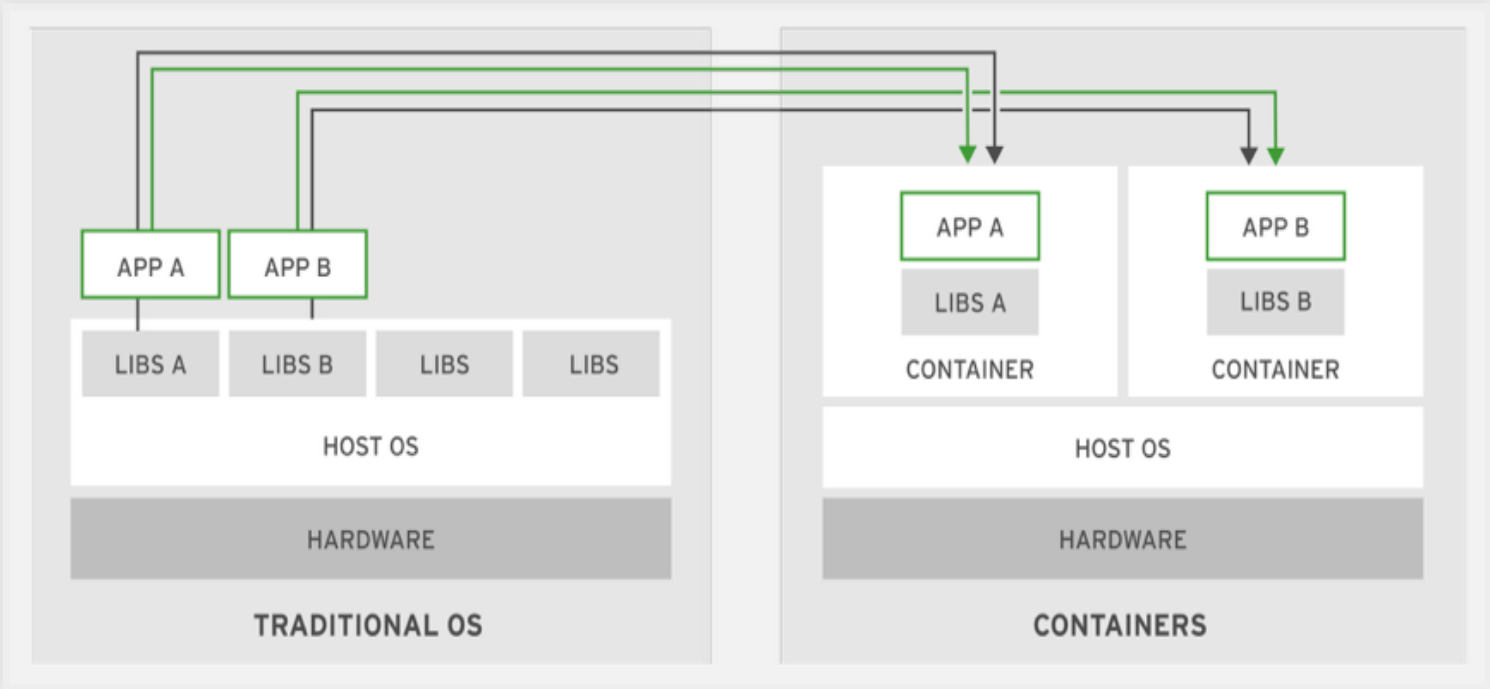


Container versus operating system differences

Container:

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

```
/
├── bin
├── etc
├── lib
├── opt
├── proc
├── usr
└── var
    └── lib
        └── container
            └── storage
```

```
/
systemd─┬─agetty
        ├─auditd───{auditd}
        ├─chronyd
        ├─crond
        ├─dbus-daemon
        ├─irqbalance
        ├─ksmtuned───sleep
        ├─polkitd───6*[{polkitd}]
        ├─rpc.idmapd
        ├─rpc.mountd
        ├─rpc.statd
        ├─rpcbind
        ├─rsyslogd───2*[{rsyslogd}]
        ├─sshd─── sshd───bash
        ├─systemd-journal
        ├─systemd-logind
        ├─systemd-machine
        ├─systemd-udevd
        └─ < app >
```

```
Container: namespace mnt, pid, net, user, uts, ipc

  /
  ├── bin
  ├── etc
  ├── lib
  ├── opt
  ├── proc
  ├── usr
  └── var

  /
  └─ < app >  8080
```

ip: 10.88.0.x

cgroups : mem, cpu

selinux : context security

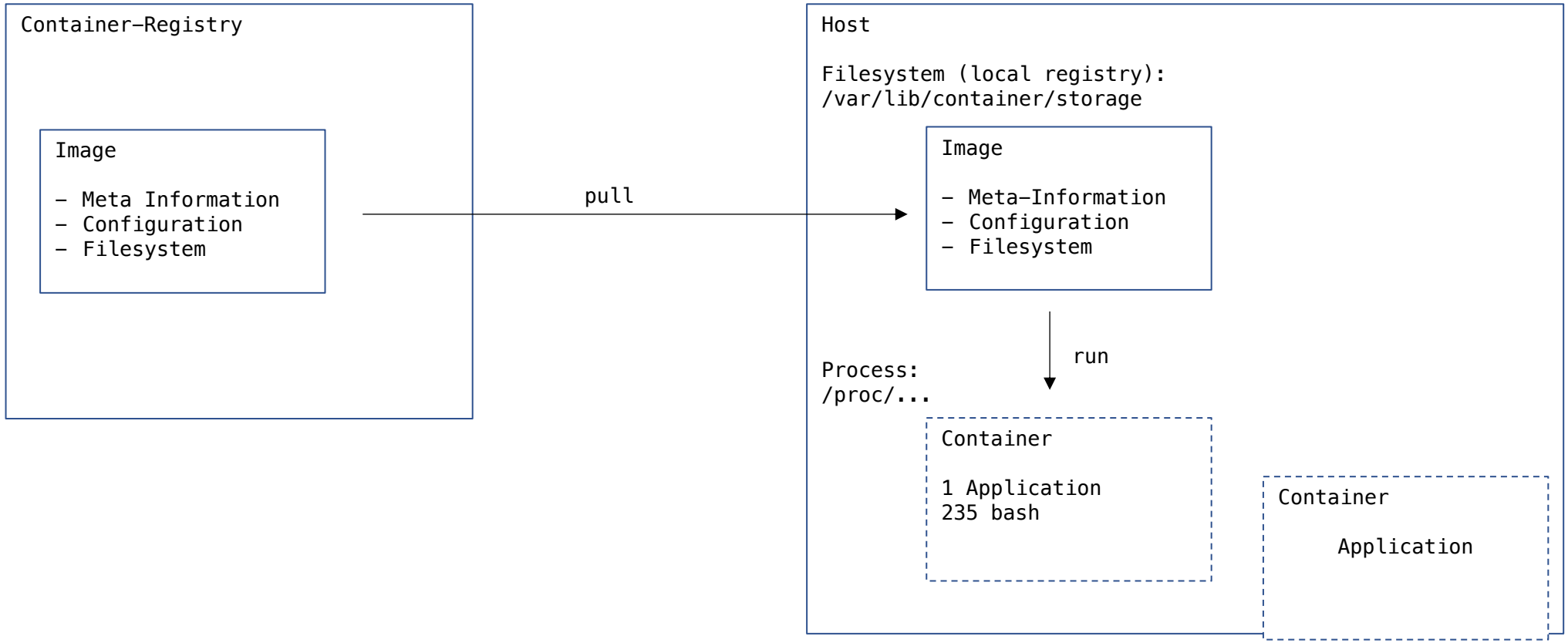
secomp : syscalls

eth0

cni0: 10.88.0.1

KERNEL

rootfull container


Container-Registry
Image
- Meta Information
- Configuration
- Filesystem
pull
Host
Filesystem (local registry):
/var/lib/container/storage
Image
- Meta-Information
- Configuration
- Filesystem
run
Process:
/proc/...
Container
1 Application
235 bash
Container
Application

https://access.redhat.com/RegistryAuthentication

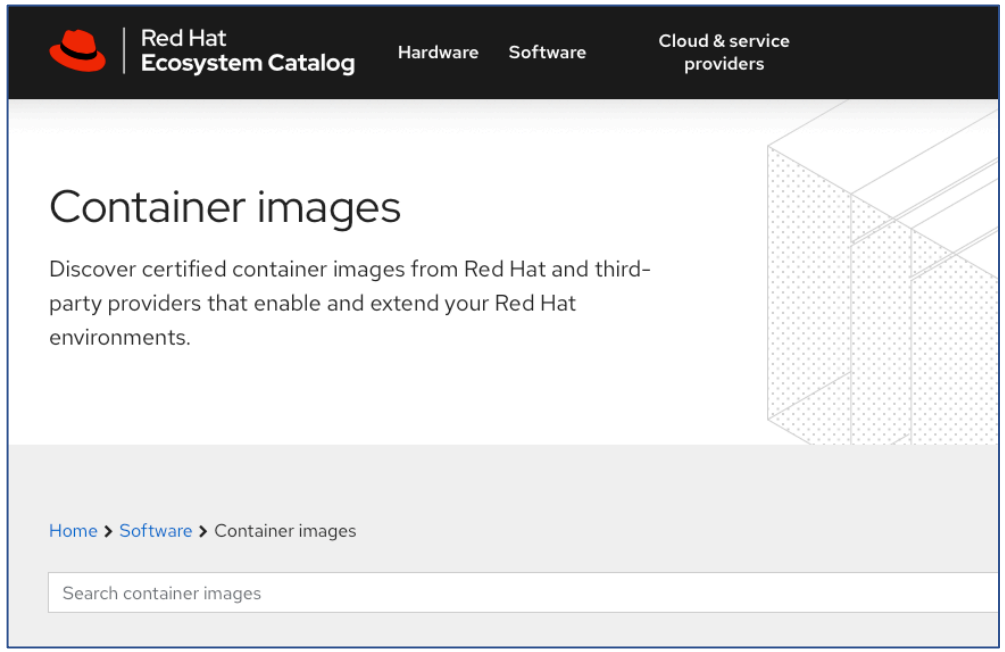
Red Hat Registries

Red Hat distributes container images through three different container registries:

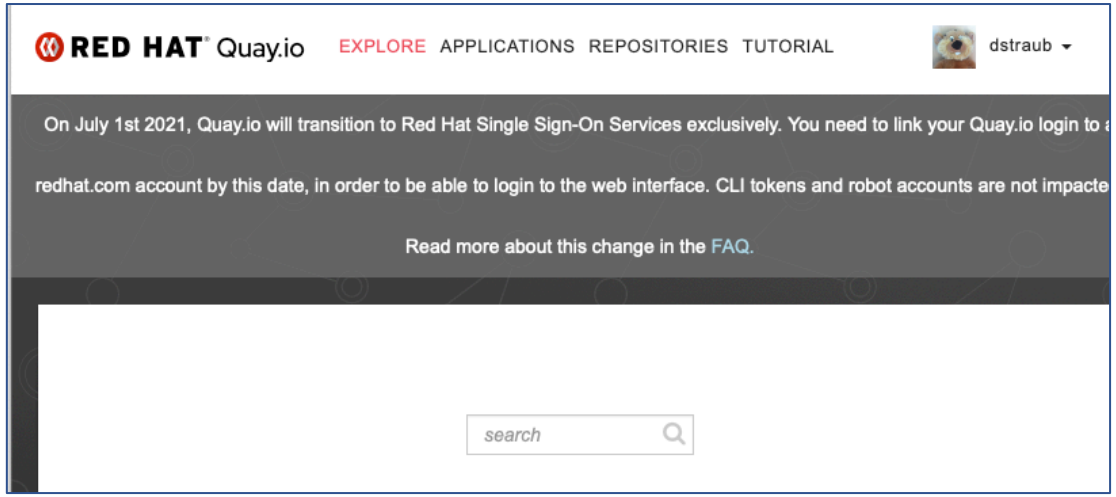
Registry	Content	Supports unauthenticated access	Supports Red Hat login	Supports registry tokens
registry.access.redhat.com	Red Hat products	Yes	No	No
registry.redhat.io	Red Hat products	No	Yes	Yes
registry.connect.redhat.com	Third-party products	No	Yes	Yes

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

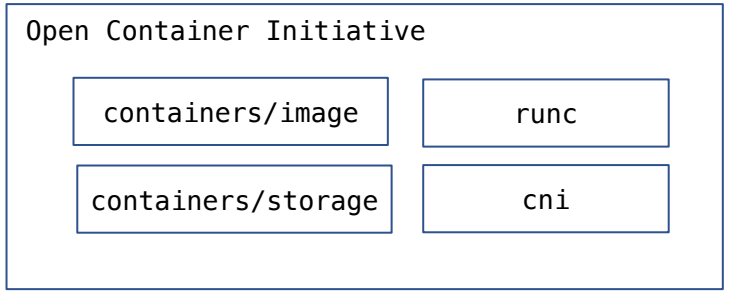


https://quay.io



https://podman.io

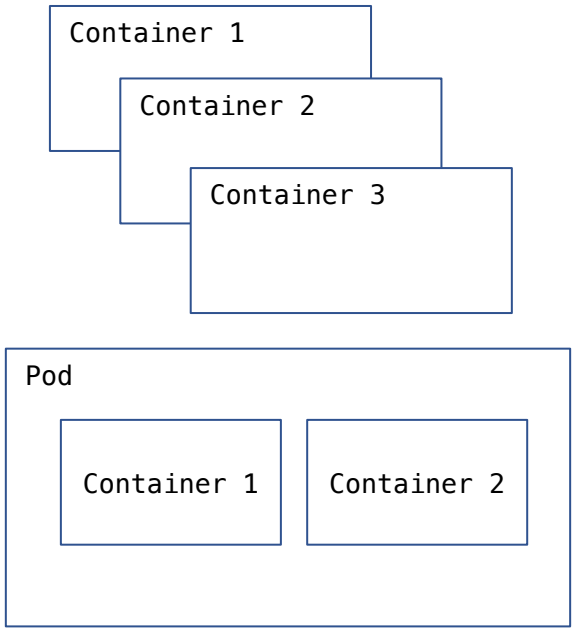
- Image- und Containermanagement
- OCI: Open Container Initiative
- keine Client/Serverarchitektur
- gleiche Befehlssyntax wie do...
- Kubernetes kompatibel
- yum install podman



https://buildah.io

- Erstellen von Images
- yum install buildah

Podman :
großer Aufwand beim Betrieb mehrerer Container, Serivce-Kommunikation, Routing



Kubernetes : Orchestrierung von Container-Anwendungen

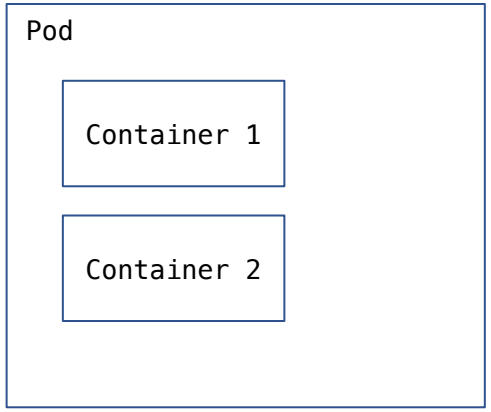
- Service Discovery, Loadbalancing
- Horizontale Skalierung
- Health Checks
- Rolling Updates
- Secret/Configmanagement
- Operatoren: native Kubernetes Anwendungen zum Cluster- und Anwendungs-Management

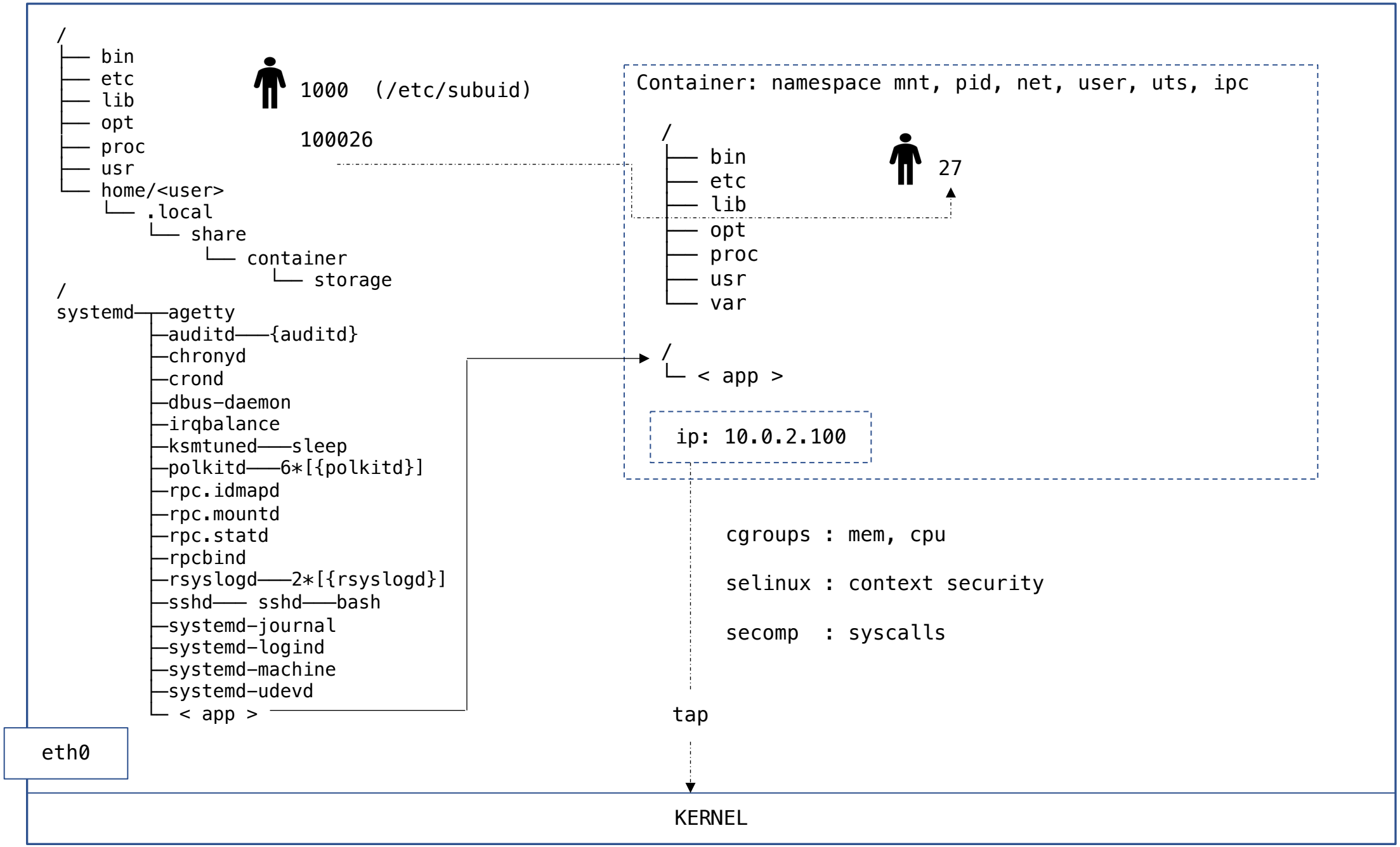
Openshift (RHOCP):

- 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

Rootless Container (Linux Kernel > v4.18.0)

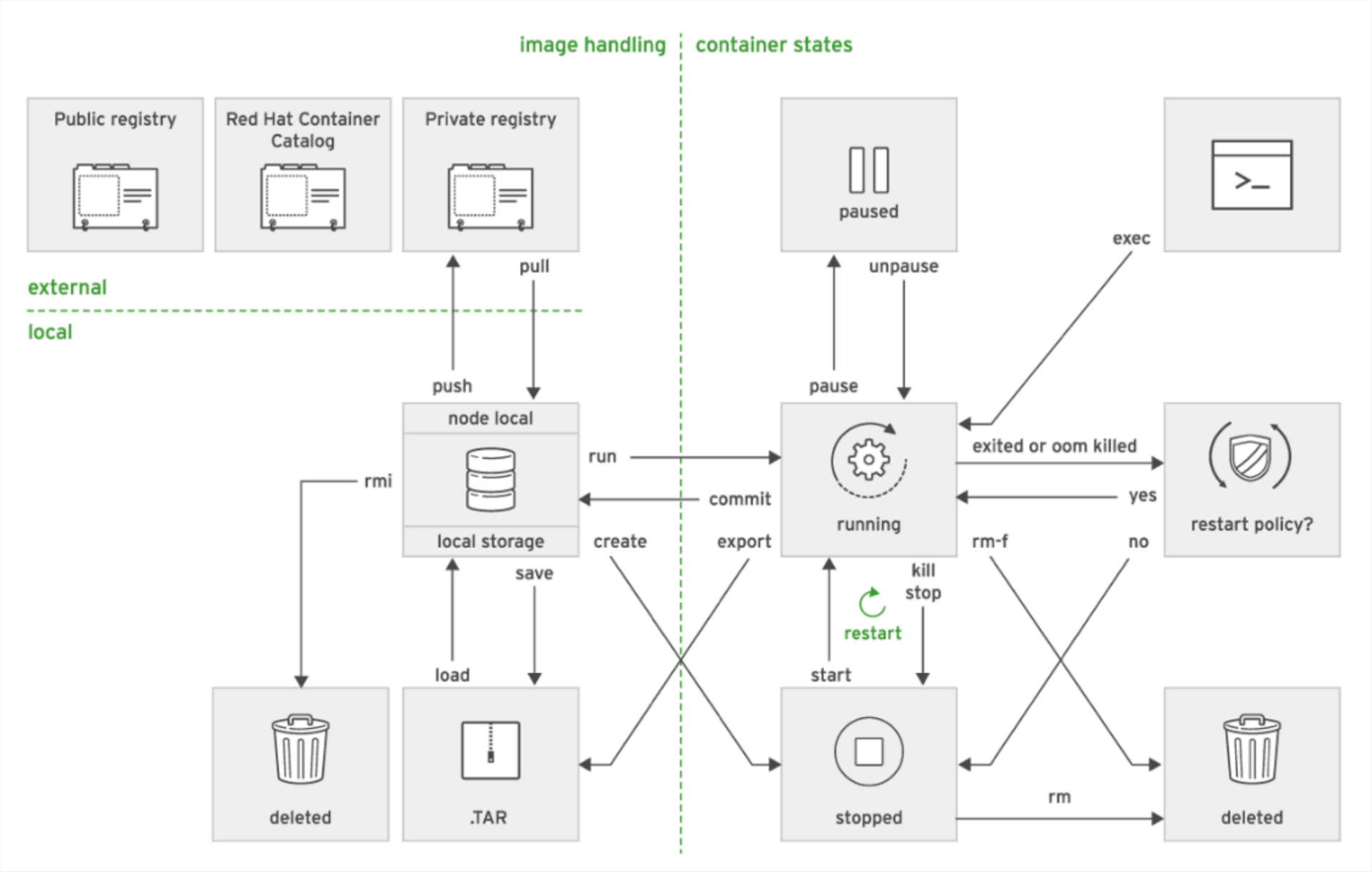
- User-Mapping : /etc/subuid , /etc/subgid
  `student:100000:65536`
  (Container-Benutzer root = Host User)

- 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 commands

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

# podman ps
CONTAINER ID  IMAGE                                COMMAND              CREATED             STATUS                PORTS  NAMES
4f9e8519685f  .../rhscl/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  .../rhscl/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_heartmonitor
[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'


# podman rm httpd
```

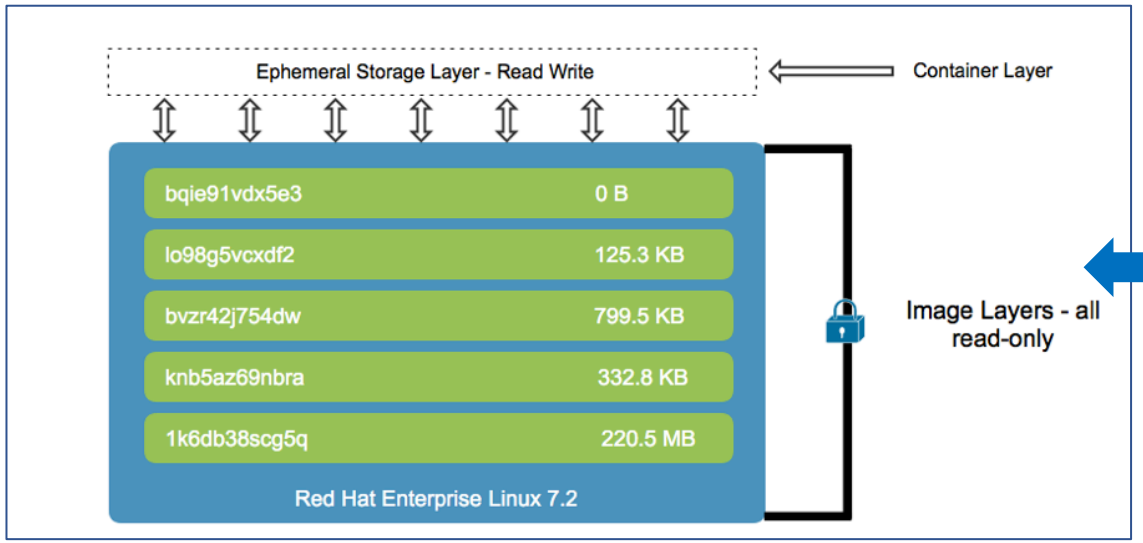
```
podman stop: sends SIGTERM, [wait -timeout], send SIGKILL
podman kill: sends SIGKILL
podman rm -f -> KILL + rm
```

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:

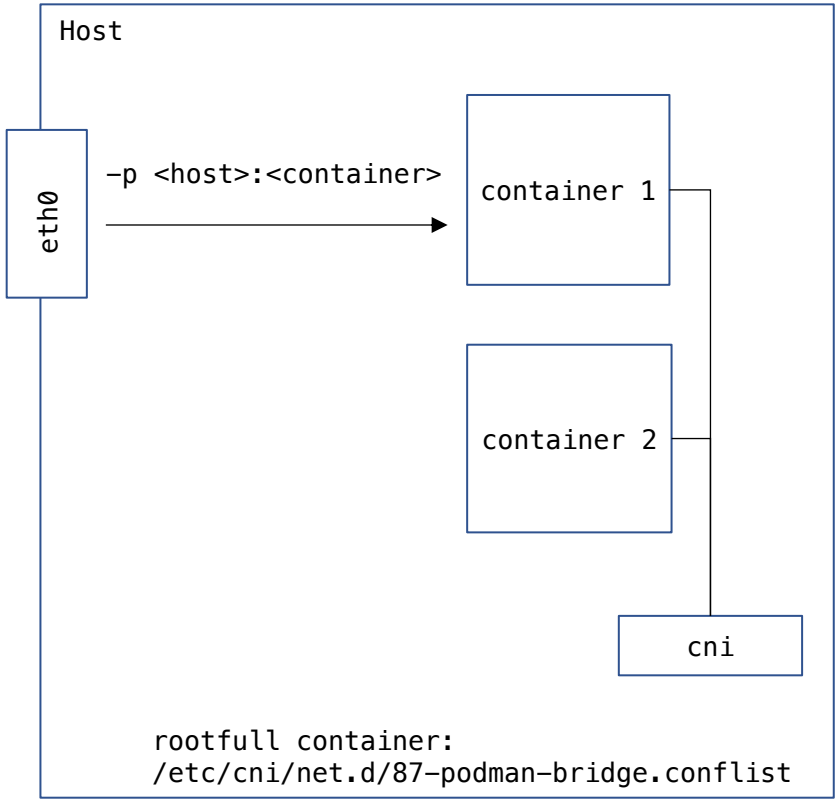
- Permissions
  chown -R <container-userid> <host-dir>
  rootless: podman unshare chown -R <container-userid> <host-dir>

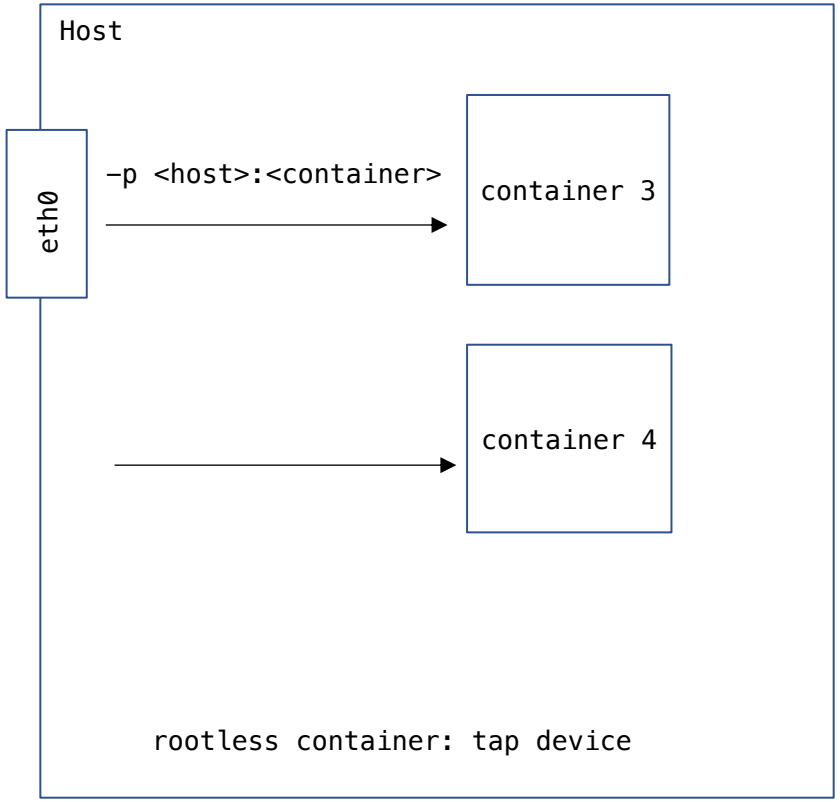
- SELinux
  semange fcontext -a -t container_file_t '<host-dir>(/.*)?'
  restorecon -Rv <dir>

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

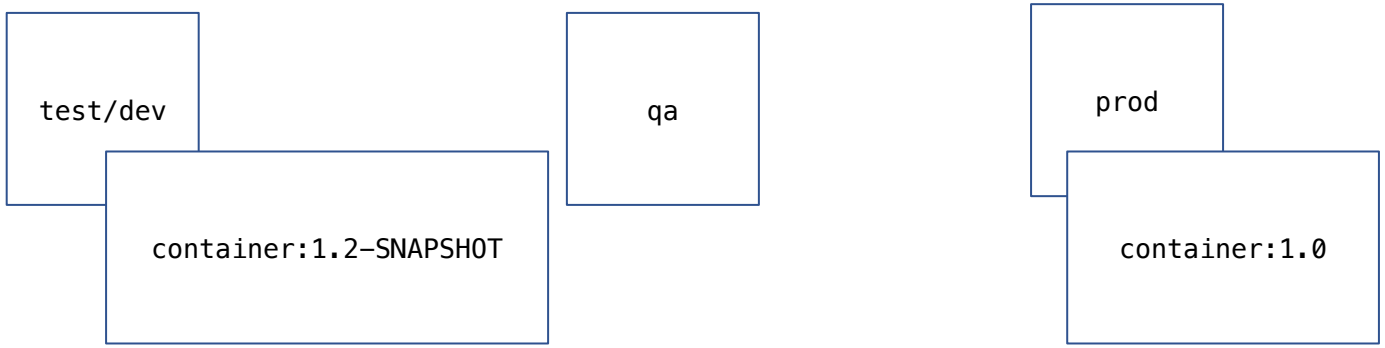
podman run - Publishing:

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

podman run -P / --publish-all

podman port -l
```





```
$ podman images
REPOSITORY                                      TAG         IMAGE ID       CREATED          SIZE
localhost/nginx                                 latest      e420c54187d7   14 seconds ago   260 MB
localhost/nginx                                 1           2fd45c021c45   9 minutes ago    260 MB

$ podman tag nginx:latest nginx:1

$ podman images
REPOSITORY                                      TAG         IMAGE ID       CREATED          SIZE
localhost/nginx                                 1           e420c54187d7   27 seconds ago   260 MB
localhost/nginx                                 latest      e420c54187d7   27 seconds ago   260 MB
<none>                                          <none>      2fd45c021c45   9 minutes ago    260 MB

$ podman image prune
2fd45c021c451352e18ed2383d967fd5d510d1551837446cc0f11202c7bbae05

$ podman images
REPOSITORY                                      TAG         IMAGE ID       CREATED              SIZE
localhost/nginx                                 latest      e420c54187d7   About a minute ago   260 MB
localhost/nginx                                 1           e420c54187d7   About a minute ago   260 MB
```



Container Registry REST-API

```
# curl -H "Authorization: Bearer $(oc whoami -t)" https://<openshift-registry>/v2/_catalog
{
  "repositories": [
    "images/nginx",
    ...
  ]
}

# curl -H "Authorization: Bearer $(oc whoami -t)" https://<openshift-registry>/v2/images/nginx/tags/list
{
  "name": "images/nginx",
  "tags": [
    "latest"
  ]
}

# curl -H "Auth ..." -H 'Accept: application/vnd.oci.image.manifest.v1+json' https://<openshift-registry>/v2/images/nginx/manifests/latest
{
  "schemaVersion": 2,
  "config": {
    "mediaType": "application/vnd.oci.image.config.v1+json",
    "digest": "sha256:54aece8b7356667b4e496c2451e818ced2d8490aa4833b681d604012968e4db2",
    "size": 468
  },
  "layers": [
    {
      "mediaType": "application/vnd.oci.image.layer.v1.tar+gzip",
      "digest": "sha256:9cda32363c5d3aa1f39a23ef855cc98cebae6cde9b60aab904497af252d3a053",
      "size": 68902376
    }
  ]
}
# curl -H 'Accept: application/vnd.oci.image.config.v1+json'
             https://<openshift-registry>/v2/images/nginx/blobs/sha256:54aece8b7356667b4e496c2451e818ced2d8490aa4833b681d604012968e4db2
...

# curl -H 'Accept: application/vnd.oci.image.layer.v1.tar+gzip'
             https://<openshift-registry>/v2/images/nginx/blobs/sha256:9cda32363c5d3aa1f39a23ef855cc98cebae6cde9b60aab904497af252d3a053
...
```

podman save – Image Operation

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

podman export – Container Operation

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

```
$ 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
...
```

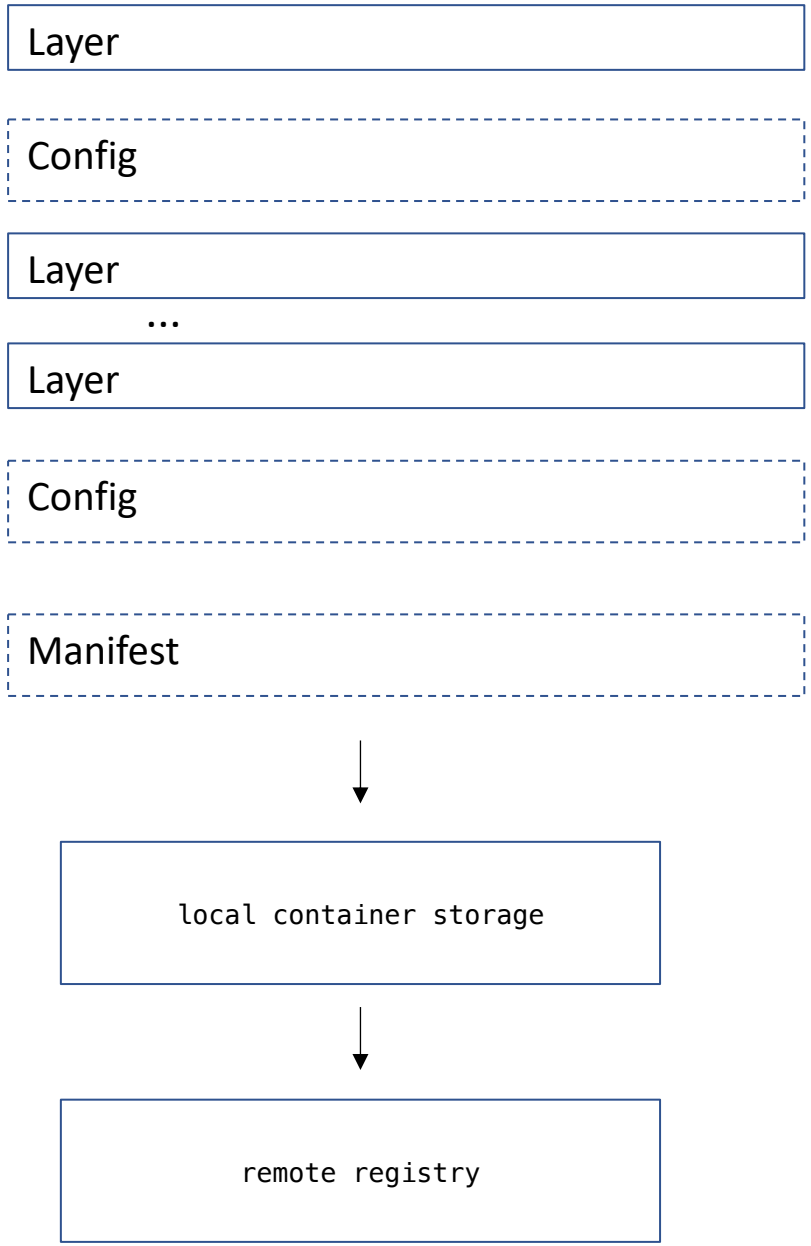
Container – Image

```
$ 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

```
FROM ubi8/ubi
LABEL version=.... maintainer=
MAINTAINER daniel
ENV key=value
ARG version=1

ADD  http://repos/app-$version.tar  /opt/app/
COPY webapp.war /opt/tomcat/webapps
RUN yum install –y tomcat  && \
        useradd tomcat && \
        chown tomcat:root /opt/tomcat &&
        yum cleanup && rm /tmp/* &&
        rm /var/cache/rpm

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

build image

Verwenden von YUM/DNF beim Image-Build

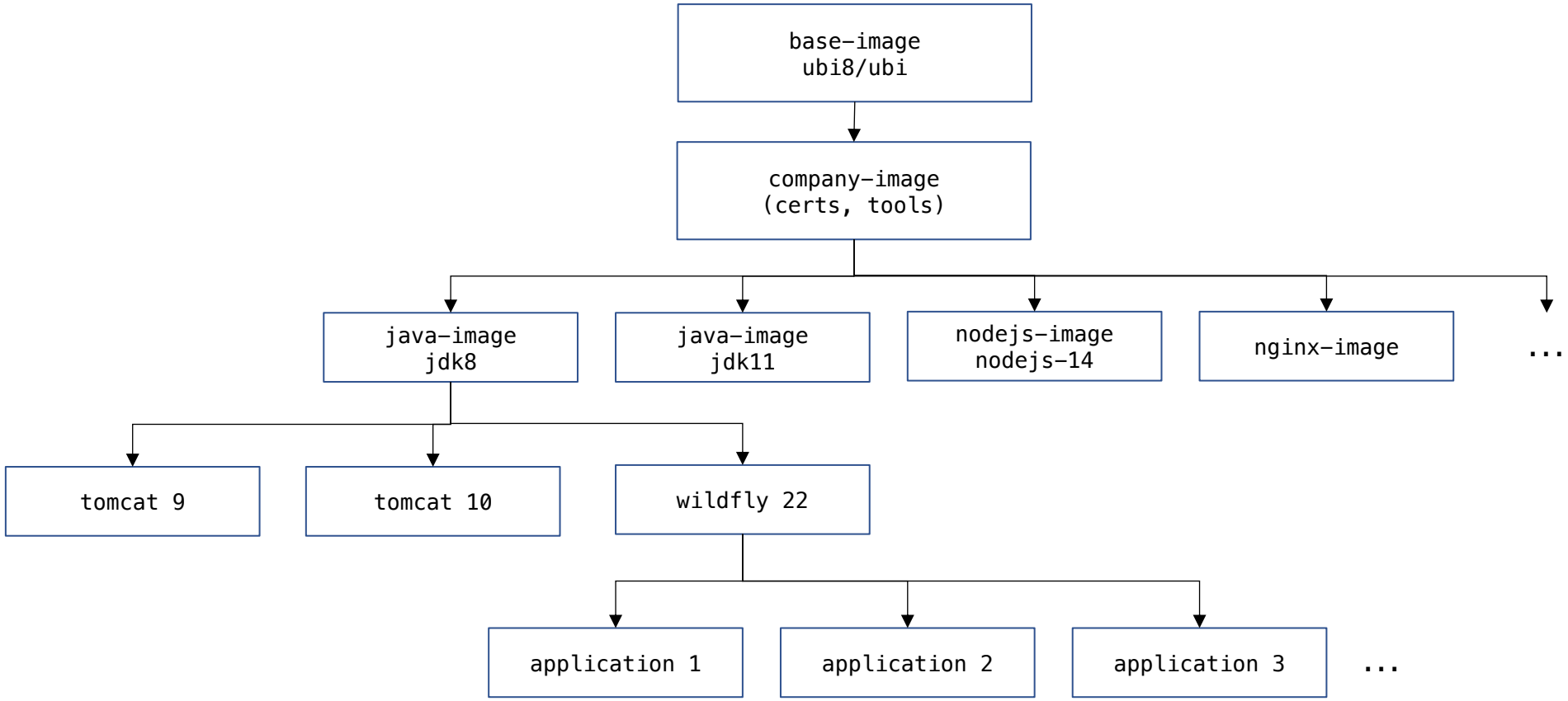
```
$ sudo podman run --rm ubi8/ubi cat /etc/yum.repos.d/ubi.repo
[ubi-8-baseos]
name = Red Hat Universal Base Image 8 (RPMs) - BaseOS
baseurl = https://cdn-ubi.redhat.com/content/public/ubi/dist/ubi8/8/$basearch/baseos/os
enabled = 1
gpgkey = file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
gpgcheck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