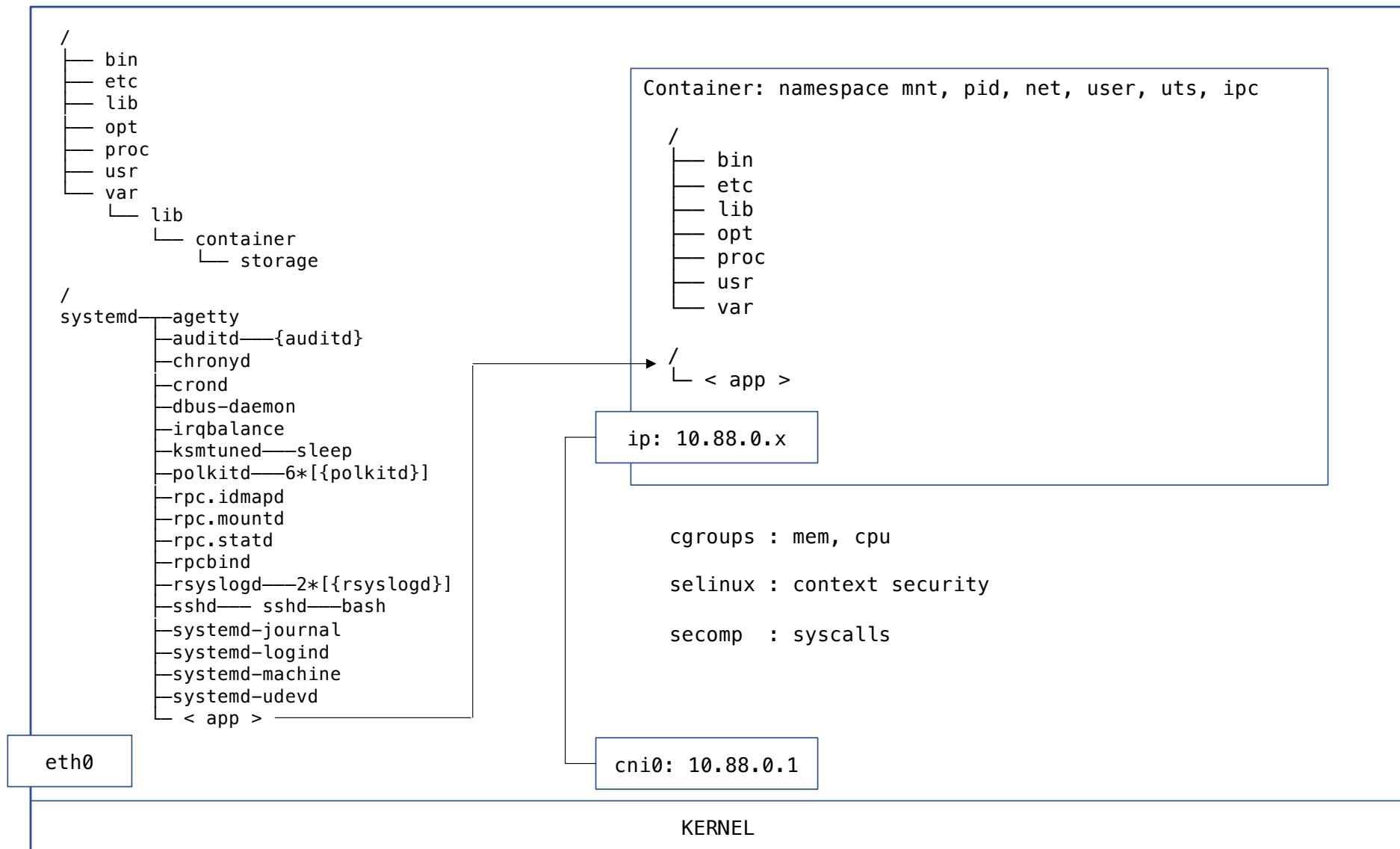
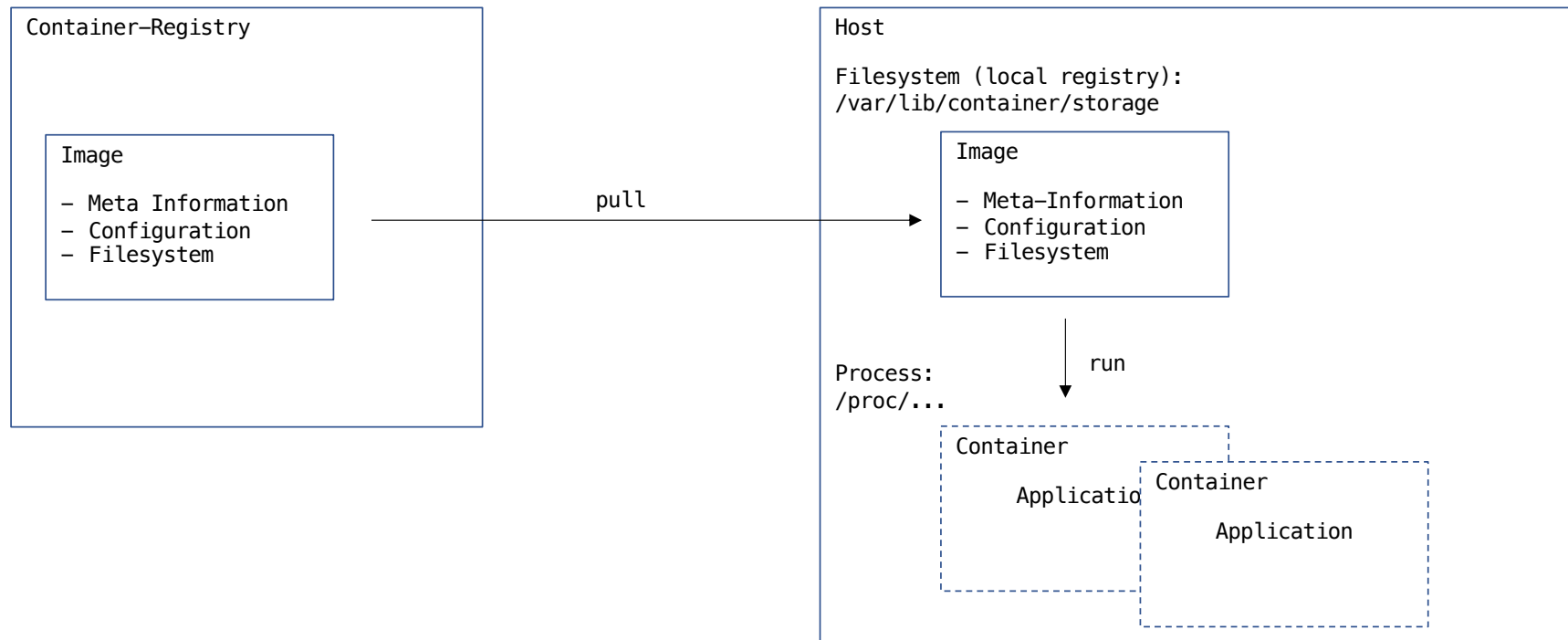


Container:

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





<https://access.redhat.com/RegistryAuthentication>

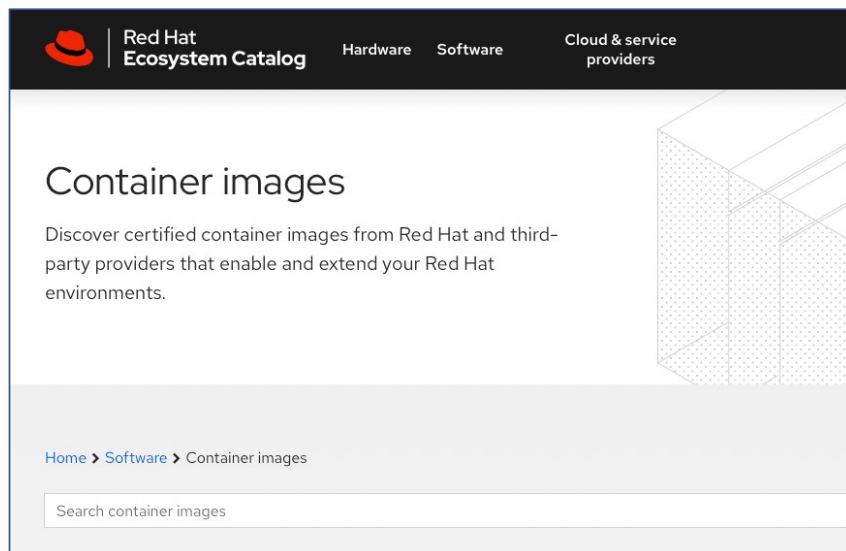
Red Hat Registries

Red Hat distributes container images through three different container registries:

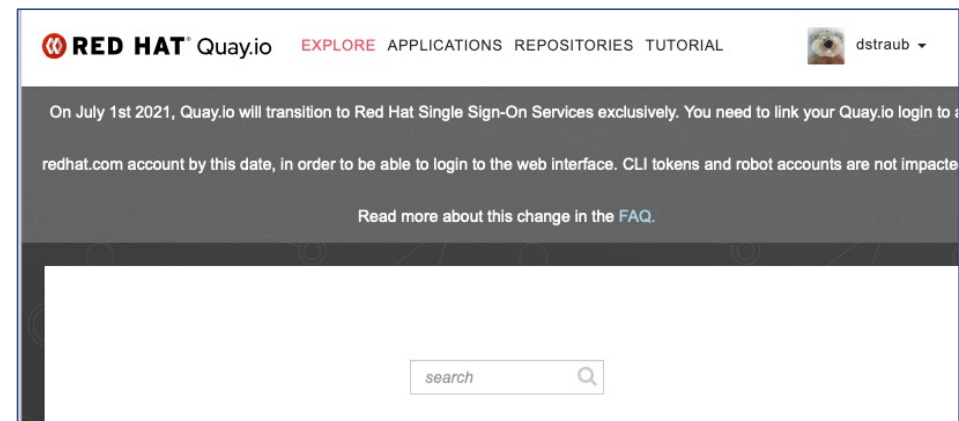
Registry	Content	Supports unauthenticated access	Supports Red Hat login	Supports registry tokens
registry.access.redhat.com	Red Hat products	Yes	No	No
registry.redhat.io	Red Hat products	No	Yes	Yes
registry.connect.redhat.com	Third-party products	No	Yes	Yes

Although both registry.access.redhat.com and registry.redhat.io hold essentially the same container images, some images that require a subscription are only available from registry.redhat.io.

<https://catalog.redhat.com/software/containers/explore>



<https://quay.io>



redhat registries

<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

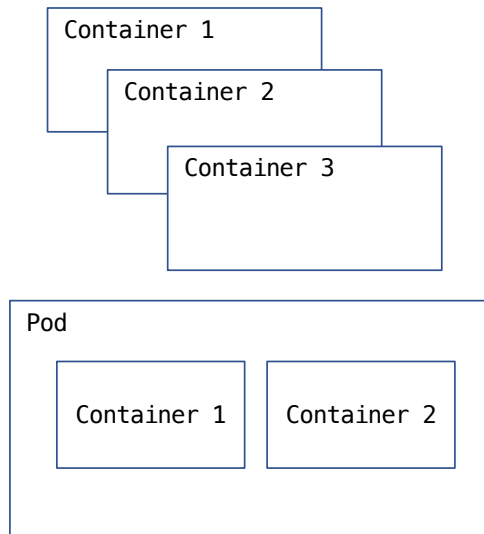
<https://buildah.io>



- Erstellen von Images
- `yum install buildah`

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):

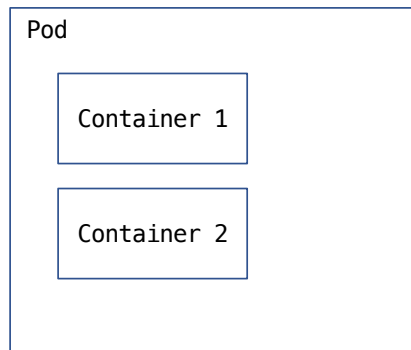
- basiert auf Kubernetes
- Entwickler-Workflow (CI/CD)
- Routing
- Metriken und Log-Management
- einheitliche Benutzeroberfläche

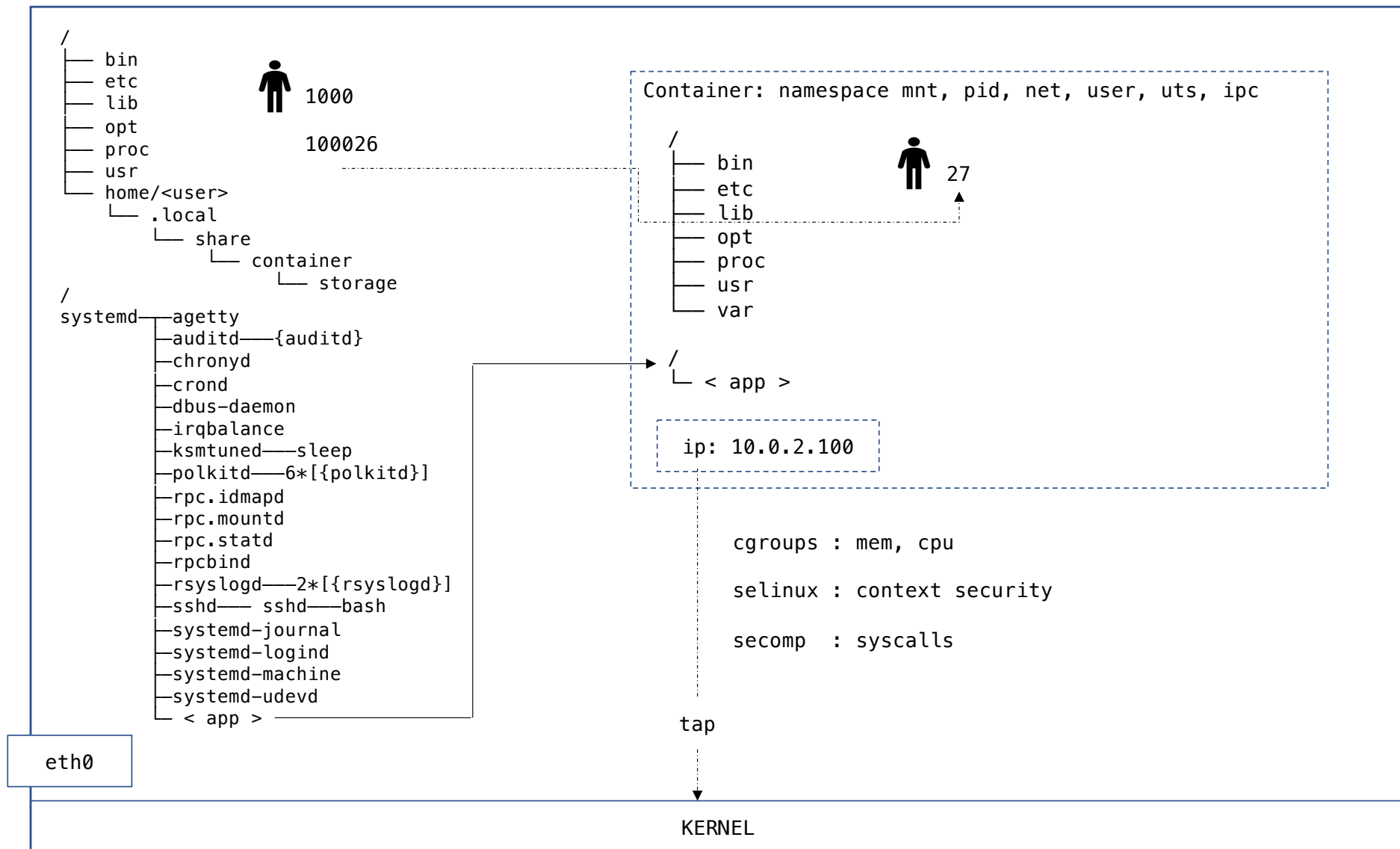
Podman:

- Verwalten von Images und Containern
- mehrere Container können in einen Pod zusammengefasst werden

Kubernetes:

- kleinste Einheit ist der Pod – Gruppe von (unterschiedlichen) Containern
- meistens 1:1 Beziehung (1 Pod enthält ein Container)





Rootless Container (Linux Kernel > v4.18.0)

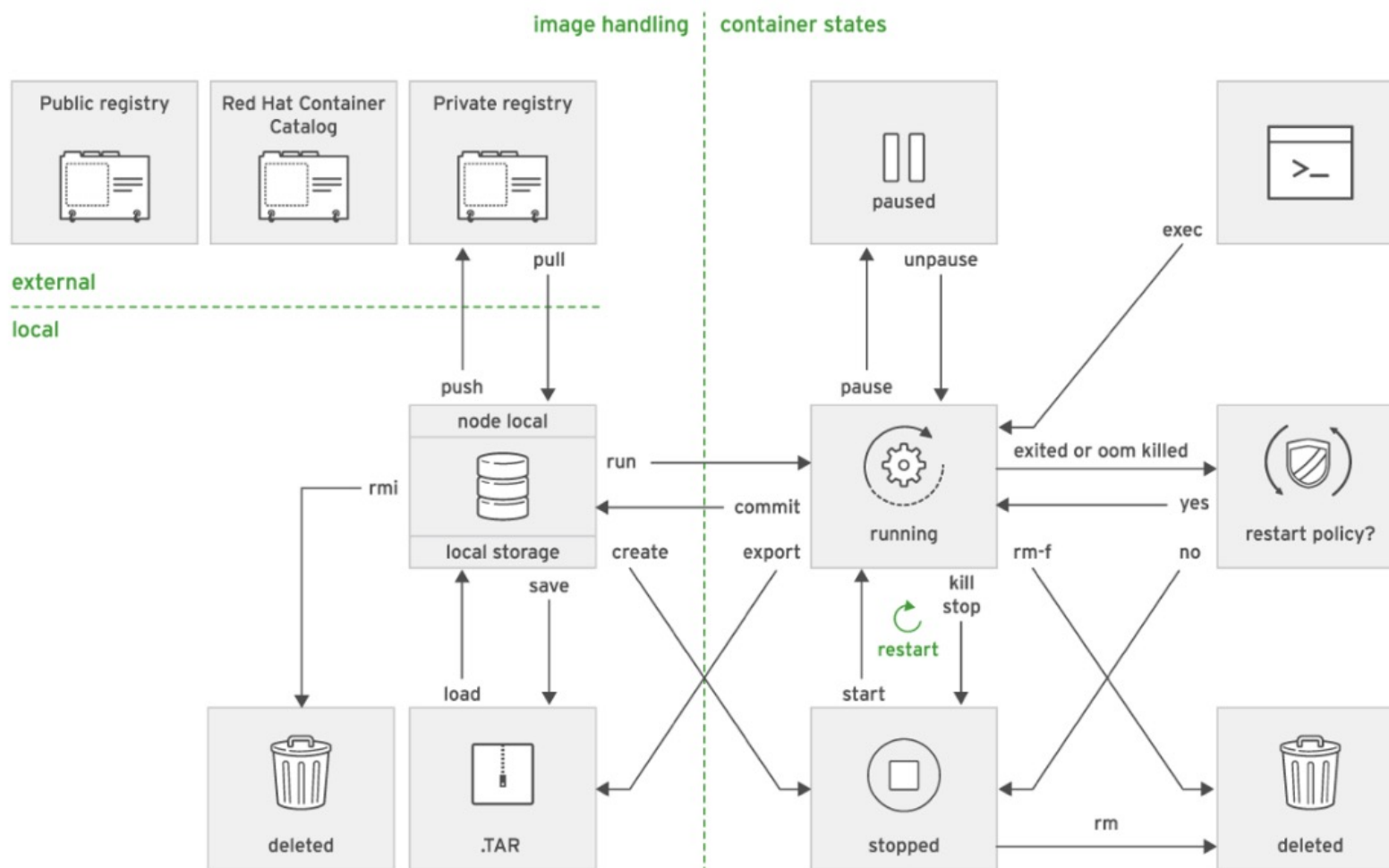
- User-Mapping : /etc/subuid , /etc/subgid
student:100000:65536
(Container-Benutzer root = Host User)

- Fuse – Filesystem statt Overlay2 (~/.local/share/containers)

- TAP – Network Device (keine reale IP-Adresse)

```
tap0: flags=67<UP,BROADCAST,RUNNING> mtu 65520
      inet 10.0.2.100 netmask 255.255.255.0 broadcast 10.0.2.255
      inet6 fe80::6093:deff:febe:f21c prefixlen 64 scopeid 0x20<link>
      ether 62:93:de:be:f2:1c txqueuelen 1000 (Ethernet)
```

https://github.com/containers/podman/blob/main/docs/tutorials/rootless_tutorial.md



Podman managing subcommands

```
# podman run -d --name httpd rhsl/httpd-24-rhel7:2.4-36.8
```

```
# podman ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
4f9e8519685f	.../rhsl/httpd-24-rhel7:2.4-36.8	/usr/bin/run-http...	About a minute ago	Up (14 seconds ago)		httpd

```
# podman exec httpd cat /etc/hosts
```

```
...
172.25.250.9    workstation.lab.example.com workstation
172.25.254.254 classroom.example.com classroom
172.25.250.254 bastion.lab.example.com bastion
10.88.0.18     4f9e8519685f
```

```
# curl -I 10.88.0.18:8080
```

```
# podman pause httpd
```

```
# podman ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
4f9e8519685f	.../rhsl/httpd-24-rhel7:2.4-36.8	/usr/bin/run-http...	2 minutes ago	Paused		httpd

```
# podman unpause httpd
```

```
# podman kill httpd
```

```
# podman logs httpd
```

```
...
[Mon May 17 17:23:42.147898 2021] [lbmethod_heartbeat:notice] [pid 1] AH02282: No slotmem from mod_heartbeat
[Mon May 17 17:23:42.153159 2021] [mpm_prefork:notice] [pid 1] AH00163: Apache/2.4.25 (Red Hat) ... resuming normal operations
[Mon May 17 17:23:42.153196 2021] [core:notice] [pid 1] AH00094: Command line: 'httpd -D FOREGROUND'
```

```
# DIR=$(podman mount httpd)
```

```
# chroot $DIR cat /etc/httpd/conf/httpd.conf
```

```
...
```

```
# sudo podman umount httpd
```

```
# sudo podman rm httpd
```

```
podman stop: sends SIGTERM, [wait -timeout], send SIGKILL
podman kill: sends SIGKILL
podman rm -f -> KILL + rm
```

```
# podman run --name web -d -p 8080:8080 registry.redhat.io/rhel8/httpd-24
# mkdir -p .config/systemd/user
# cd ~/.config/systemd/user/
# podman generate systemd --name web --files --new
# systemctl --user daemon-reload
# systemctl --user enable --now container-web.service
# loginctl enable-linger
```

```
# container-web.service
# autogenerated by Podman 3.3.1
# Tue Jul 26 02:30:53 EDT 2022

[Unit]
Description=Podman container-web.service
Documentation=man:podman-generate-systemd(1)
Wants=network-online.target
After=network-online.target
RequiresMountsFor=%t/containers

[Service]
Environment=PODMAN_SYSTEMD_UNIT=%n
Restart=on-failure
TimeoutStopSec=70
ExecStartPre=/bin/rm -f %t/%n.ctr-id
ExecStart=/usr/bin/podman run --cidfile=%t/%n.ctr-id --sdnotify=common --cgroups=no-conmon --rm --replace --name web -d -p 8080:8080 registry.redhat.io/rhel8/httpd-24
ExecStop=/usr/bin/podman stop --ignore --cidfile=%t/%n.ctr-id
ExecStopPost=/usr/bin/podman rm -f --ignore --cidfile=%t/%n.ctr-id
Type=notify
NotifyAccess=all

[Install]
WantedBy=multi-user.target default.target
```

podman run : Environment

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

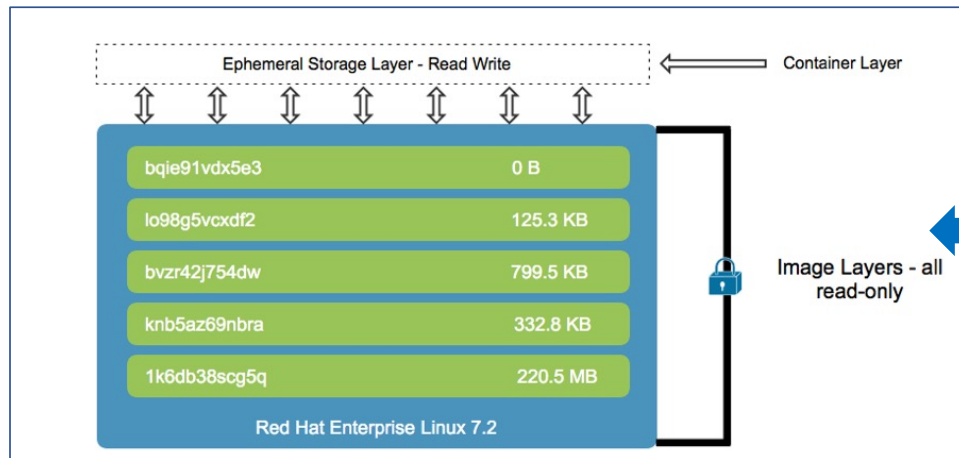
```
podman run --env-file=<host-file>
```

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

podman run : Volumes

```
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>  
podman unshare chmod 0777 <nfs-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 ps
CONTAINER ID   IMAGE                                COMMAND      CREATED        STATUS        PORTS        NAMES
696264634e90   registry.redhat.io/rhel8/mysql-80:1 run-mysqld   5 minutes ago  Up 5 minutes ago          mysql

# podman unshare

# DIR=$(podman mount mysql)
# echo $DIR
/home/student/.local/share/containers/storage/overlay/c9443478c411f51f32a65b12a63c56ffda96ec21e4e40ed38a5cb16e69de1aef/merged

# ls -al /var/lib/mysql/data/
total 106956
drwxrwxr-x. 1 mysql root      4096 Jul 26 08:18 .
drwxrwxr-x. 1 mysql root      102 Jul 26 08:18 ..
-rw-r-----. 1 mysql mysql      2 Jul 26 08:18 696264634e90.pid
-rw-r-----. 1 mysql mysql     56 Jul 26 08:17 auto.cnf
-rw-r-----. 1 mysql mysql   3133 Jul 26 08:18 binlog.000001

# podman umount mysql

# exit

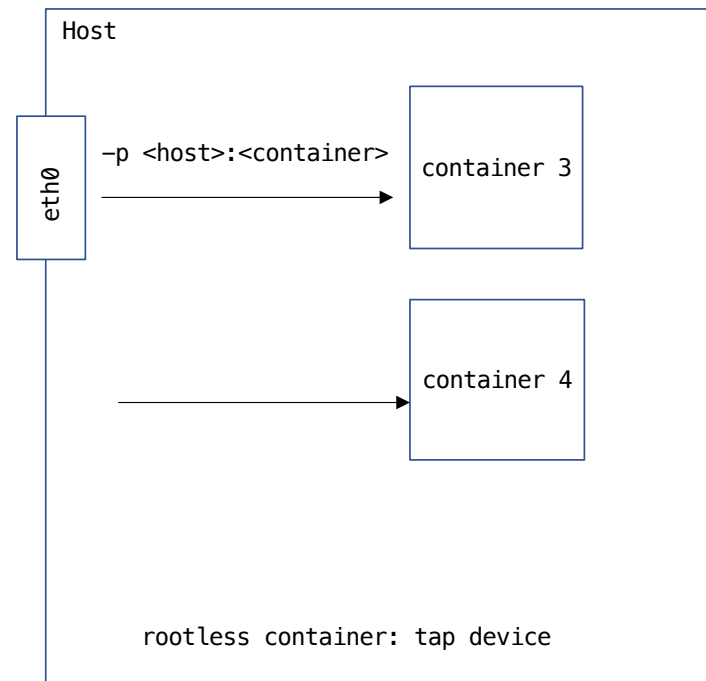
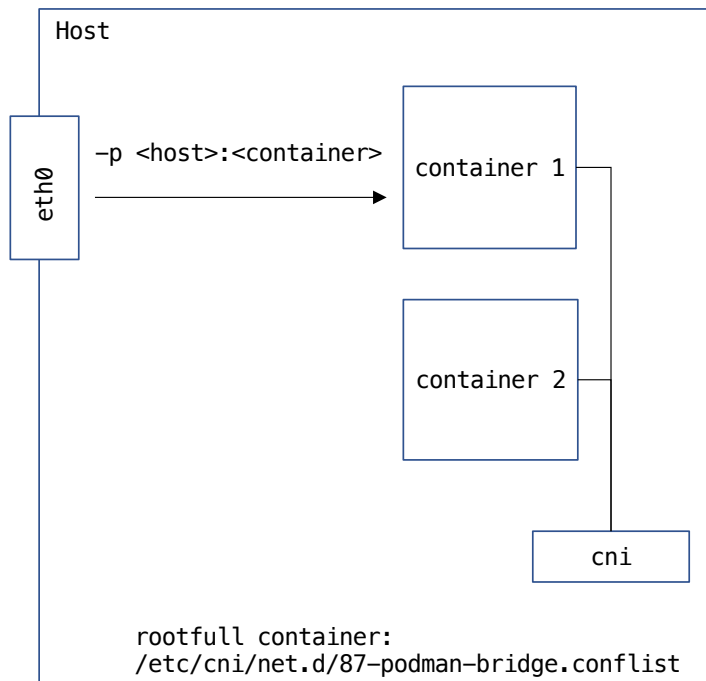
```

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
```

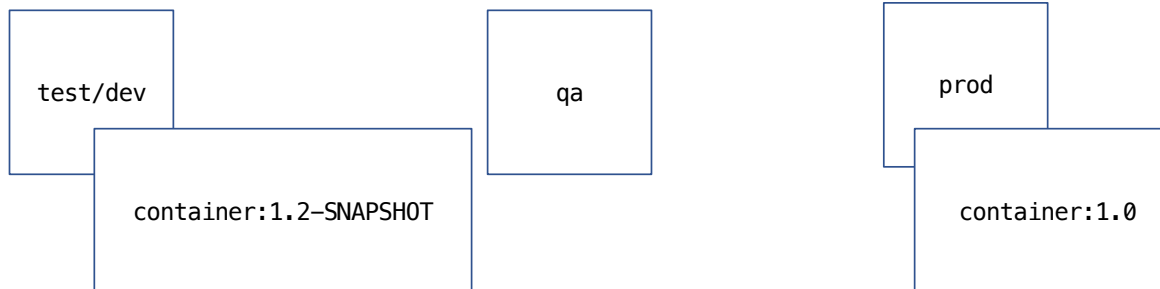
```
$ podman images
REPOSITORY
localhost/nginx
localhost/nginx
<none>
```

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



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 save – Image Operation

erstellt ein TAR von einem Image
(Meta-Informationen, Configuration und Filesystem)

```
$ podman run -d --name ubi ubi7/ubi sleep infinity
82a21f9598b78835566487cb3e9427a9d709ef464813247693c044baa4687b2e
```

```
$ podman ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
82a21f9598b7	registry.access.redhat.com/ubi7/ubi:latest	sleep infinity	11 seconds ago	Up 10 seconds ago		ubi

```
$ podman images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
registry.access.redhat.com/ubi7/ubi	latest	899998a87be7	3 weeks ago	216 MB

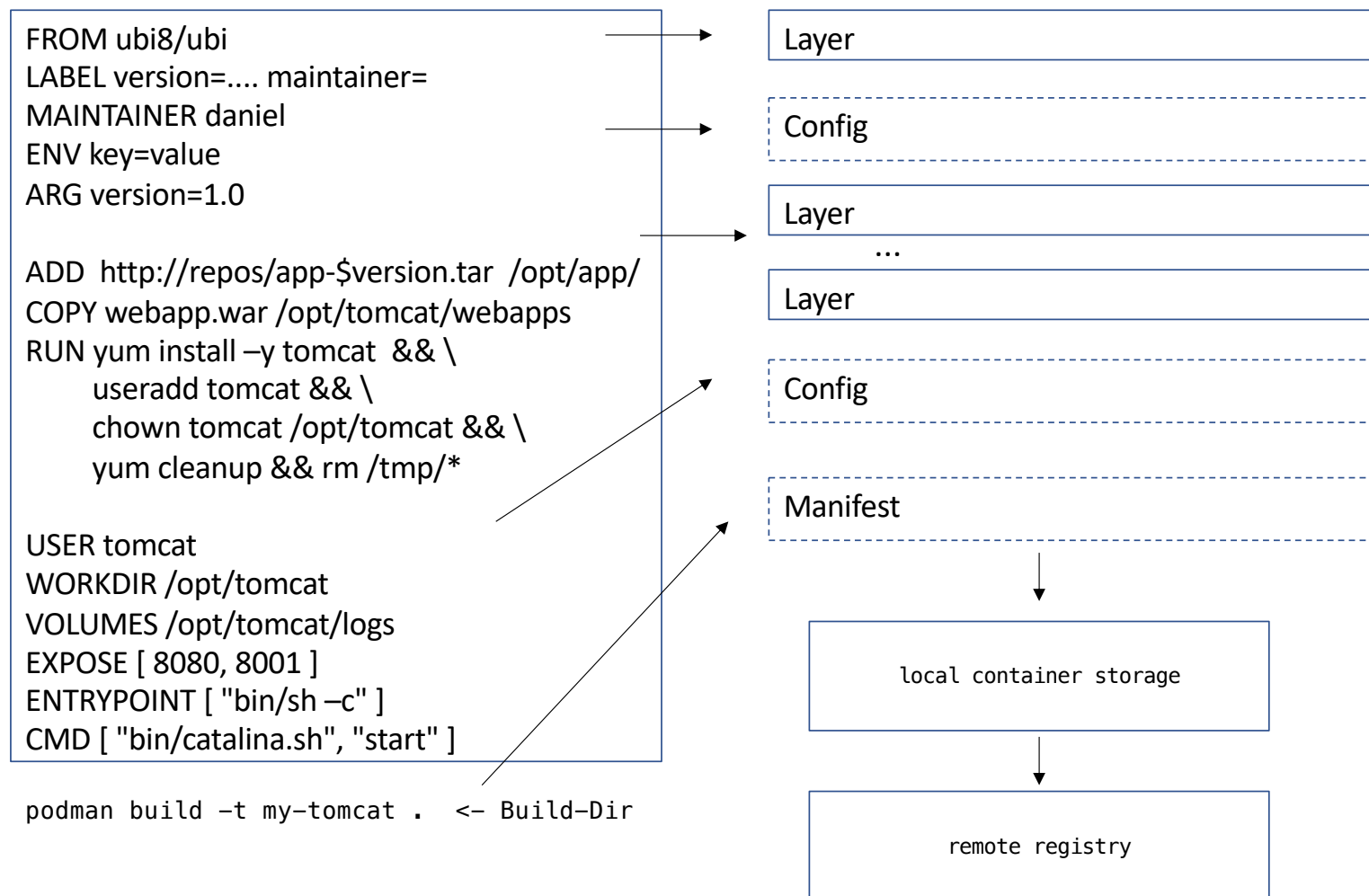
```
$ podman save --output ubi.tar 899
$ tar -tf ubi.tar
123257361dae1cde14e6e5df3b2060adca917932129aae8a26b86c7f1e38b016.tar
c9e02f9d3afeaf029958df4ab4cdce99fc99adabc16c94975967fb5057e932c9.tar
...
repositories
manifest.json
```

```
$ podman export --output ubi-container.tar ubi
$ tar -tf ubi-container.tar
bin
boot/
dev/
etc/
etc/.pwd.lock
etc/DIR_COLORS
...
```

podman export – Container Operation

erstellt ein TAR von einem Container – Filesystem
ohne Meta-Information und Configuration

podman build - Containerfile



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

Verwenden von YUM/DNF beim Image-Build

```
$ sudo 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
...
```

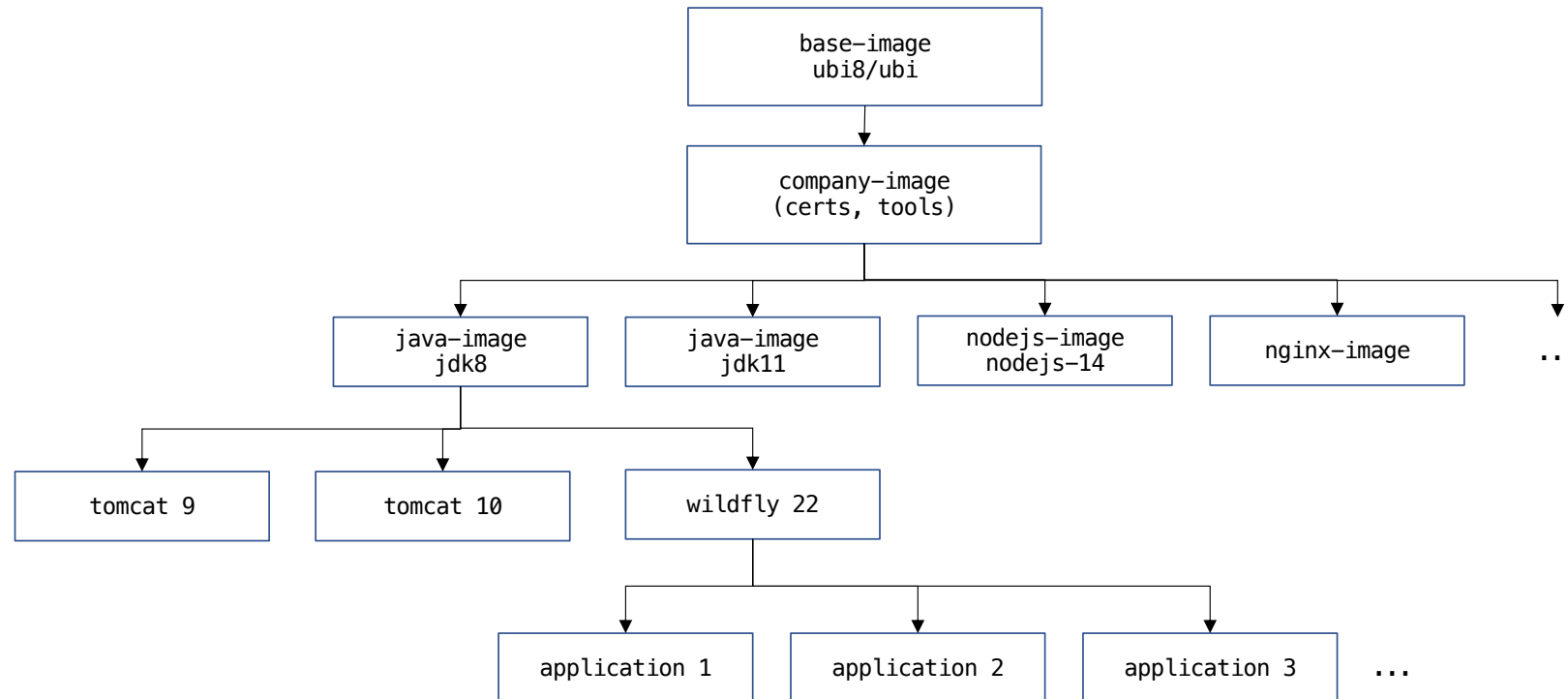
yum "telefoniert"
nach aussen !

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 !

```

[root@workstation ~]# CONTAINER=$(buildah from scratch)
[root@workstation ~]# FS_ROOT=$(buildah mount $CONTAINER)
[root@workstation ~]# echo $FS_ROOT
/var/lib/containers/storage/overlay/62029734ce7a1534208e9b0c07055c35f8c46f7344f2e940afe6bd687feb434a/merged
[root@workstation ~]# ls -alh $FS_ROOT
dr-xr-xr-x. 1 root root 6 Nov 8 06:08 .
drwx-----. 6 root root 69 Nov 8 06:08 ..

[root@workstation ~]# dnf install -y --installroot $FS_ROOT --releasever 8 glibc-minimal-langpack java-11-openjdk-headless
...
Installing:
  glibc-minimal-langpack          x86_64          2.28-189.1.el8
  java-11-openjdk-headless       x86_64          1:11.0.14.1.1-6.el8
Installing dependencies:
...
basesystem                      noarch          11-5.el8
filesystem                      x86_64          3.8-6.el8
...
[root@workstation ~]# tree -d -L 1 $FS_ROOT
/var/lib/containers/storage/overlay/62029734ce7a1534208e9b0c07055c35f8c46f7344f2e940afe6bd687feb434a/merged
├── bin -> usr/bin
├── boot
├── dev
├── etc
├── home
├── lib -> usr/lib
├── lib64 -> usr/lib64
├── ...
└── var

[root@workstation ~]# buildah commit --quiet --squash --rm $CONTAINER java-11:latest
43b1b622db0e59be323038630e7e33d630a80a8e66672861b396be95d7724576
[root@workstation ~]# podman images
REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
localhost/java-11   latest      43b1b622db0e     30 seconds ago  524 MB

[root@workstation ~]# podman run --rm java-11 java -version
openjdk version "11.0.14.1" 2022-02-08 LTS

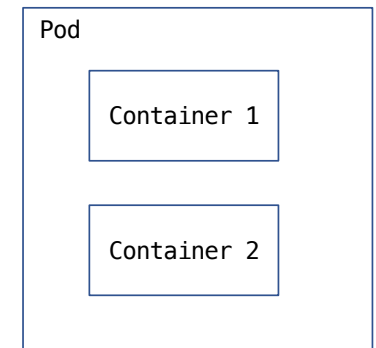
```

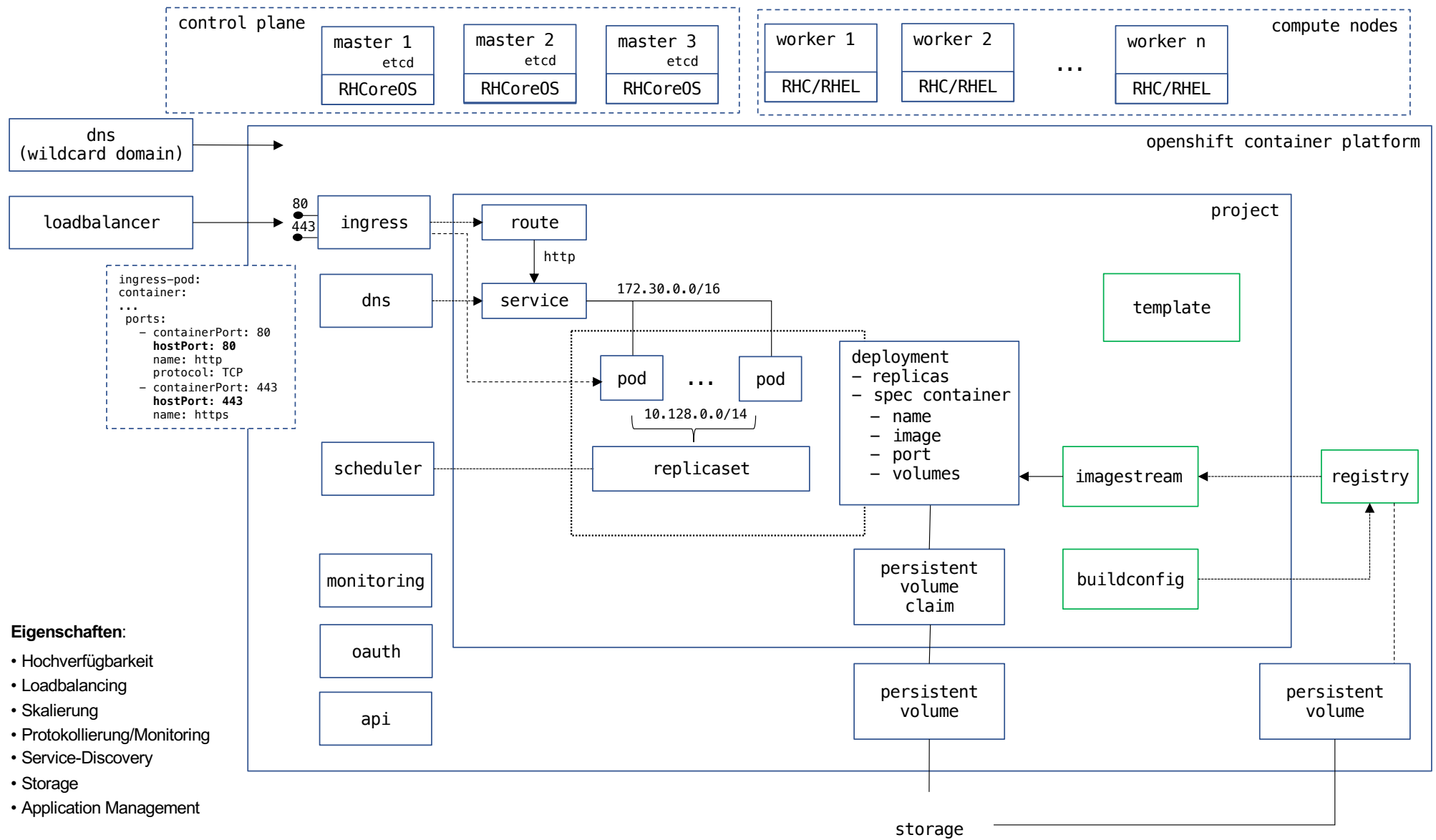
- Openshift
Orchestrierungsservice zur Bereitstellung, Verwaltung und Skalierung von Container-Anwendungen
- Deklaratives System
Status wird in Ressourcen (YAML/JSON) definiert und durch Controller hergestellt
IaC – Infrastructure as Code (<https://blog.nelhage.com/post/declarative-configuration-management>)

```
$ 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)

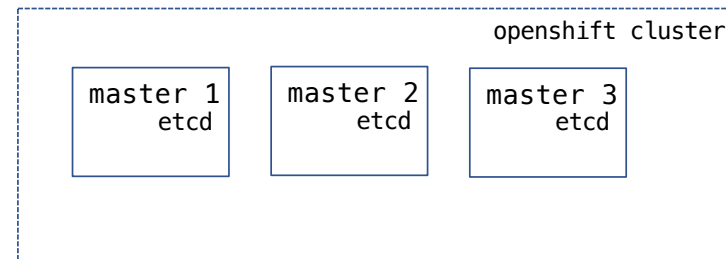




Openshift Resources (Manifest)

```
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

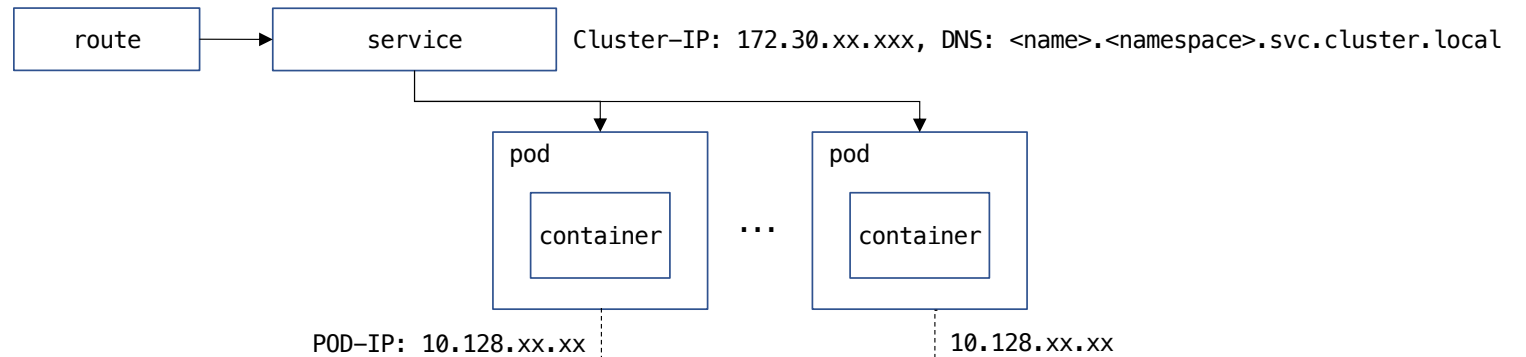
```

```

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
```

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
```

```
$ 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)

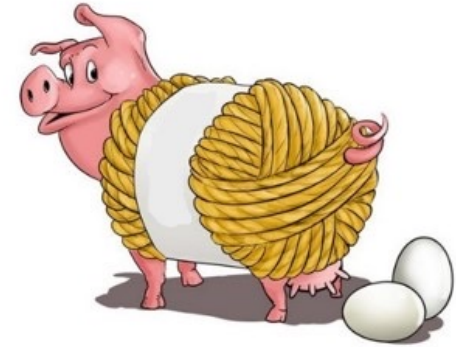


Deployment
Config

Service

Imagestream

BuildConfig



- `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 dc <deploymentconfig>`
 - `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-config>`

```
git clone https://github.com/dstraub/do180-sample.git
```

```
├─ mysql
├─ php-helloworld
└─ webserver
```

Webserver:

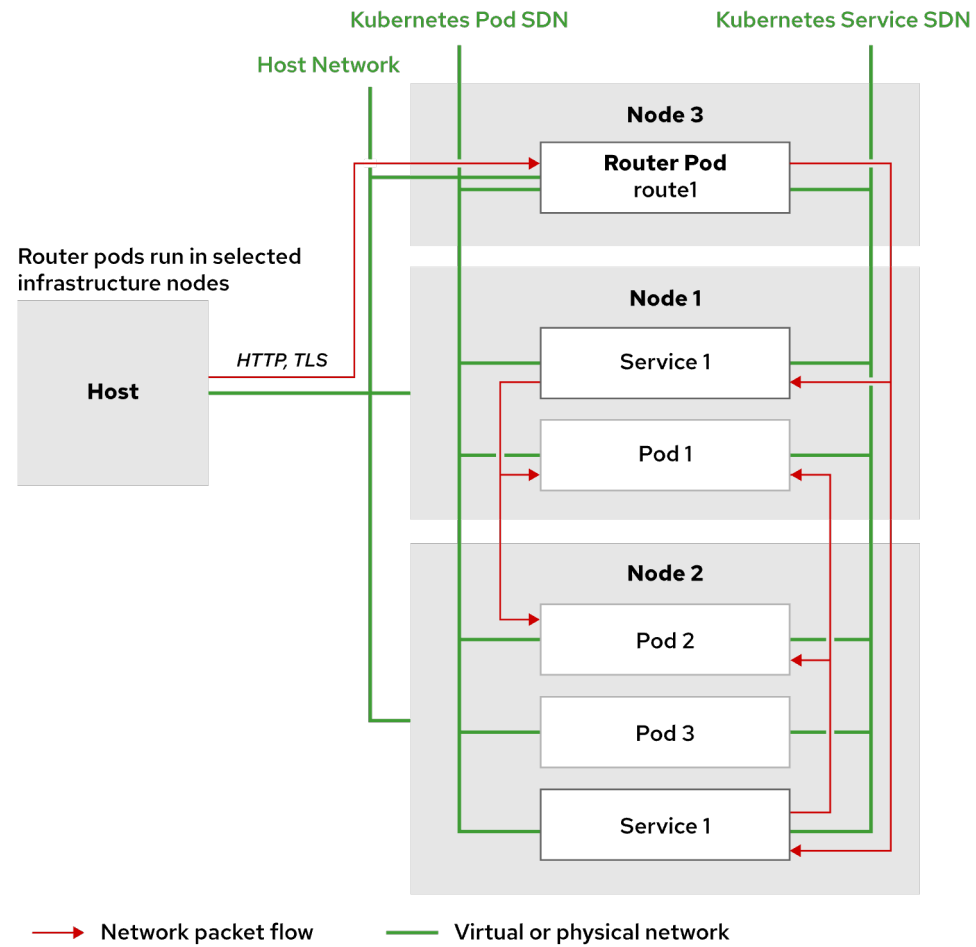
- oc create -f webserver/simple
- oc get all
- curl do180.apps.eu410.prod.nextcle.com

Lab openshift-resources:

- Alternative zu 2: oc create -f mysql
- Aufräumen (bevor 5): oc delete all,pvc --selector app=mysql

Lab openshift-routes:

- Alternative zu 2.1: oc create -f php-helloworld
- Aufräumen: oc delete all --selector app=php-hellworld



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



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

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

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

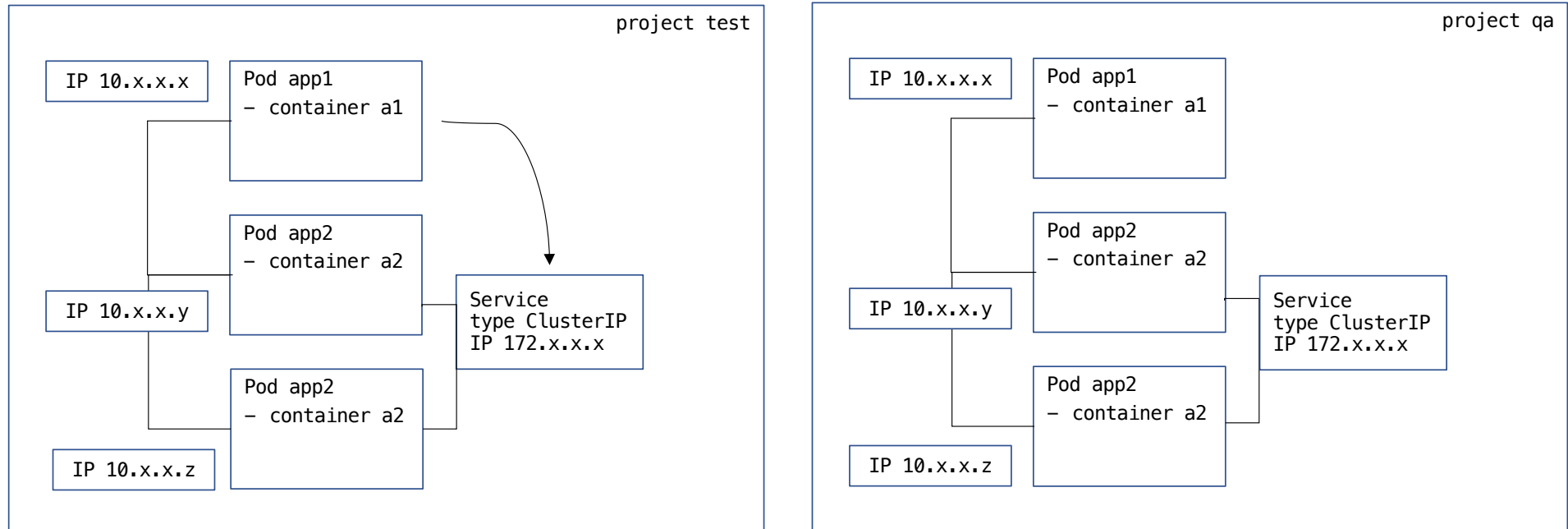
```
$ nslookup dummy.apps.eu45.prod.nextcle.com
Name:      dummy.apps.eu45.prod.nextcle.com
Address: 161.156.16.195
```

```
$ nslookup do180.ctrlaltdel.de
Name:      do180.ctrlaltdel.de
Address: 161.156.16.195
```

← weiterer A-Record auf Wildcard-Domain ...

```
$ curl -H 'Host: do180.ctrlaltdel.de' 161.156.16.195
<html>
<head><title>Index of /</title></head>
...
```

```
$ oc expose service nginx --name do180 --hostname=do180.ctrlaltdel.de
$ curl do180.ctrlaltdel.de
<html>
<head><title>Index of /</title></head>
...
```



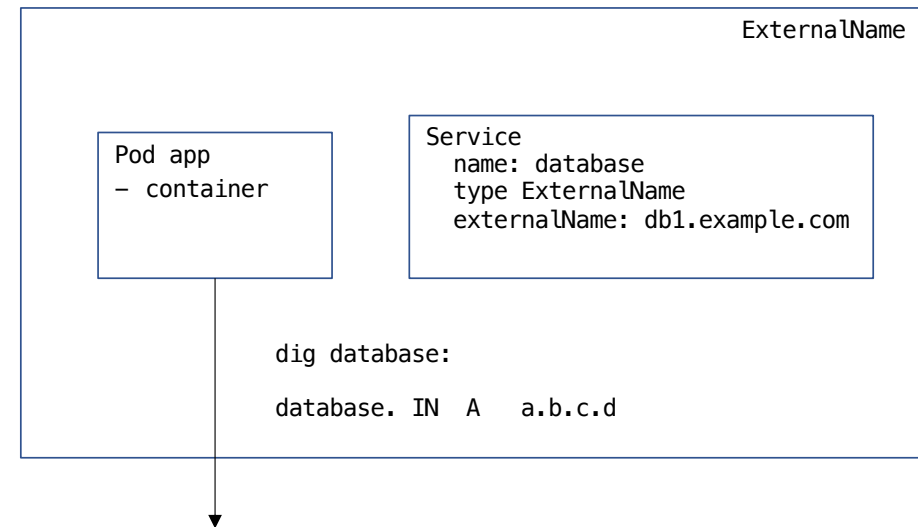
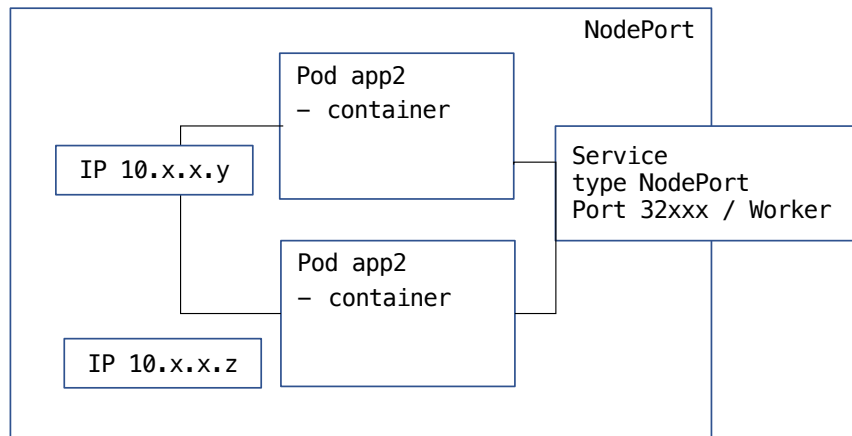
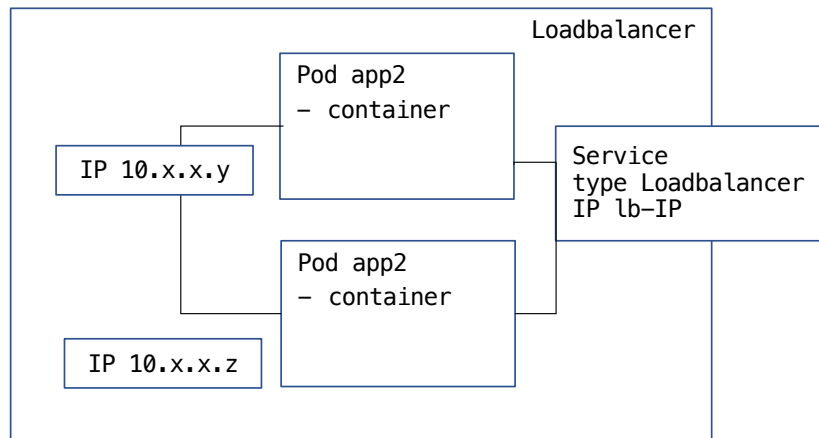
DNS:
A: <service>.test.svc.cluster.local
SVC: _443._tcp.https.<service>.test.svc.cluster.local

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

DNS:
A: <service>.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



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:
    target-port: https
```

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
```

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 !

Imagestream:

- enthält Verweise (Zeiger) auf Images und deren Tags (keine Images)
- automatische Aktualisierung möglich (15 Min. bei externen Registries)
- Verwendung in DeploymentConfig als Image und Trigger

```
$ oc import-image nginx --from=quay.io/<user>/nginx --confirm --scheduled -all
$ oc describe is nginx
Name:                               nginx
Unique Images:                      4
Tags:                               4

latest
  updates automatically from registry quay.io/<user>/nginx:latest
  * quay.io/dstraub/nginx@sha256:c34f57431167fca470730b67a1a8636126d2464eee619ec8d0b577c8e63bffeef

1.2
  updates automatically from registry quay.io/<user>/nginx:1.2
  * quay.io/dstraub/nginx@sha256:ee508edacfe0bc1e6af43a15348b400a7d97121507348bd5fb5effb6b9f8d84e

1.1
  updates automatically from registry quay.io/<user>/nginx:1.1
  * quay.io/dstraub/nginx@sha256:674ab485f6e83f162eb4bdaf12986469c7b4f484f65fbb18f3b03218fd5f36e4

1.0
  updates automatically from registry quay.io/<user>/nginx:1.0
  * quay.io/dstraub/nginx@sha256:693b30b107da...
```

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

Templates – Vorlagen für Resourcedefinitionen mit Parametern

```
kind: Template
apiVersion: v1
metadata:
  name: todonodejs-persistent
objects:
- apiVersion: v1
  kind: Pod
  spec:
    containers:
    - image: quay.io/${RHT_OCP4_QUAY_USER}/do180-mysql-57-rhel7
      name: mysql
    ...
parameters:
- description: Quay namespace the images are stored in
  name: RHT_OCP4_QUAY_USER
  required: true
```

```
$ oc create -f todo-template.yml
```

The screenshot shows the OpenShift Developer Catalog interface. On the left, there's a sidebar with categories like Languages, Databases, Middleware, CI/CD, and Other. The 'All Items' tab is selected. In the main area, the 'todonodejs-persistent' template is highlighted. A modal window titled 'Instantiate Template' is open, showing the 'Namespace' dropdown set to 'danielstraub-template' and the 'RHT_OCP4_QUAY_USER' field. Below the field, it says 'Quay namespace the images are stored in'. To the right of the modal, the namespace 'todonodejs-persistent' is displayed, and a list of resources to be created: PersistentVolumeClaim, Pod, and Service.

Developer Catalog

Add shared apps, services, or source-to-image builders to your project from the Developer Catalog. automatically.

All Items

Languages
Databases
Middleware
CI/CD
Other

Type

☐ Operator Backed (0)
☐ Helm Charts (0)

todonodejs-persistent

Instantiate Template

Namespace *
PR danielstraub-template

RHT_OCP4_QUAY_USER *

Quay namespace the images are stored in

todonodejs-persistent

The following resources will be created:

- PersistentVolumeClaim
- Pod
- Service

Create **Cancel**

Ausgabe auf stdout:

```
$ oc process todonodejs-persistent -p RHT_OCP4_QUAY_USER=... -o yaml
```

Verarbeitung:

```
$ oc process todonodejs-persistent -p RHT_OCP4_QUAY_USER=... | oc create -f -
```

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
...
```

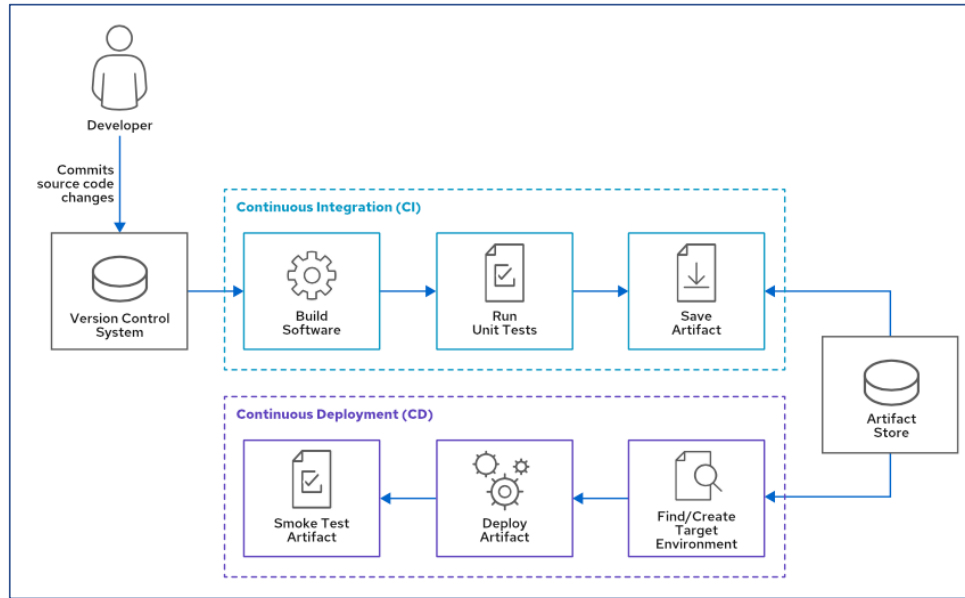
https://kubectl.docs.kubernetes.io/guides/extending_kustomize/exec_krm_functions

```
$ 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
```



Continuous Integration
Continuous Delivery

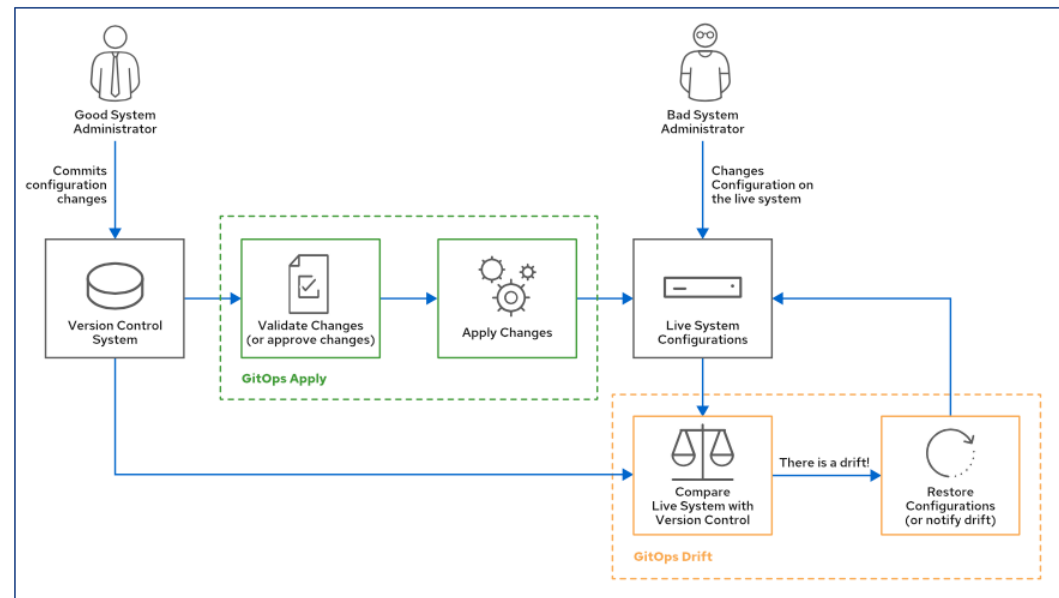
→ Developer
→ running application

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

GitOps Workflow

→ Administrators
→ live System

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



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	⋮

Serviceaccounts

- werden als Identität für Prozesse (Pods/Container) verwendet
- 'technische Benutzer' in Kubernetes
- beim Anlegen eines Namespace wird ein 'default' – Serviceaccount angelegt (+ Token)
- Empfehlung: dedizierten Serviceaccount verwenden

```
$ oc create sa do180
$ oc get sa do180 -o yaml
kind: ServiceAccount
apiVersion: v1
metadata:
  name: do180
  namespace: do180
imagePullSecrets:
- name: do180-dockercfg-ps2gj  ← Zugriff auf interne Container-Registry
secrets:
- name: do180-token-fwpsv
- name: do180-dockercfg-ps2gj

$ SECRET_NAME=$(oc get sa do180 -o json | jq -r ".secrets[] | select(.name|startswith(\"do180-token\")).name")
$ SERVICE_ACCOUNT_TOKEN=$(oc get secret $SECRET_NAME -o jsonpath='{.data.token}' | base64 -d)

$ oc --token $SERVICE_ACCOUNT_TOKEN whoami
$ system:serviceaccount:do180:do180
```

Verwenden eines Serviceaccounts im Deployment:

```
apiVersion: apps/v1
kind: Deployment
...
spec:
  ...
  template:
    spec:
      serviceAccountName: do180
```

→ Serviceaccount-Token ist in jedem Container des Deployments (Pods) !

```
oc exec test-68fb6d564c-46zhm -- cat /var/run/secrets/kubernetes.io/serviceaccount/token | cut -d . -f 2 | base64 -d
{
```

```
  "aud": [
    "https://kubernetes.default.svc"
  ],
  "exp": 1699567083,
  "iat": 1668031083,
  "iss": "https://kubernetes.default.svc",
  "kubernetes.io": {
    "namespace": "do180",
    "pod": {
      "name": "test-68fb6d564c-46zhm",
      "uid": "91e8b445-8fb7-4755-a45b-958e917bdca8"
    },
    "serviceaccount": {
      "name": "do180",
      "uid": "0ac431f9-4571-49cb-b7d1-863ec3a75513"
    },
    ...
  }
}
```

```
date -Is -d '@1699567083'
2023-11-09T16:58:03-05:00
```

Hashicorp Vault - Auth-Method "kubernetes": Serviceaccount eines Pods wird zur Authentifizierung verwendet

- SA-Token wird validiert, zur Validierung kann der gleiche Serviceaccount oder ein anderer verwendet werden (benötigt ClusterRole 'system:auth-delegator')
- in Vault wird dann an den diesen Service-Account eine Rolle/Policies gebunden, die den Zugriff auf die Secrets regeln

The image shows a composite of three screenshots from the Hashicorp Vault web interface. The top-left screenshot shows the 'kubernetes' auth method configuration page with tabs for 'Roles' and 'Configuration'. The bottom-left screenshot shows the 'do180' role configuration page with fields for 'Alias name source', 'Audience', 'Bound service account names', and 'Bound service account namespaces'. The right screenshot is a detailed view of the 'Configure Kubernetes' configuration page, showing the 'Kubernetes host' field with the value 'https://172.30.0.1' and the 'Kubernetes CA Certificate' field with a file upload button. A text box on the right side of the configuration page states: 'ist Vault im gleichen Cluster muss nur die Adresse des API-Servers konfiguriert werden !'.

ACCESS

kubernetes

Auth Methods

Multi-factor authentication

Entities

Groups

Roles

Configuration

do180

ist Vault im gleichen Cluster muss nur die Adresse des API-Servers konfiguriert werden !

Configure Kubernetes

Configuration

Method Options

Kubernetes host ⓘ

https://172.30.0.1

Kubernetes CA Certificate ⓘ

Choose a file... No file chosen

Select a file from your computer

ACCESS

do180

Auth Methods

Multi-factor authentication

Entities

Groups

Leases

OIDC Provider

Alias name source serviceaccount

Audience

Bound service account names do180

Bound service account namespaces do180

Verwendung im Deployment

- entweder als "InitContainer" zum einmaligen Auslesen von Vault-Token/Daten (über Vault-Agent/Template) oder als "SidecarContainer" wenn Vault-Token/Daten ständig erneuert werden müssen
- Datenaustausch zwischen "Vault-Agent Container" und "Anwendung" erfolgt über emptyDir (<https://kubernetes.io/docs/concepts/storage/volumes/#emptydir>)

```
apiVersion: apps/v1
kind: Deployment
...
spec:
  ...
  template:
    spec:
      serviceAccountName: do180
      initContainers:
        - name: agent
          image: quay.io/danielstraub/vault
          command:
            - vault
          args:
            - agent
            - -config=/etc/vault/vault.conf
          volumeMounts:
            - name: data
              mountPath: /data
            - name: vault-conf
              mountPath: /etc/vault
      containers:
        - name: test
          image: ...
          volumeMounts:
            - name: data
              mountPath: /data
      volumes:
        - name: data
          emptyDir: {}
        - name: vault-conf
          configMap:
            name: vault
            items:
              - key: vault-agent
                path: vault.conf
```

<https://developer.hashicorp.com/vault/tutorials/auth-methods/agent-kubernetes>
<https://developer.hashicorp.com/vault/docs/auth/kubernetes>

Einfacher Vault-Container:

```
#!/bin/bash

CONTAINER=$(buildah from scratch)
FS_ROOT=$(buildah mount $CONTAINER)

dnf install -y --installroot $FS_ROOT --releasever 8 --nodocs --quiet --setopt=install_weak_deps=False \
    glibc-minimal-langpack jq bash curl

curl -sL https://releases.hashicorp.com/vault/1.12.1/vault_1.12.1_linux_amd64.zip -o /tmp/vault.zip
unzip /tmp/vault.zip -d $FS_ROOT/usr/local/bin
chmod +x $FS_ROOT/usr/local/bin/vault
rm -f /tmp/vault.zip

buildah commit --quiet --rm --squash $CONTAINER quay.io/danielstraub/vault
podman push quay.io/danielstraub/vault
```