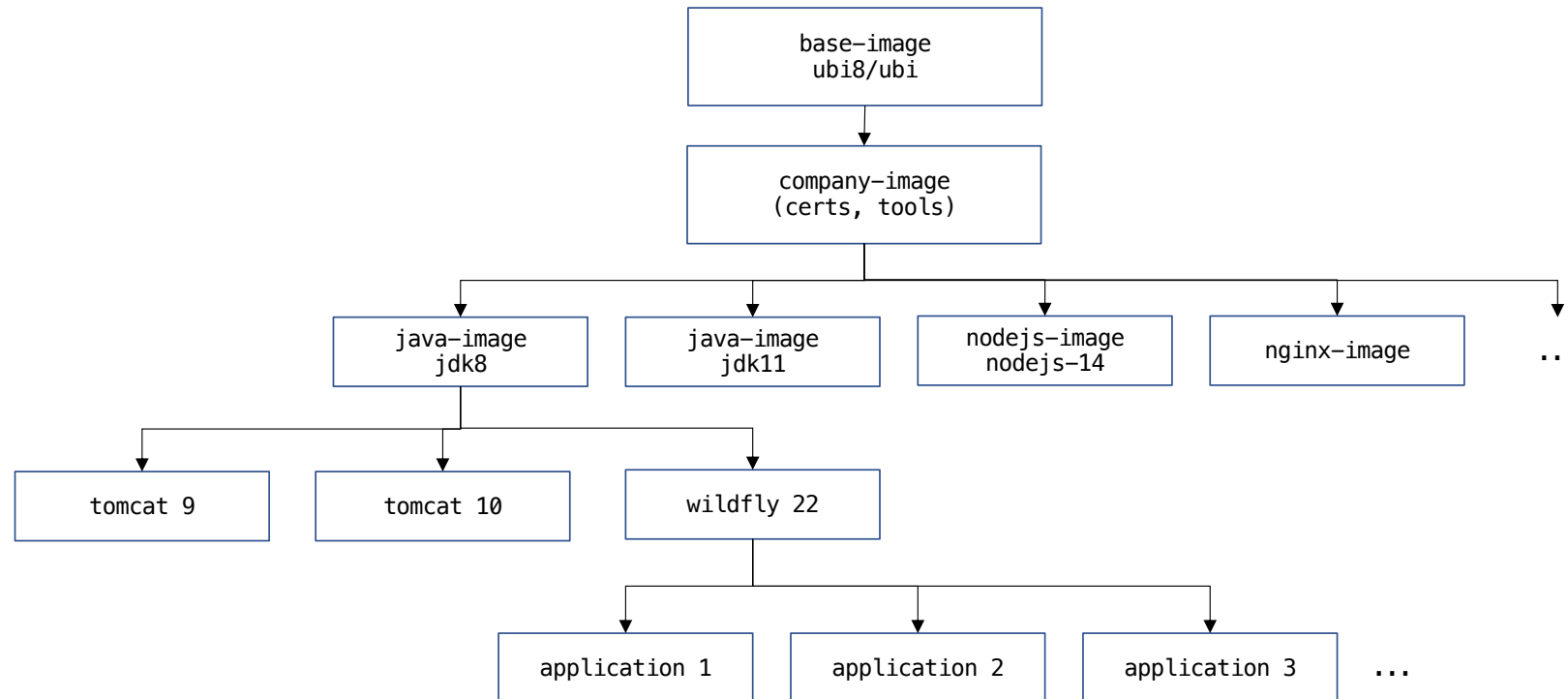
yum "telefoniert"
nach aussen !

Lösung: beim podman-build andere yum-Konfiguration (z.B. vom Host) mounten !

Bei Verwendung von Satellite/Subscriptions ggf. auch die notwendigen Zertifikate/GPG Schlüssel.

```
$ sudo podman build -v /etc/yum.repos.d:/etc/yum.repos.d -v /etc/pki:/etc/pki -v /etc/rhsm:/etc/rhsm .
```

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>

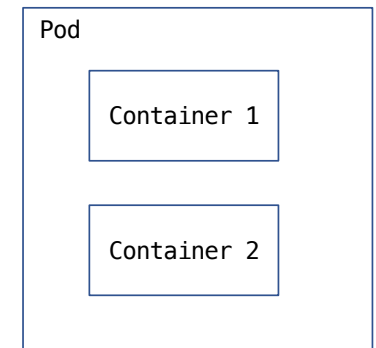
Lokales Testen eines Containers: `podman run --user 1000680000:0 <image>`

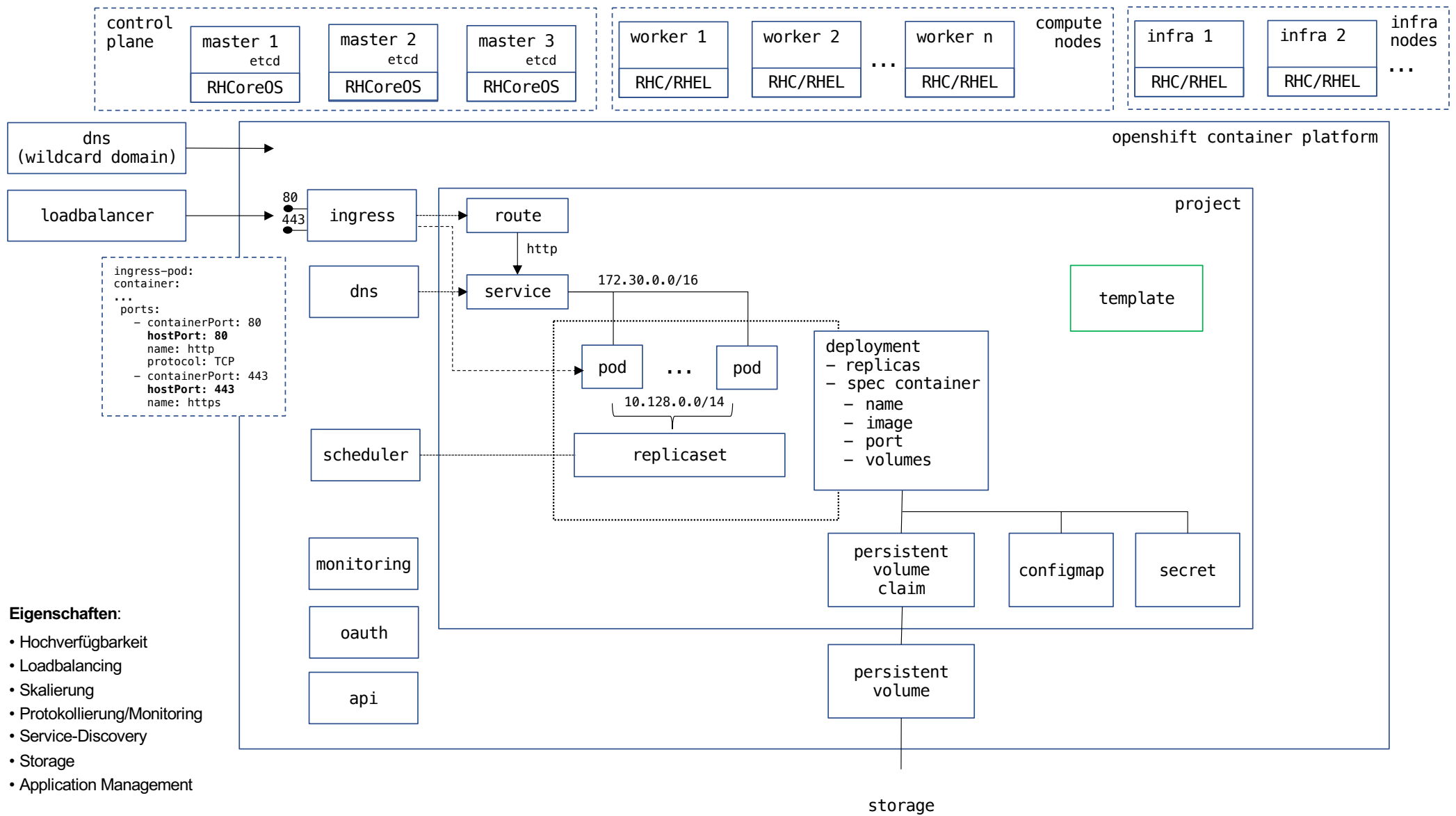
- Openshift
Orchestrierungsservice zur Bereitstellung, Verwaltung und Skalierung von Container-Anwendungen
- Deklaratives System
Status wird in Ressourcen (YAML/JSON) definiert und durch Controller hergestellt
IaC – Infrastructure as Code (<https://blog.nelhage.com/post/declarative-configuration-management>)

```
$ 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
Route
PersistenceVolumeClaim
Secrets
Configmaps
Imagestream
BuildConfig
Node
PersistenceVolume
Operator
CustomResourceDefinition

- kleinste Workload-Resource ist der Pod → Gruppe von unterschiedlichen Containern
- meistens 1:1 Beziehung (1 Pod enthält ein Container)

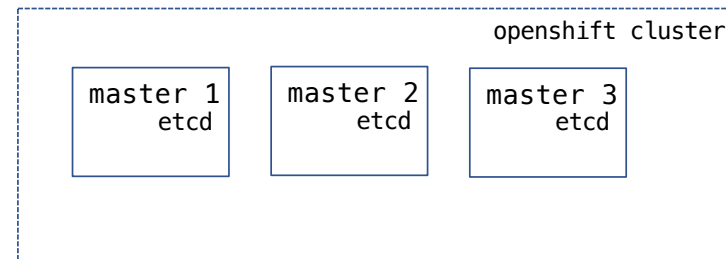




Openshift Resources (Manifest)

```
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: webserver
  namespace: do180
  labels:
    app: webserver
spec:
  replicas: 2
  selector:
    matchLabels:
      app: webserver
  template:
    metadata:
      labels:
        app: webserver
    spec:
      containers:
        - name: webserver
          image: quay.io/danielstraub/webserver:do180
          ports:
            - containerPort: 8080
              name: http
              protocol: TCP

```

```

apiVersion: v1
kind: Service
metadata:
  name: webserver
  namespace: do180
  labels:
    app: webserver
spec:
  type: ClusterIP
  selector:
    app: webserver
  ports:
    - name: http
      port: 80
      protocol: TCP
      targetPort: http

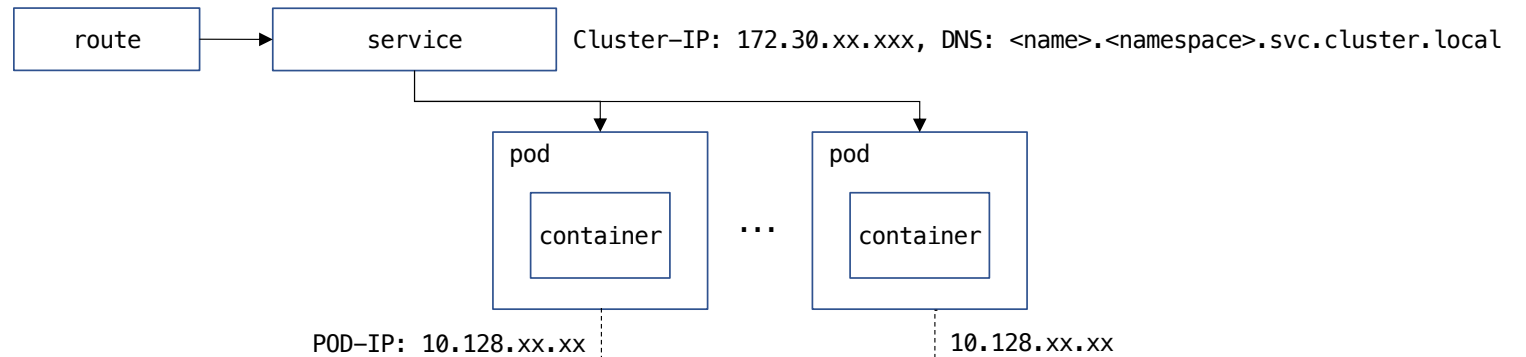
```

```

apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: webserver
  namespace: do180
  labels:
    app: webserver
spec:
  host: do180.apps.eu410.prod.nextc1e.com
  to:
    kind: Service
    name: webserver
    port:
      targetPort: http

```

Host: <name>.<namespace>.<wildcard-domain>



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

```
$ oc create -f .
deployment.apps/webserver created
route.route.openshift.io/webserver created
service/webserver created
```

```
$ oc get all
```

NAME	READY	STATUS	RESTARTS	AGE
pod/webserver-86bb596c54-54865	1/1	Running	0	21s

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/webserver	ClusterIP	172.30.89.171	<none>	80/TCP	7m49s

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/webserver	1/1	1	1	7m49s

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/webserver-86bb596c54	1	1	1	21s

NAME	HOST/PORT	PATH	SERVICES	PORT	TERMINATION	WILDCARD
route.route.openshift.io/webserver	do180.apps.eu410.prod.nextcle.com		webserver	http		

```
$ curl http://do180.apps.eu410.prod.nextcle.com
Hello, D0180
```

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

...

Usage:

```
oc new-app (IMAGE | IMAGESTREAM | TEMPLATE | PATH | URL ...) [flags]
```

Beispiele:

```
$ oc new-app https://quay.io/dstraub/nginx --name nginx
```

Container-Image

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

Builder-Image
(s2i)

Git-Projekt (Source)

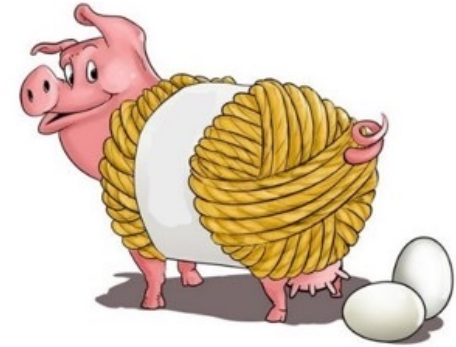


Deployment

Service

Imagestream

BuildConfig



```

$ oc create --help
Usage:
  oc create -f FILENAME [flags]
...
Available Commands:
  ...
  configmap      Create a config map from a local file, directory or literal value
  deployment     Create a deployment with the specified name
  route          Expose containers externally via secured routes
  secret         Create a secret using specified subcommand
  service        Create a service using a specified subcommand

$ oc create deployment --image=quay.io/danielstraub/webserver --port=8080 -o yaml webserver
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webserver
  labels:
    app: webserver
spec:
  replicas: 1
  selector:
    matchLabels:
      app: webserver
  template:
    metadata:
      labels:
        app: webserver
    spec:
      containers:
      - image: quay.io/danielstraub/webserver
        ports:
        - containerPort: 8080

```

```
$ 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

- `oc login -u <user> -p <password> <api-server-url>`
- `oc new-project <name>`
- `oc create -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 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/4.12/cli_reference/openshift_cli/developer-cli-commands.html

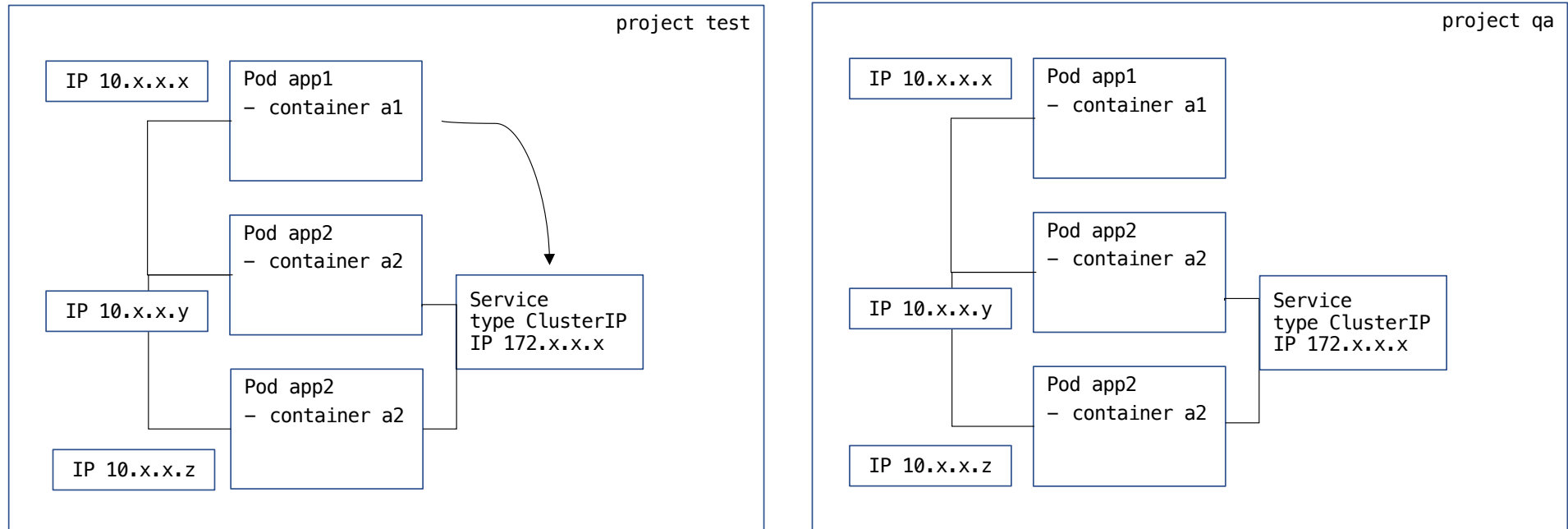

```
$ oc expose service <service>
```

Route: <service>-<project>.<wildcard-domain> ← Wildcard-Domain im DNS

```
$ oc expose service <service> --hostname=<domain>
```

console-openshift-console.apps.eu46.prod.nextcle.com





DNS:
A: person.test.svc.cluster.local
SVC: _443._tcp.https.<service>.test.svc.cluster.local

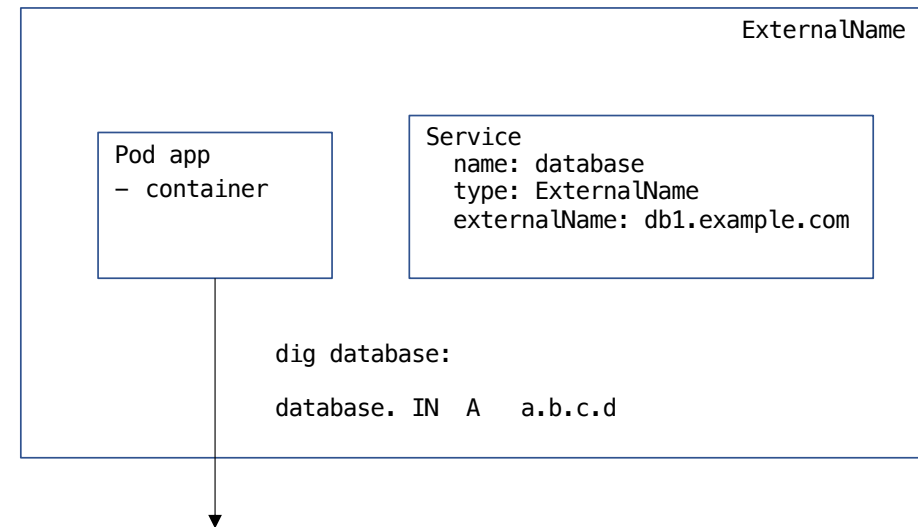
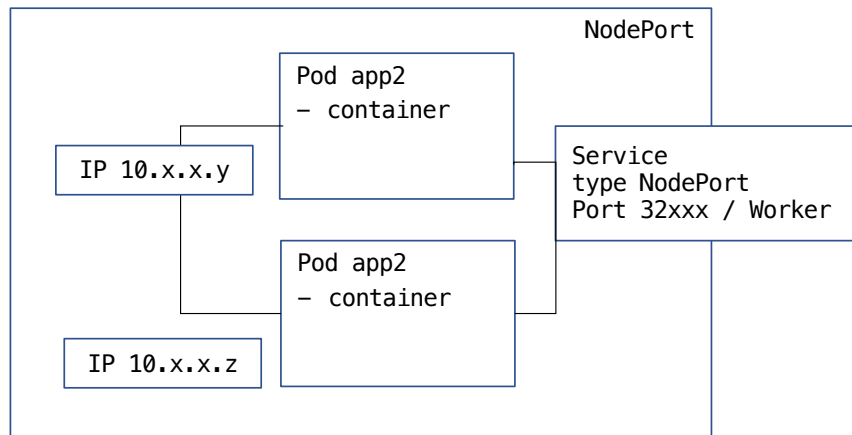
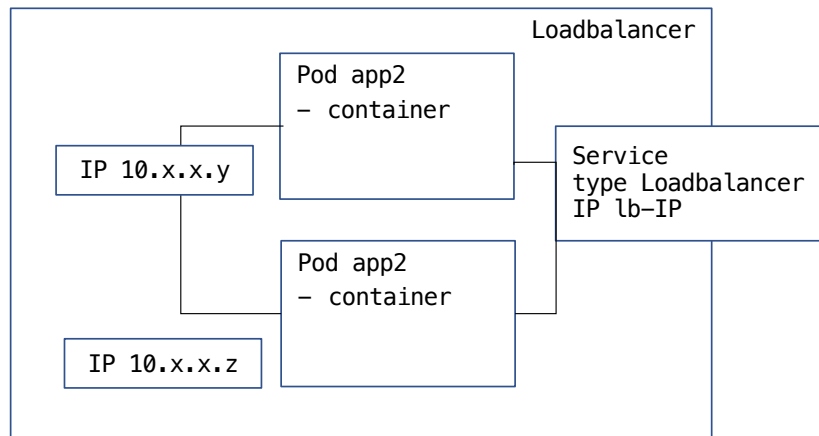
/etc/resolv.conf:
search test.svc.cluster.local svc.cluster.local ...

DNS:
A: person.qa.svc.cluster.local
SVC: _443._tcp.https.<service>.qa.svc.cluster.local

/etc/resolv.conf:
search qa.svc.cluster.local svc.cluster.local ...

→ einfacher DNS-Lookup nach <service> in jedem Projekt

nur Cloud-Provider ! (oder MetalLB-Operator)



db1.example.com
a.b.c.d

Pod | Service | Route

```
apiVersion: v1
kind: Pod
metadata:
  name: webserver
  labels:
    app.kubernetes.io/instance: httpd
spec:
  containers:
  - name: httpd
    image: ...
    ports:
    - name: http
      containerPort: 8080
    - name: https
      containerPort: 8443
```

```
apiVersion: v1
kind: Service
metadata:
  name: webserver
spec:
  selector:
    app.kubernetes.io/instance: httpd
  ports:
  - name: http
    port: 80
    protocol: TCP
    targetPort: http
  - name: https
    port: 443
    protocol: TCP
    targetPort: https
```

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: webserver-secure
spec:
  host: webserver.apps....
  to:
    kind: Service
    name: webserver
  port:
    targetPort: https
```

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>

Lokales Testen eines Containers: `podman run --user 1000680000:0 <image>`

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 !

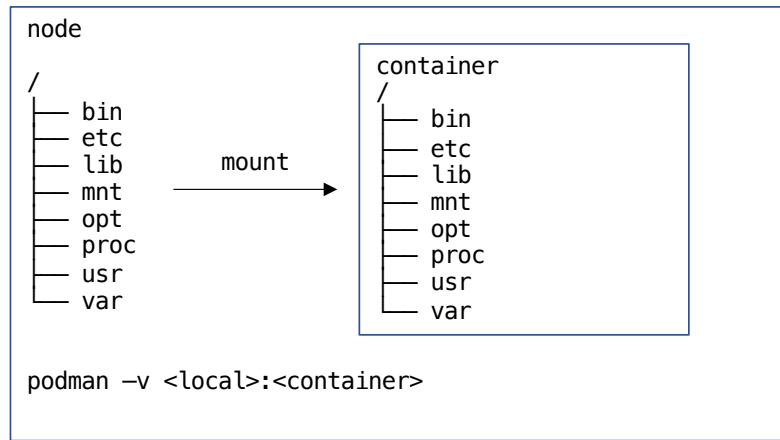
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gitea
  namespace: apps
...
spec:
  template:
    spec:
      serviceAccountName: gitea
  ...
```

```
# 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 !

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
- ...

Secrets:

- Passwörter, Token, Zertifikate ...
- typisiert: basic-auth, dockercfg, tls, opaque
- Inhalte sind base64-encodiert, 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
```

beim Anlegen im Manifest:

```
stringData:
  password: 123456
```

```
$ 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>
```

```
$ oc create -f cm.yml | oc apply -f cm.yml
```

ConfigMap:

- generische Key-Value Daten

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: ...
  namespace: ...
  annotation:
binaryData:
  keystore: |
    7oAMCAQICCF7Dt6ZDf6TgMA0GCSqGSIb3DQEBAQEI1ZSQULa
    MTEQMA4GA1UECwwHU ...
data:
  HOME: /usr/share/nginx
  default.conf: |
    server {
      listen 8080 default_server;
      server_name _;
      location / {
        root /usr/share/nginx/html;
        index index.html index.htm;
      }
    }
```

Secrets: Verwendung als Umgebungs-Variablen

```
apiVersion: v1
kind: Pod
metadata:
  name: secret-env-pod
spec:
  containers:
  - name: mycontainer
    image: redis
    envFrom:
      configMapRef:
        name: < cm >
    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
```

Persistence

Administrator erzeugt PersistentVolume

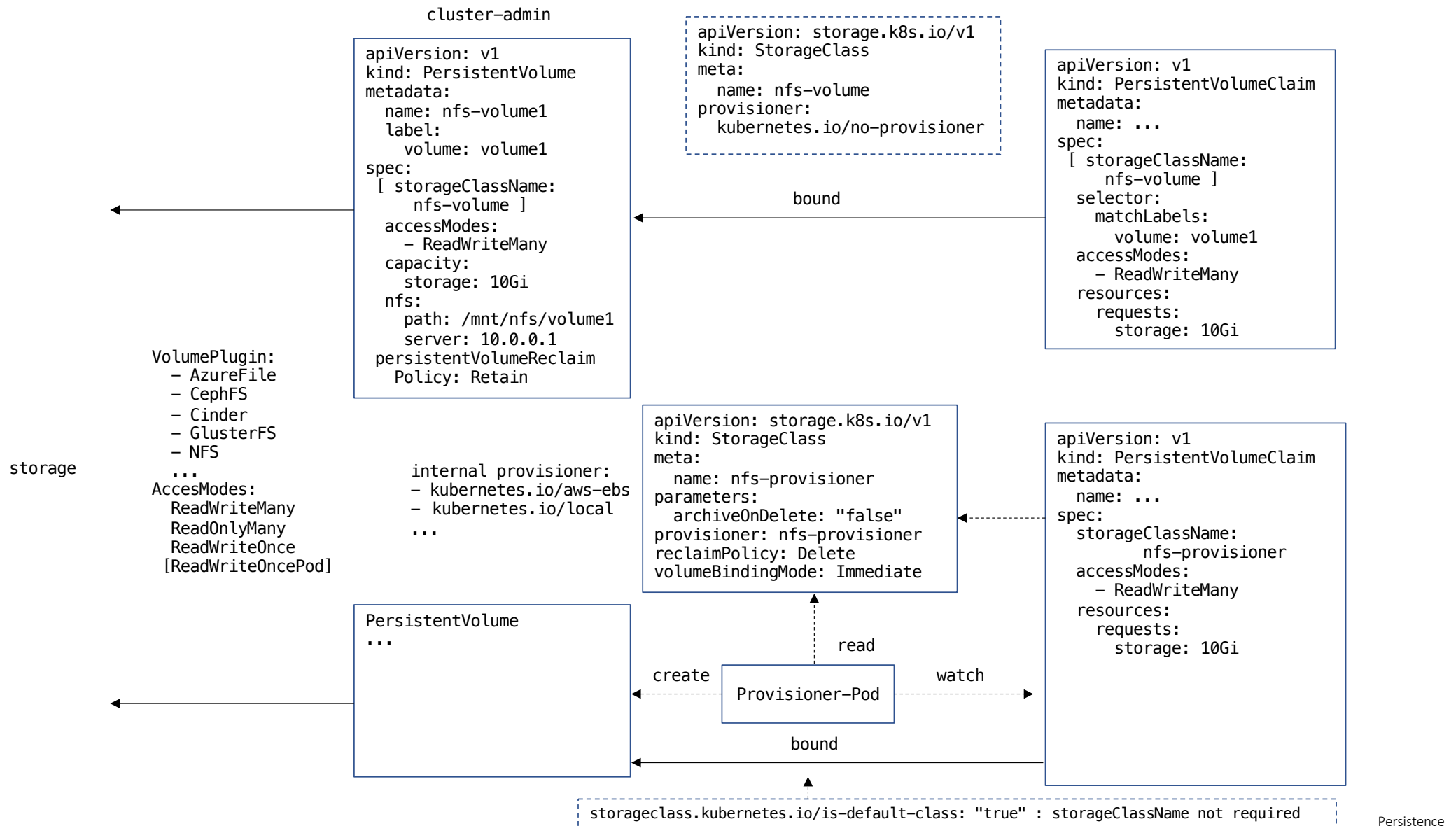
```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfs-data
  labels:
    volume: nfs-data
spec:
  accessModes:
    - ReadWriteMany
  capacity:
    storage: 10Gi
  nfs:
    path: /mnt/nfs/data
    server: 10.0.0.1
  persistentVolumeReclaimPolicy: Retain
```

Anwendung erstellt Anforderung

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: html-data
spec:
  accessModes:
    - ReadWriteMany
  selector:
    matchLabels:
      volume: nfs-data
  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

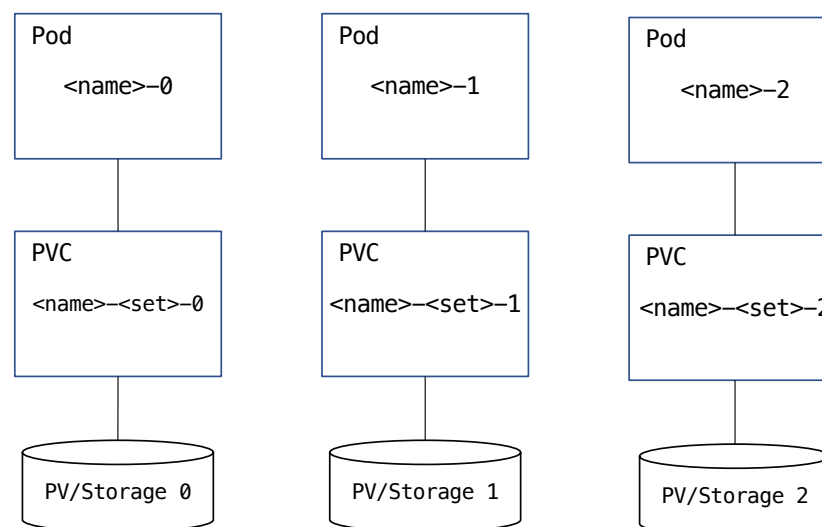
```

apiVersion: v1
kind: Service
metadata:
  name: <svc-name>
  labels:
    app: nginx
spec:
  clusterIP: None
  ports:
  - name:
    ...
  selector:
    app: <name>
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: <name>
spec:
  serviceName: <svc-name>
  replicas: 2
  ...
  template:
    metadata:
      labels:
        app: <name>
    spec:
      containers:
      - name: ...
        ...
        volumeMounts:
        - name: <pvc-name>
          mountPath: ...
  volumeClaimTemplates:
  - metadata:
      name: <pvc-name>
    spec:
      accessModes:
      - ReadWriteOnce
      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



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 `.spec.containers.startupProbe`
Container wird bei negativer Startup-Probe sofort beendet

Probes:

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

```
httpGet:
  path: /healthz
  port: healthz-port
  scheme: 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

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 ... 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

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

<https://kubernetes.io/docs/concepts/configuration/manage-resources-containers>

<https://sysdig.com/blog/troubleshoot-kubernetes-oom>

<https://www.datadoghq.com/blog/kubernetes-cpu-requests-limits>

```
apiVersion: v1
kind: Deployment
metadata:
  ...
template:
  spec:
    containers:
      - name: <name>
        resources:
          requests:
            memory: 64Mi
            cpu: 200m
          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%)
```


Quotas und LimitRange:

```
$ oc create -f quotas.yml
$ oc describe resourcequotas quotas
Name:          quotas
Namespace:     demo
Resource       Used  Hard
-----
requests.cpu   0    100m
requests.memory 0    500M
```

```
$ oc create -f server.yml
deployment.apps/webserver created
```

```
$ oc get all
NAME                                READY  UP-TO-DATE  AVAILABLE  AGE
deployment.apps/webserver           0/1    0            0           11s
```

...

```
$ oc get events
LAST SEEN   TYPE      REASON      OBJECT                      MESSAGE
50s         Warning   FailedCreate  replicaset/webserver-c48d6fbd Error creating: pods "webserver-c48d6fbd-fh97s" is forbidden: failed quota: quotas: must specify requests.cpu,requests.memory
```

```
$ oc create -f limit.yml
limitrange/limits created
$ oc describe limitranges limits
```

Name:	limits					
Namespace:	demo					
Type	Resource	Min	Max	Default Request	Default Limit	Max Limit/Request Ratio
Container	cpu	-	-	50m	50m	-
Container	memory	-	-	200Mi	200Mi	-

```
$ oc rollout restart deployment webserver
deployment.apps/webserver restarted
```

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: quotas
spec:
  hard:
    requests.cpu: 100m
    requests.memory: 500M
```

```
apiVersion: v1
kind: LimitRange
metadata:
  name: limits
spec:
  limits:
    - type: Container
      default:
        memory: 200Mi
        cpu: 50m
    ...
```

Quotas:

```
$ oc rollout restart deployment webserver
deployment.apps/webserver restarted
```

```
$ oc get pods
```

NAME	READY	STATUS	RESTARTS	AGE
webserver-789b574675-z4vsz	1/1	Running	0	56s

```
$ oc describe resourcequotas quotas
```

Name:	quotas	
Namespace:	demo	
Resource	Used	Hard
requests.cpu	50m	100m
requests.memory	200Mi	500M

```
$ oc scale deployment webserver --replicas=3
deployment.apps/webserver scaled
```

```
$ oc get all
```

NAME	READY	STATUS	RESTARTS	AGE
pod/webserver-789b574675-n5rp8	1/1	Running	0	12s
pod/webserver-789b574675-z4vsz	1/1	Running	0	2m10s

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/webserver	2/3	2	2	8m

```
$ oc describe resourcequotas quotas
```

Name:	quotas	
Namespace:	demo	
Resource	Used	Hard
requests.cpu	100m	100m
requests.memory	400Mi	500M

```
apiVersion: apps/v1
kind: Deployment
metadata:
...
spec:
...
  template:
...
    spec:
      containers:
      - image: quay.io...
        name: webserver
        resources:
          requests:
            cpu: 50m
            memory: 200Mi
```

<https://docs.openshift.com/container-platform/4.10/applications/quotas/quotas-setting-per-project.html>

<https://docs.openshift.com/container-platform/4.10/applications/quotas/quotas-setting-across-multiple-projects.html>

<https://docs.openshift.com/container-platform/4.10/nodes/clusters/nodes-cluster-limit-ranges.html>

Deployment-Strategien

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

Beenden eines Pods:

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

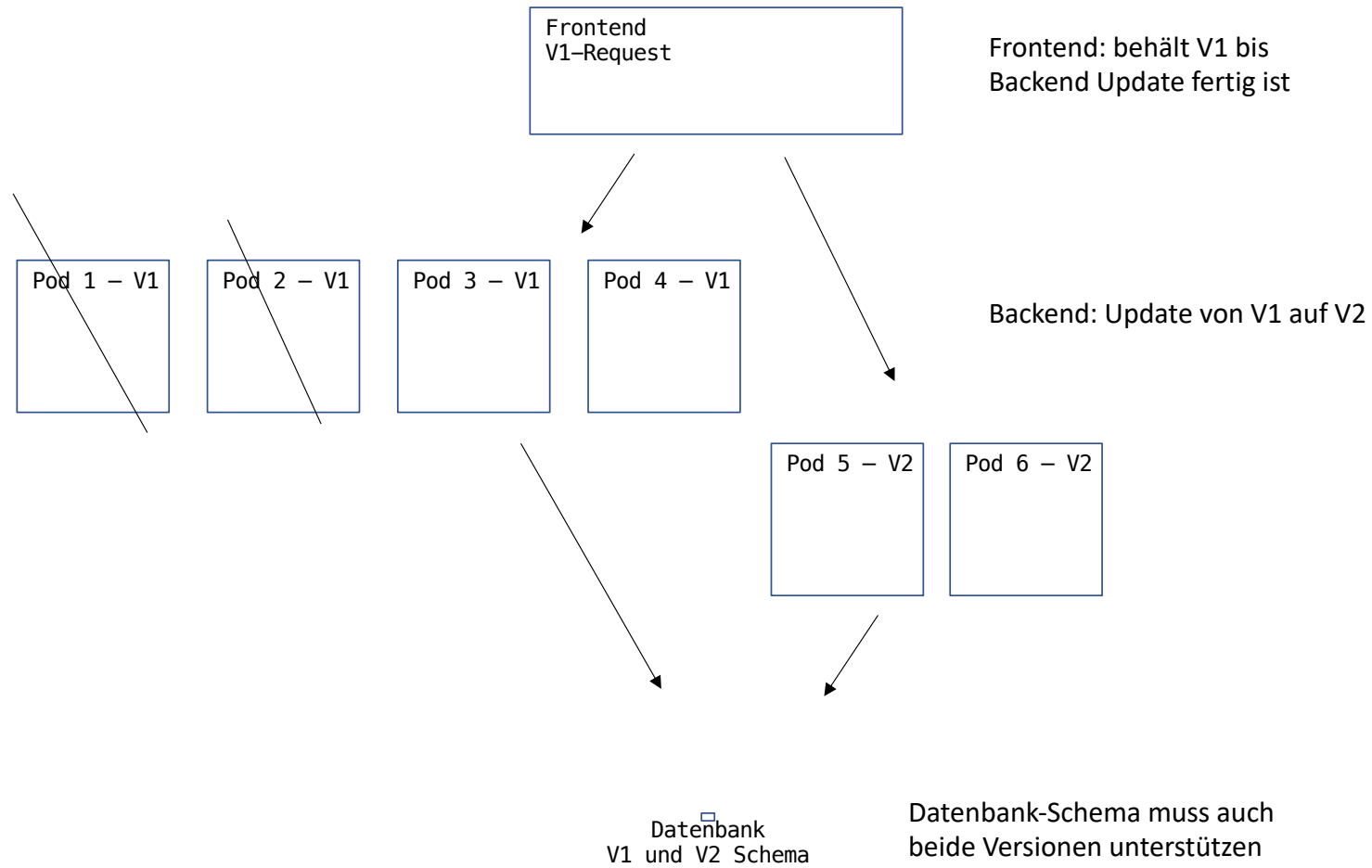
```
kind: Deployment
metadata:
  name: ...
spec:
  revisionHistoryLimit: 3   (default: 10)
  replicas: 4
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1           ← max. 5 Pods aktiv
      maxUnavailable: 1
  ...
  template:
    spec:
      containers:
      - ...
      terminationGracePeriodSeconds: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

```
oc rollout SUBCOMMAND (DEPLOYMENTCONFIG | DEPLOYMENT)
```

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 (DEPLOYMENTCONFIG | DEPLOYMENT) [--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
```

Imagestream:

- enthält Verweise (Zeiger) auf Images und deren Tags (keine Images)
- Verwendung in Deployment als Image und Trigger oder in BuildConfig als (S2I) BuilderImage
- Import aus externer Registry oder Ergebnis eines Build

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

latest
  updates automatically from registry quay.io/do288/nginx:latest
  * quay.io/do288/nginx@sha256:c34f57431167fca470730b67a1a8636126d2464eee619ec8d0b577c8e63bffe0

1.2
  updates automatically from registry quay.io/do288/nginx:1.2
  * quay.io/do288/nginx@sha256:ee508edacfe0bc1e6af43a15348b400a7d97121507348bd5fb5effb6b9f8d84e

1.1
  updates automatically from registry quay.io/do288/nginx:1.1
  * quay.io/do288/nginx@sha256:674ab485f6e83f162eb4bdaf12986469c7b4f484f65fbb18f3b03218fd5f36e4

1.0
  updates automatically from registry quay.io/do288/nginx:1.0
  * quay.io/do288/nginx@sha256:693b30b107da26
```

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

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
```

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 -}}
...
```

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

helm template (lokales processing)
```


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

```
kustomization.yml

kind: Kustomization
apiVersion: kustomize.config.k8s.io/v1beta1

namespace: sample

resources:
- deployment.yml
- service.yml
- route.yml
- https://<gitrepo>/... -> kustomize.yml in Git-Repository

commonLabels:
  app.kubernetes.io/instance: sample

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

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

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

```
deployment.yml

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

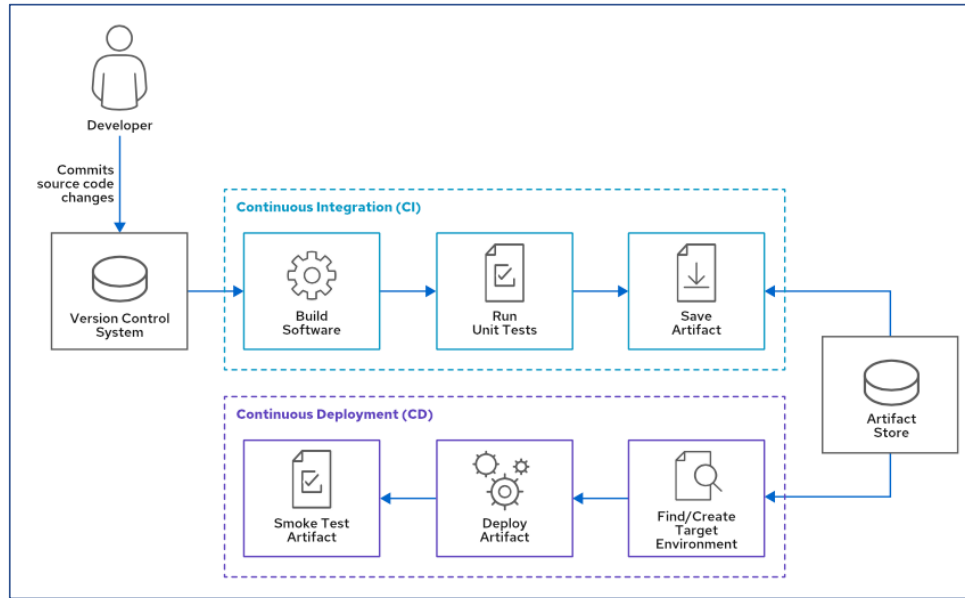
namespace: production

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://kubectl.docs.kubernetes.io/guides/extending_kustomize/exec_krm_functions



Continuous Integration
Continuous Delivery

→ Developer
→ running application

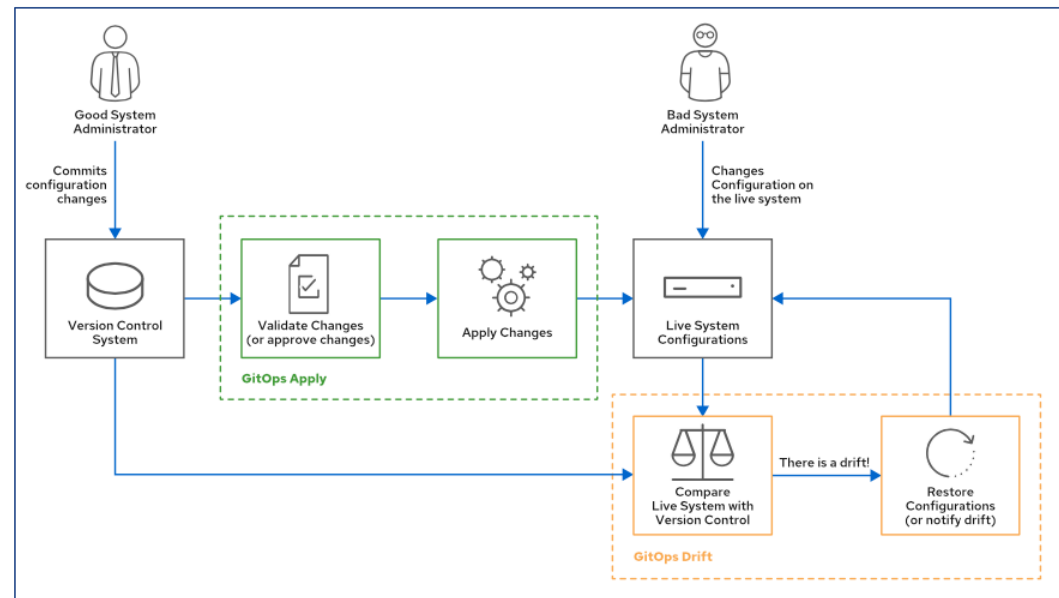
Jenkins, CruiseControl, TeamCity, GitLab ...
Kubernetes native - Tekton

You build it, you run it !

GitOps Workflow

→ Administrators
→ live System

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



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 (openshift-gitops)

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

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	⋮

General

Access

Collaborators
Moderation options

Code and automation

Branches
Tags
Actions
Webhooks
Environments
Codespaces
Pages

Security

Code security and analysis
Deploy keys
Secrets and variables

Webhooks / Manage webhook

Settings

Recent Deliveries

We'll send a POST request to the URL below with details of any subscribed events, in the format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information [documentation](#).

Payload URL *

https://argocd.cluster.straubcloud.de/api/webhook

Content type

application/json

Secret

SSL verification

By default, we verify SSL certificates when delivering payloads.

☒ Enable SSL verification
☐ Disable (not recommended)

Synchronisierung mit Webhook
oder Polling

