Container und Container Images

Was verbirgt sich dahinter?

Dan Čermák

Chemnitzer Linux-Tage
22nd and 23rd March 2025

who -u

Dan Čermák

- Software Developer @SUSE
- i3 SIG, Package maintainer
- Developer Tools, Testing and Documentation, Home Automation
- https://dancermak.name
- dcermak
- @Defolos@mastodon.social
- @defolos.bsky.social

Agenda

- Containers from Scratch
- Docker & Podman
- Container Orchestration
- Should I use containers?

Software Delivery

Software Delivery

- dependencies suck → bundle everything

Software Delivery

- dependencies suck → bundle everything
- Can't we just ship everything?

Software Delivery

- dependencies suck → bundle everything
- Can't we just ship everything?
- yes, but VMs suck as well 🫠

Let's build a container: `chroot`

Let's build a container: `chroot`

1. Create a chroot: `rsync -avz --exclude=/sysroot/ / /sysroot/`

Let's build a container: `chroot`

1. Create a chroot: `rsync -avz --exclude=/sysroot/ / /sysroot/`
2. build your app

Let's build a container: `chroot`

1. Create a chroot: `rsync -avz --exclude=/sysroot/ / /sysroot/`
2. build your app
3. clean `/sysroot` of everything unneeded

Let's build a container: `chroot`

1. Create a chroot: `rsync -avz --exclude=/sysroot/ / /sysroot/`
2. build your app
3. clean `/sysroot` of everything unneeded
4. `tar -czvf my-app.tar.gz /sysroot`

Let’s run the container

Let's run the container

1. `tar -xzvf my-app.tar.gz -C /path/to/app`

Let's run the container

1. `tar -xzvf my-app.tar.gz -C /path/to/app`
2. `chroot /path/to/app /usr/local/bin/my-app`

Is that it?

Is that it?

- inconvenient build process

Is that it?

- inconvenient build process
- no upgrade path

Is that it?

- inconvenient build process
- no upgrade path
- data sharing?

Is that it?

- inconvenient build process
- no upgrade path
- data sharing?
- no security

Linux Namespaces

Linux Namespaces

- kernel level resource isolation

Linux Namespaces

- kernel level resource isolation

  available namespaces:

Linux Namespaces

- kernel level resource isolation

available namespaces:

- user

Linux Namespaces

- kernel level resource isolation

  available namespaces:

  - user
  - mnt

Linux Namespaces

- kernel level resource isolation

  available namespaces:

  - user
  - mnt
  - pid

Linux Namespaces

- kernel level resource isolation

  available namespaces:

  - user
  - mnt
  - pid
  - net

Linux Namespaces

- kernel level resource isolation

  available namespaces:

  - user
  - mnt
  - pid
  - net
  - ipc

Linux Namespaces

- kernel level resource isolation

  available namespaces:

  - user
  - mnt
  - pid
  - net
  - ipc
  - uts

Linux Namespaces

- kernel level resource isolation

  available namespaces:

  - user
  - mnt
  - pid
  - net
  - ipc
  - uts
  - cgroup

Linux Namespaces

- kernel level resource isolation

  available namespaces:

  - user
  - mnt
  - pid
  - net
  - ipc
  - uts
  - cgroup
  - time

Try it:

Try it:

```
$ unshare --user --map-root-user \
      --pid --fork --mount-proc \
      /bin/bash
# whoami
root
# ps -a
    PID TTY          TIME CMD
      1 pts/8    00:00:00 bash
    104 pts/8    00:00:00 ps
```

Try it:

```
$ unshare --user --map-root-user \
      --pid --fork --mount-proc \
      /bin/bash
# whoami
root
# ps -a
    PID TTY          TIME CMD
      1 pts/8    00:00:00 bash
    104 pts/8    00:00:00 ps
```

Try it:

```
1 $ unshare --user --map-root-user \
2       --pid --fork --mount-proc \
3       /bin/bash
4 # whoami
5 root
6 # ps -a
7     PID TTY          TIME CMD
8       1 pts/8    00:00:00 bash
9     104 pts/8    00:00:00 ps
```

cgroups

cgroups

- apply resource limits to processes

cgroups

- apply resource limits to processes
- measure resource usage

cgroups

- apply resource limits to processes
- measure resource usage

```
1 # cgcreate -g memory:memlimit
2 # cgset -r memory.max=1K memlimit
3 # cgexec -g memory:memlimit ls -al
4 Killed
```

cgroups

- apply resource limits to processes
- measure resource usage

```
1 # cgcreate -g memory:memlimit
2 # cgset -r memory.max=1K memlimit
3 # cgexec -g memory:memlimit ls -al
4 Killed
```

cgroups

- apply resource limits to processes
- measure resource usage

```
# cgcreate -g memory:memlimit
# cgset -r memory.max=1K memlimit
# cgexec -g memory:memlimit ls -al
Killed
```

Are we there yet?

Are we there yet?

- 👍 great process isolation

Are we there yet?

- 👍 great process isolation
- 👎 standardized build process

Are we there yet?

- 👍 great process isolation
- 👎 standardized build process
- 👎 distribution mechanism

Introducing: Docker

Introducing: Docker

Introducing: Docker

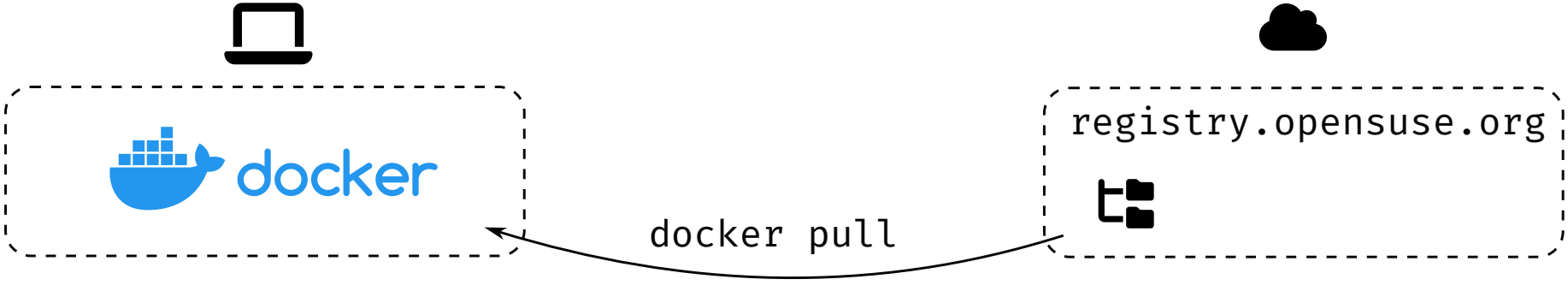
1. `docker build`

Introducing: Docker

1. `docker build`
2. Docker registry

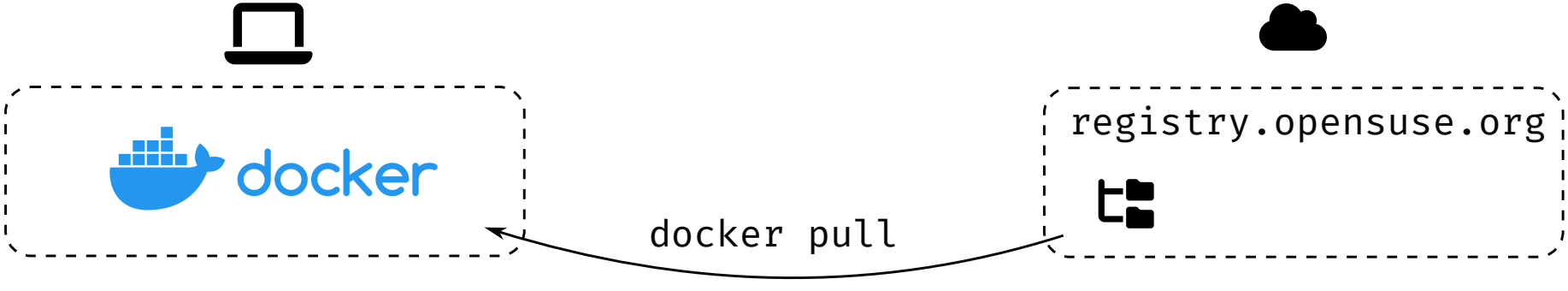
Docker Registry

Docker Registry



<registry>/<name>:<tag>@<digest>

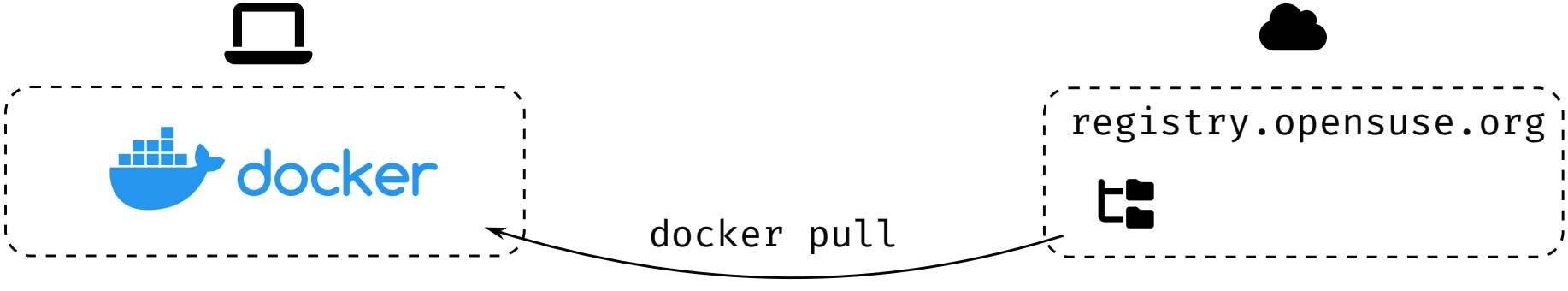
Docker Registry



<registry>/<name>:<tag>@<digest>

```
1 docker pull registry.opensuse.org/opensuse/leap
2 docker pull registry.opensuse.org/opensuse/leap:15.6
3 docker pull registry.opensuse.org/opensuse/leap:15.5@sha256:a5e
```

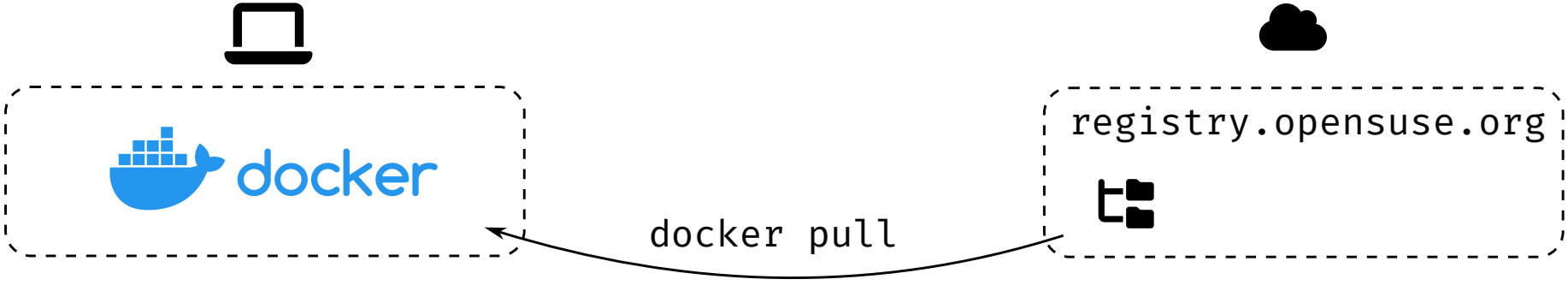
Docker Registry



<registry>/<name>:<tag>@<digest>

```
1 docker pull registry.opensuse.org/opensuse/leap
2 docker pull registry.opensuse.org/opensuse/leap:15.6
3 docker pull registry.opensuse.org/opensuse/leap:15.5@sha256:a5e
```

Docker Registry



<registry>/<name>:<tag>@<digest>

```
1 docker pull registry.opensuse.org/opensuse/leap
2 docker pull registry.opensuse.org/opensuse/leap:15.6
3 docker pull registry.opensuse.org/opensuse/leap:15.5@sha256:a5e
```

Container Image Build

Container Image Build

```
docker build .
```

Container Image Build

```
docker build .
```

```
FROM registry.opensuse.org/opensuse/tumbleweed

COPY . /src/
WORKDIR /src/

RUN zypper -n in python3-pip; \
    pip install . ; \
    zypper -n rm --clean-deps gcc; zypper -n clean; \
    rm -rf {/target,}/var/log/{alternatives.log,lastlog,tallyl

EXPOSE 80
CMD ["/usr/bin/python", "-m", "my-app"]
```

Container Image Build

```
docker build .
```

```
FROM registry.opensuse.org/opensuse/tumbleweed

COPY . /src/
WORKDIR /src/

RUN zypper -n in python3-pip; \
    pip install . ; \
    zypper -n rm --clean-deps gcc; zypper -n clean; \
    rm -rf {/target,}/var/log/{alternatives.log,lastlog,tallyl

EXPOSE 80
CMD ["/usr/bin/python", "-m", "my-app"]
```

Container Image Build

```
docker build .
```

```
FROM registry.opensuse.org/opensuse/tumbleweed

COPY . /src/
WORKDIR /src/

RUN zypper -n in python3-pip; \
    pip install . ; \
    zypper -n rm --clean-deps gcc; zypper -n clean; \
    rm -rf {/target,}/var/log/{alternatives.log,lastlog,tallyl

EXPOSE 80
CMD ["/usr/bin/python", "-m", "my-app"]
```

Container Image Build

```
docker build .
```

```
 1 FROM registry.opensuse.org/opensuse/tumbleweed
 2
 3 COPY . /src/
 4 WORKDIR /src/
 5
 6 RUN zypper -n in python3-pip; \
 7     pip install . ; \
 8     zypper -n rm --clean-deps gcc; zypper -n clean; \
 9     rm -rf {/target,}/var/log/{alternatives.log,lastlog,tallyl
10
11 EXPOSE 80
12 CMD ["/usr/bin/python", "-m", "my-app"]
```

Container Image Build

```
docker build .
```

```
FROM registry.opensuse.org/opensuse/tumbleweed

COPY . /src/
WORKDIR /src/

RUN zypper -n in python3-pip; \
    pip install . ; \
    zypper -n rm --clean-deps gcc; zypper -n clean; \
    rm -rf {/target,}/var/log/{alternatives.log,lastlog,tallyl

EXPOSE 80
CMD ["/usr/bin/python", "-m", "my-app"]
```

Container Image Build

```
docker build .
```

```
FROM registry.opensuse.org/opensuse/tumbleweed

COPY . /src/
WORKDIR /src/

RUN zypper -n in python3-pip; \
    pip install . ; \
    zypper -n rm --clean-deps gcc; zypper -n clean; \
    rm -rf {/target,}/var/log/{alternatives.log,lastlog,tallyl

EXPOSE 80
CMD ["/usr/bin/python", "-m", "my-app"]
```

Container Image Build

```
docker build .
```

```
FROM registry.opensuse.org/opensuse/tumbleweed

COPY . /src/
WORKDIR /src/

RUN zypper -n in python3-pip; \
    pip install . ; \
    zypper -n rm --clean-deps gcc; zypper -n clean; \
    rm -rf {/target,}/var/log/{alternatives.log,lastlog,tallyl

EXPOSE 80
CMD ["/usr/bin/python", "-m", "my-app"]
```

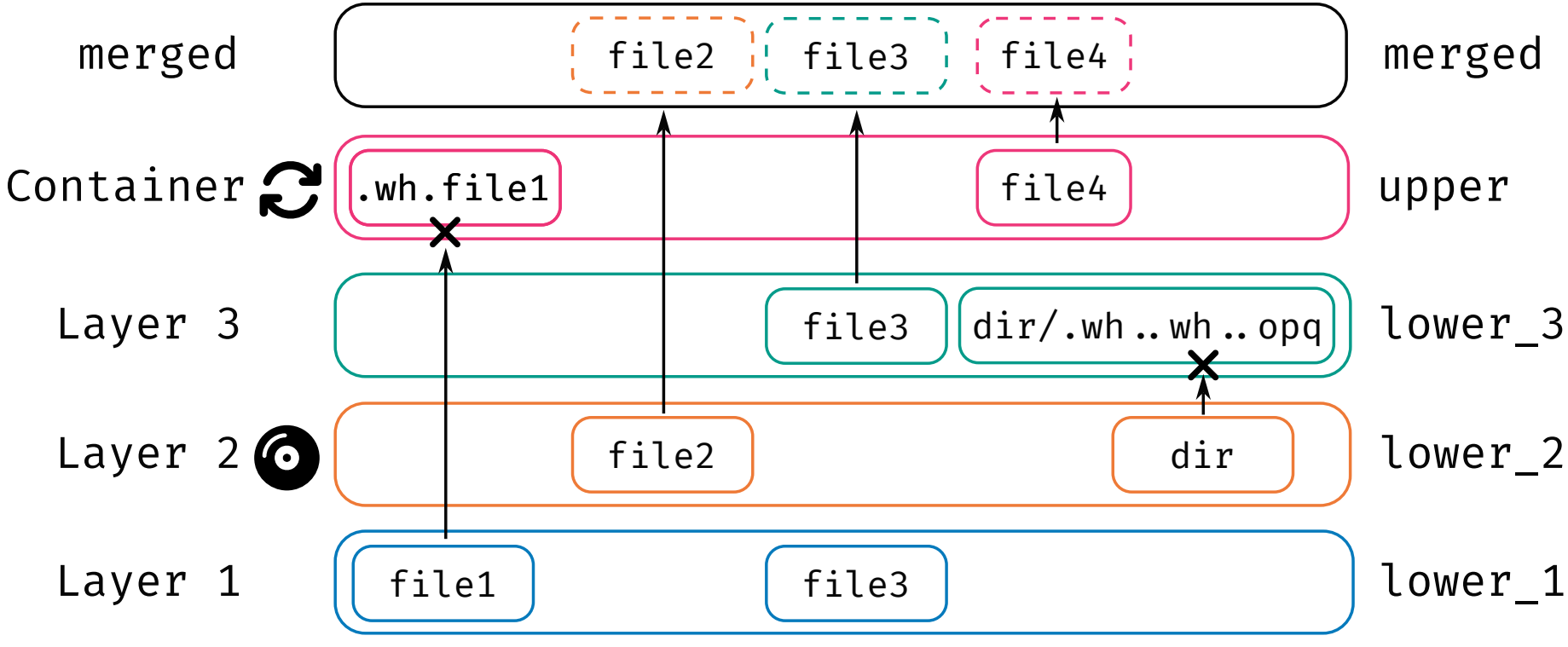
Container Image Build

```
docker build .
```

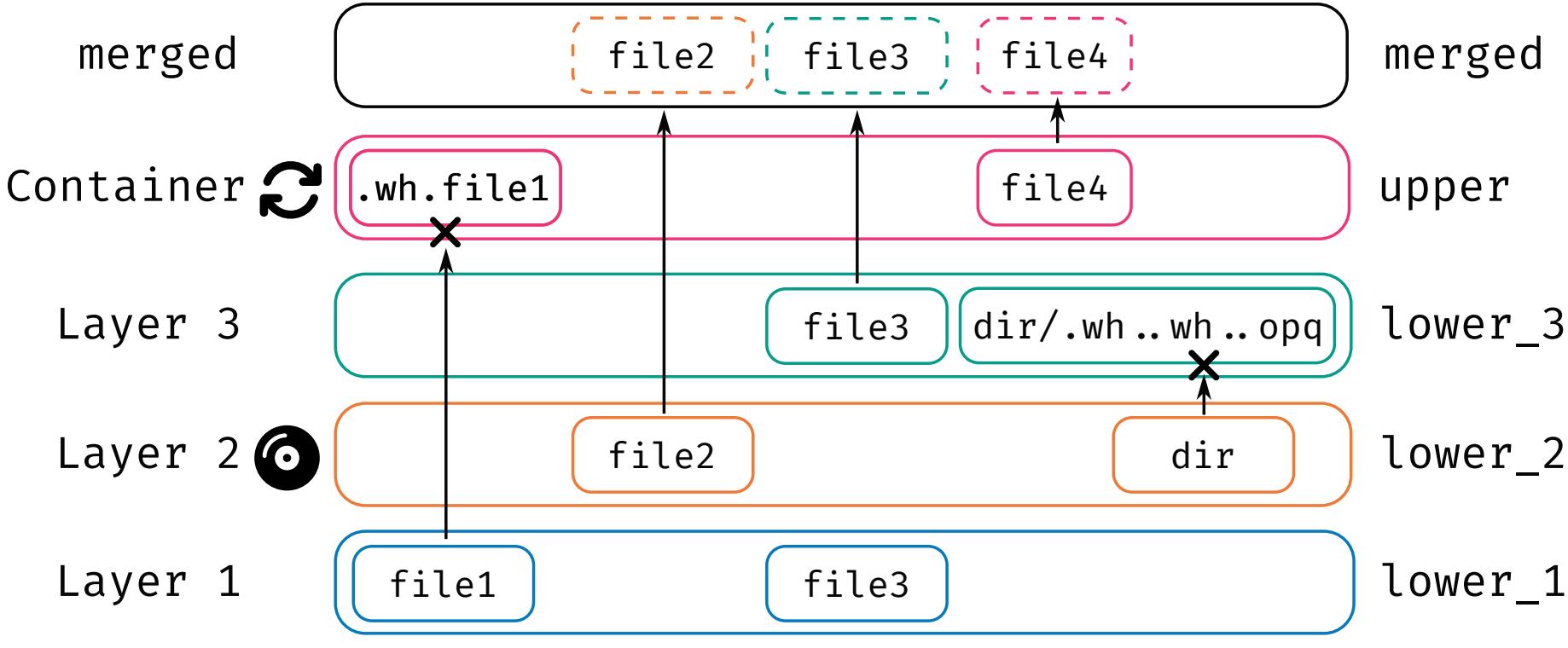
```
FROM registry.opensuse.org/opensuse/tumbleweed

COPY . /src/
WORKDIR /src/

RUN zypper -n in python3-pip; \
    pip install . ; \
    zypper -n rm --clean-deps gcc; zypper -n clean; \
    rm -rf {/target,}/var/log/{alternatives.log,lastlog,tallyl

EXPOSE 80
CMD ["/usr/bin/python", "-m", "my-app"]
```

UnionFS

UnionFS


merged
file2
file3
file4
merged
Container
.wh.file1
file4
upper
Layer 3
file3
dir/.wh..wh..opq
lower_3
Layer 2
file2
dir
lower_2
Layer 1
file1
file3
lower_1

UnionFS




```
mount -t overlay overlay \
      -o lowerdir=lower_3:lower_2:lower_1,\
         upperdir=upper,workdir=/work/ \
           merged
```

Dockerfile

Dockerfile

```
 1 FROM registry.opensuse.org/opensuse/tumbleweed
 2 COPY ./project/ /src/
 3 ENV USER="geeko"
 4 RUN zypper -n in openssh-clients; \
 5     ssh-keygen -t ed25519 -f /root/.ssh/id_ed25519 -N ""; \
 6     zypper -n rm --clean-deps openssh-clients; \
 7     zypper -n clean; rm -rf /var/log/lastlog;
 8 VOLUME ["/src/data"]
 9 WORKDIR /src/
10 EXPOSE 22
11 RUN useradd $USER
12 USER $USER
13 CMD ["echo hello"]
14 ENTRYPOINT ["/bin/bash", "-ce"]
```

Dockerfile

```
 1 FROM registry.opensuse.org/opensuse/tumbleweed
 2 COPY ./project/ /src/
 3 ENV USER="geeko"
 4 RUN zypper -n in openssh-clients; \
 5     ssh-keygen -t ed25519 -f /root/.ssh/id_ed25519 -N ""; \
 6     zypper -n rm --clean-deps openssh-clients; \
 7     zypper -n clean; rm -rf /var/log/lastlog;
 8 VOLUME ["/src/data"]
 9 WORKDIR /src/
10 EXPOSE 22
11 RUN useradd $USER
12 USER $USER
13 CMD ["echo hello"]
14 ENTRYPOINT ["/bin/bash", "-ce"]
```

Dockerfile

```
 1 FROM registry.opensuse.org/opensuse/tumbleweed
 2 COPY ./project/ /src/
 3 ENV USER="geeko"
 4 RUN zypper -n in openssh-clients; \
 5     ssh-keygen -t ed25519 -f /root/.ssh/id_ed25519 -N ""; \
 6     zypper -n rm --clean-deps openssh-clients; \
 7     zypper -n clean; rm -rf /var/log/lastlog;
 8 VOLUME ["/src/data"]
 9 WORKDIR /src/
10 EXPOSE 22
11 RUN useradd $USER
12 USER $USER
13 CMD ["echo hello"]
14 ENTRYPOINT ["/bin/bash", "-ce"]
```

Dockerfile

```
 1 FROM registry.opensuse.org/opensuse/tumbleweed
 2 COPY ./project/ /src/
 3 ENV USER="geeko"
 4 RUN zypper -n in openssh-clients; \
 5     ssh-keygen -t ed25519 -f /root/.ssh/id_ed25519 -N ""; \
 6     zypper -n rm --clean-deps openssh-clients; \
 7     zypper -n clean; rm -rf /var/log/lastlog;
 8 VOLUME ["/src/data"]
 9 WORKDIR /src/
10 EXPOSE 22
11 RUN useradd $USER
12 USER $USER
13 CMD ["echo hello"]
14 ENTRYPOINT ["/bin/bash", "-ce"]
```

Dockerfile

```
 1 FROM registry.opensuse.org/opensuse/tumbleweed
 2 COPY ./project/ /src/
 3 ENV USER="geeko"
 4 RUN zypper -n in openssh-clients; \
 5     ssh-keygen -t ed25519 -f /root/.ssh/id_ed25519 -N ""; \
 6     zypper -n rm --clean-deps openssh-clients; \
 7     zypper -n clean; rm -rf /var/log/lastlog;
 8 VOLUME ["/src/data"]
 9 WORKDIR /src/
10 EXPOSE 22
11 RUN useradd $USER
12 USER $USER
13 CMD ["echo hello"]
14 ENTRYPOINT ["/bin/bash", "-ce"]
```

Dockerfile

```
 1 FROM registry.opensuse.org/opensuse/tumbleweed
 2 COPY ./project/ /src/
 3 ENV USER="geeko"
 4 RUN zypper -n in openssh-clients; \
 5     ssh-keygen -t ed25519 -f /root/.ssh/id_ed25519 -N ""; \
 6     zypper -n rm --clean-deps openssh-clients; \
 7     zypper -n clean; rm -rf /var/log/lastlog;
 8 VOLUME ["/src/data"]
 9 WORKDIR /src/
10 EXPOSE 22
11 RUN useradd $USER
12 USER $USER
13 CMD ["echo hello"]
14 ENTRYPOINT ["/bin/bash", "-ce"]
```

Dockerfile

```
 1 FROM registry.opensuse.org/opensuse/tumbleweed
 2 COPY ./project/ /src/
 3 ENV USER="geeko"
 4 RUN zypper -n in openssh-clients; \
 5     ssh-keygen -t ed25519 -f /root/.ssh/id_ed25519 -N ""; \
 6     zypper -n rm --clean-deps openssh-clients; \
 7     zypper -n clean; rm -rf /var/log/lastlog;
 8 VOLUME ["/src/data"]
 9 WORKDIR /src/
10 EXPOSE 22
11 RUN useradd $USER
12 USER $USER
13 CMD ["echo hello"]
14 ENTRYPOINT ["/bin/bash", "-ce"]
```

Dockerfile

```
 1 FROM registry.opensuse.org/opensuse/tumbleweed
 2 COPY ./project/ /src/
 3 ENV USER="geeko"
 4 RUN zypper -n in openssh-clients; \
 5     ssh-keygen -t ed25519 -f /root/.ssh/id_ed25519 -N ""; \
 6     zypper -n rm --clean-deps openssh-clients; \
 7     zypper -n clean; rm -rf /var/log/lastlog;
 8 VOLUME ["/src/data"]
 9 WORKDIR /src/
10 EXPOSE 22
11 RUN useradd $USER
12 USER $USER
13 CMD ["echo hello"]
14 ENTRYPOINT ["/bin/bash", "-ce"]
```

Dockerfile

```
 1 FROM registry.opensuse.org/opensuse/tumbleweed
 2 COPY ./project/ /src/
 3 ENV USER="geeko"
 4 RUN zypper -n in openssh-clients; \
 5     ssh-keygen -t ed25519 -f /root/.ssh/id_ed25519 -N ""; \
 6     zypper -n rm --clean-deps openssh-clients; \
 7     zypper -n clean; rm -rf /var/log/lastlog;
 8 VOLUME ["/src/data"]
 9 WORKDIR /src/
10 EXPOSE 22
11 RUN useradd $USER
12 USER $USER
13 CMD ["echo hello"]
14 ENTRYPOINT ["/bin/bash", "-ce"]
```

Dockerfile

```
 1 FROM registry.opensuse.org/opensuse/tumbleweed
 2 COPY ./project/ /src/
 3 ENV USER="geeko"
 4 RUN zypper -n in openssh-clients; \
 5     ssh-keygen -t ed25519 -f /root/.ssh/id_ed25519 -N ""; \
 6     zypper -n rm --clean-deps openssh-clients; \
 7     zypper -n clean; rm -rf /var/log/lastlog;
 8 VOLUME ["/src/data"]
 9 WORKDIR /src/
10 EXPOSE 22
11 RUN useradd $USER
12 USER $USER
13 CMD ["echo hello"]
14 ENTRYPOINT ["/bin/bash", "-ce"]
```

Dockerfile

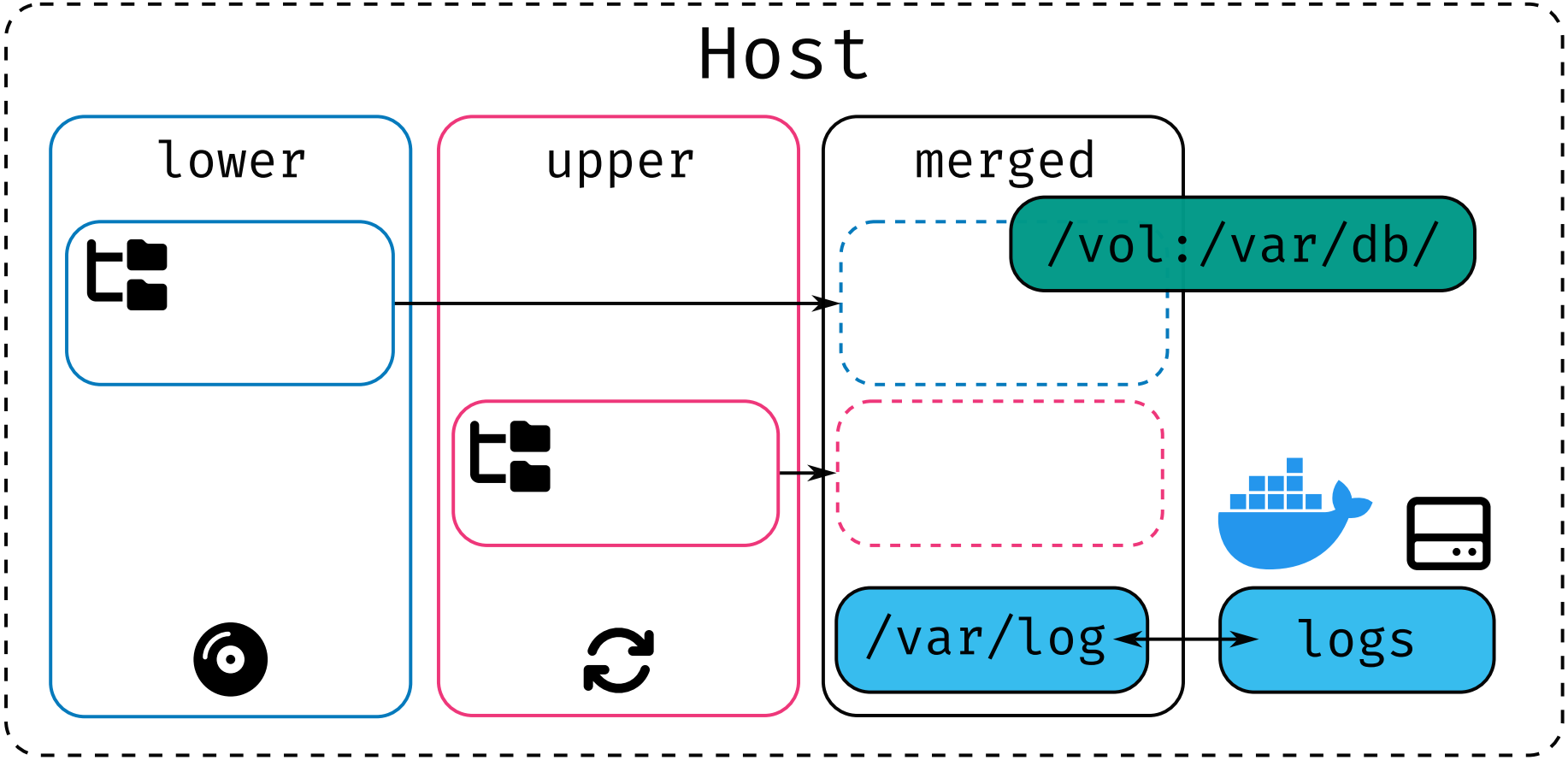
```
 1 FROM registry.opensuse.org/opensuse/tumbleweed
 2 COPY ./project/ /src/
 3 ENV USER="geeko"
 4 RUN zypper -n in openssh-clients; \
 5     ssh-keygen -t ed25519 -f /root/.ssh/id_ed25519 -N ""; \
 6     zypper -n rm --clean-deps openssh-clients; \
 7     zypper -n clean; rm -rf /var/log/lastlog;
 8 VOLUME ["/src/data"]
 9 WORKDIR /src/
10 EXPOSE 22
11 RUN useradd $USER
12 USER $USER
13 CMD ["echo hello"]
14 ENTRYPOINT ["/bin/bash", "-ce"]
```

Dockerfile

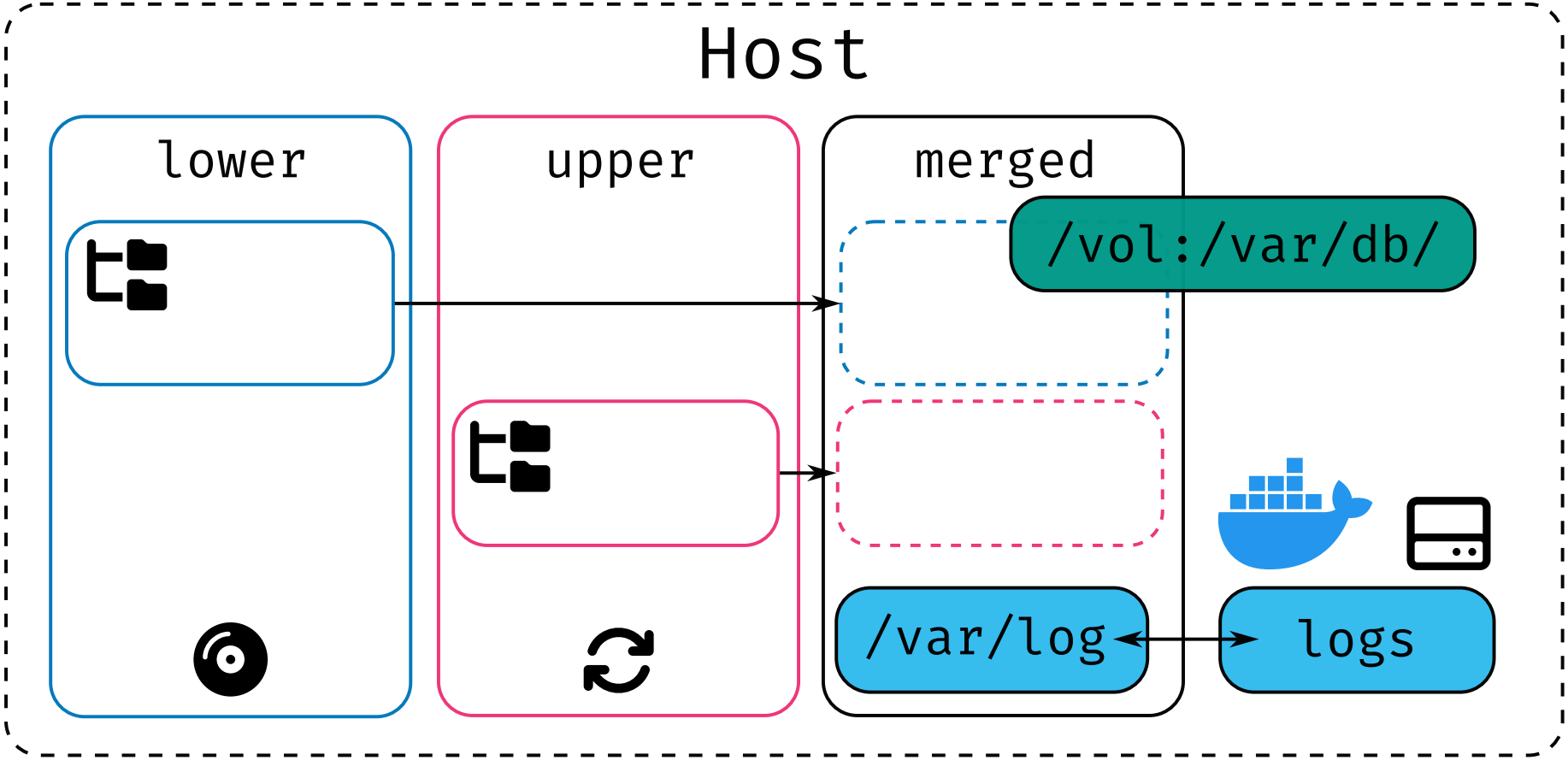
```
 1 FROM registry.opensuse.org/opensuse/tumbleweed
 2 COPY ./project/ /src/
 3 ENV USER="geeko"
 4 RUN zypper -n in openssh-clients; \
 5     ssh-keygen -t ed25519 -f /root/.ssh/id_ed25519 -N ""; \
 6     zypper -n rm --clean-deps openssh-clients; \
 7     zypper -n clean; rm -rf /var/log/lastlog;
 8 VOLUME ["/src/data"]
 9 WORKDIR /src/
10 EXPOSE 22
11 RUN useradd $USER
12 USER $USER
13 CMD ["echo hello"]
14 ENTRYPOINT ["/bin/bash", "-ce"]
```

Volumes

Volumes



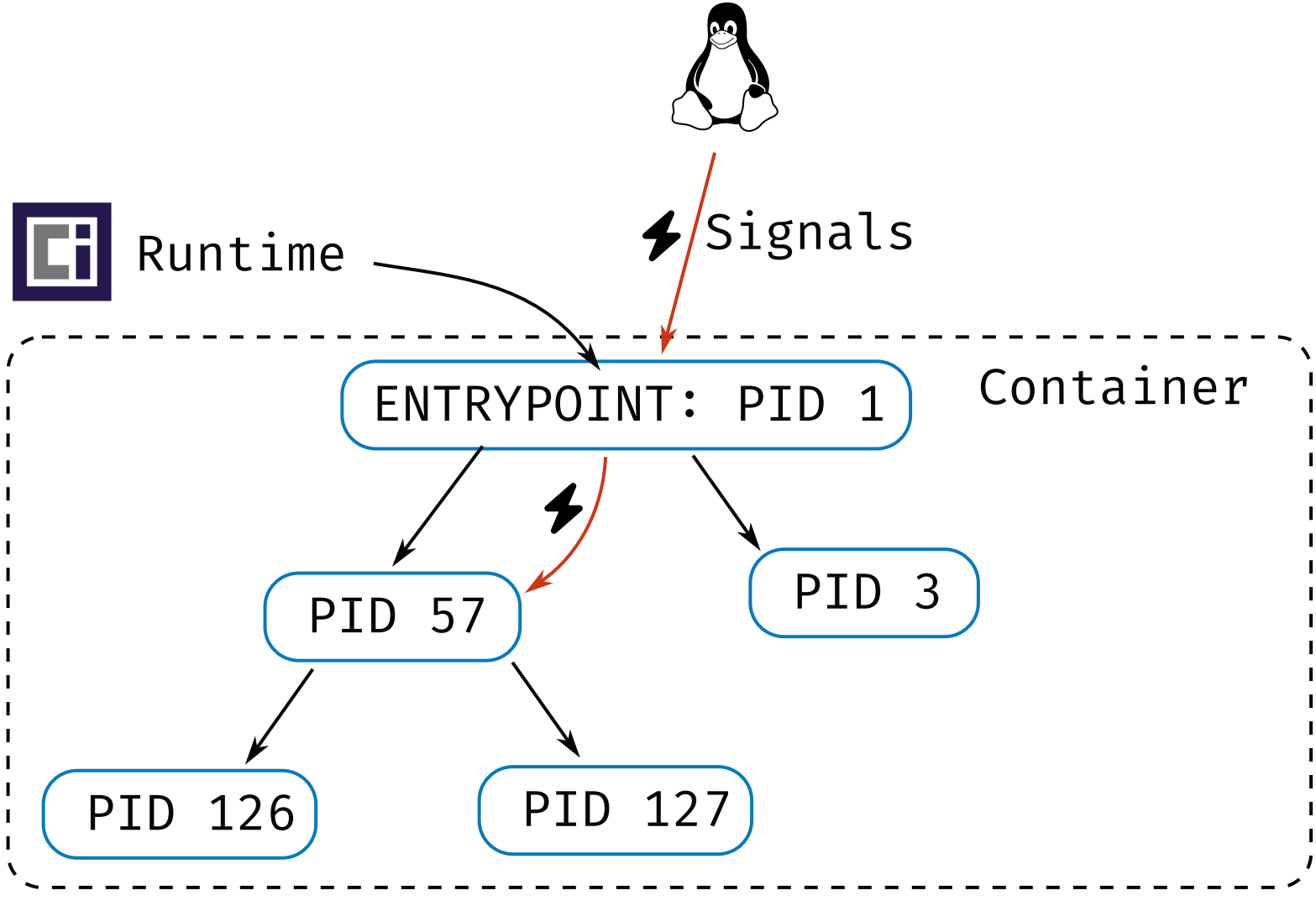
Volumes




```
docker run -v /vol/:/var/db/ -v logs:/var/log $img
```

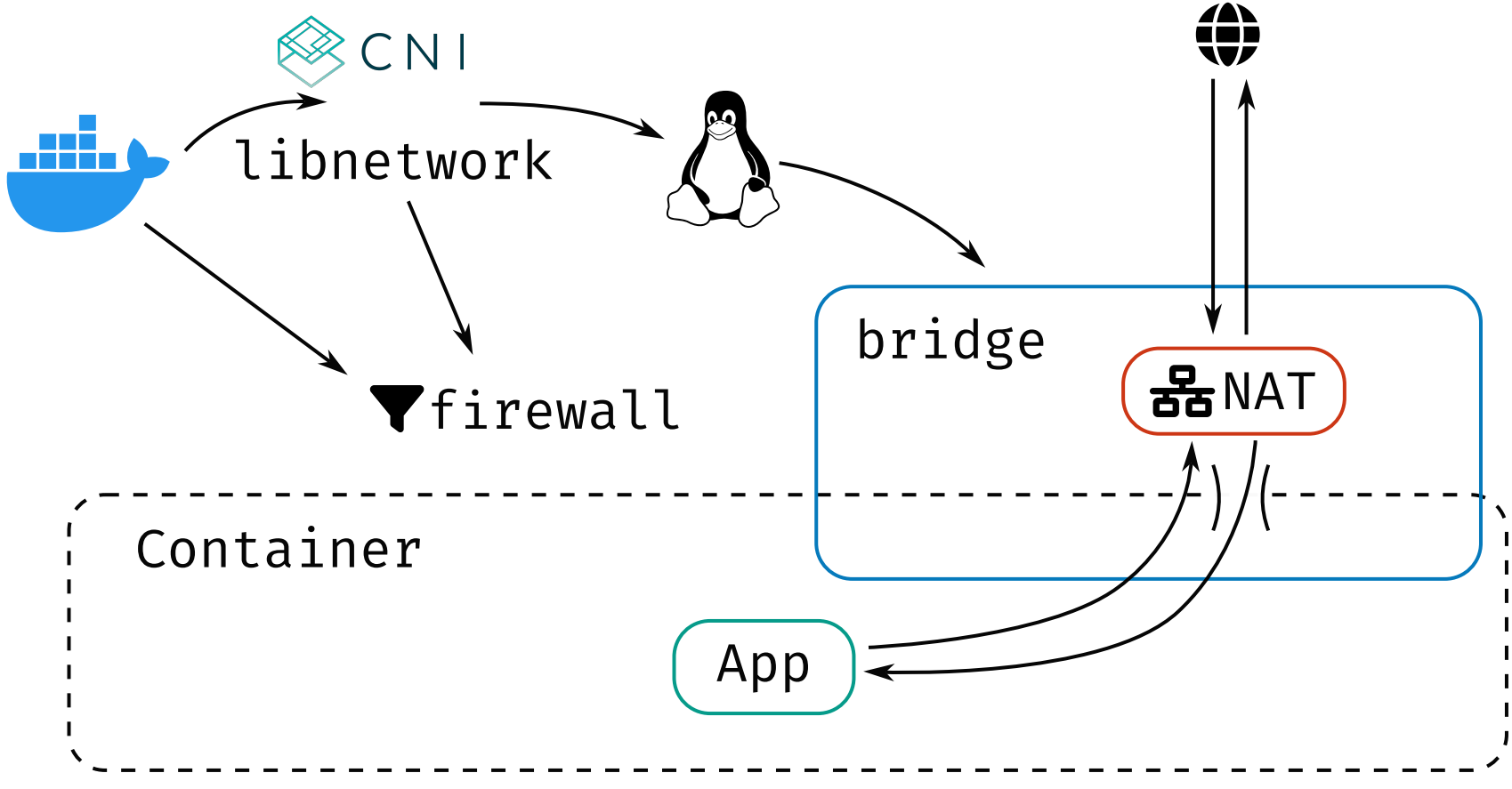
Entrypoint

Entrypoint


Runtime
Signals
ENTRYPOINT: PID 1
Container
PID 57
PID 3
PID 126
PID 127

Networking

Networking



Best Practices

Best Practices

```
RUN zypper -n in python3-pip; \
    pip install . ; \
    zypper -n rm --clean-deps gcc; zypper -n clean; \
    rm -rf {/target,}/var/log/{alternatives.log,lastlog,tallylog,
```

```
$ podman run -e POSTGRES_PORT=1234 \
             -e POSTGRES_USER=pg \
                 my-app
$ podman run my-app bash
#
```

```
$ podman run -e POSTGRES_PORT=1234 \
             -e POSTGRES_USER=pg \
                 my-app
$ podman run my-app bash
#
```

or:

```
$ podman run -e POSTGRES_PORT=1234 \
             -e POSTGRES_USER=pg \
                 my-app
$ podman run my-app bash
#
```

or:

```
$ podman run my-app
#
```

Volumes are your friend:

Volumes are your friend:

```
VOLUME ["/var/db/"]
# /var/db/ is now erased after each step!
```

Volumes are your friend:

```
VOLUME ["/var/db/"]
# /var/db/ is now erased after each step!
```

use the exec-form:

Volumes are your friend:

```
VOLUME ["/var/db/"]
# /var/db/ is now erased after each step!
```

use the exec-form:

```
ENTRYPOINT ["/usr/bin/my-app", "-param", "value"]
```

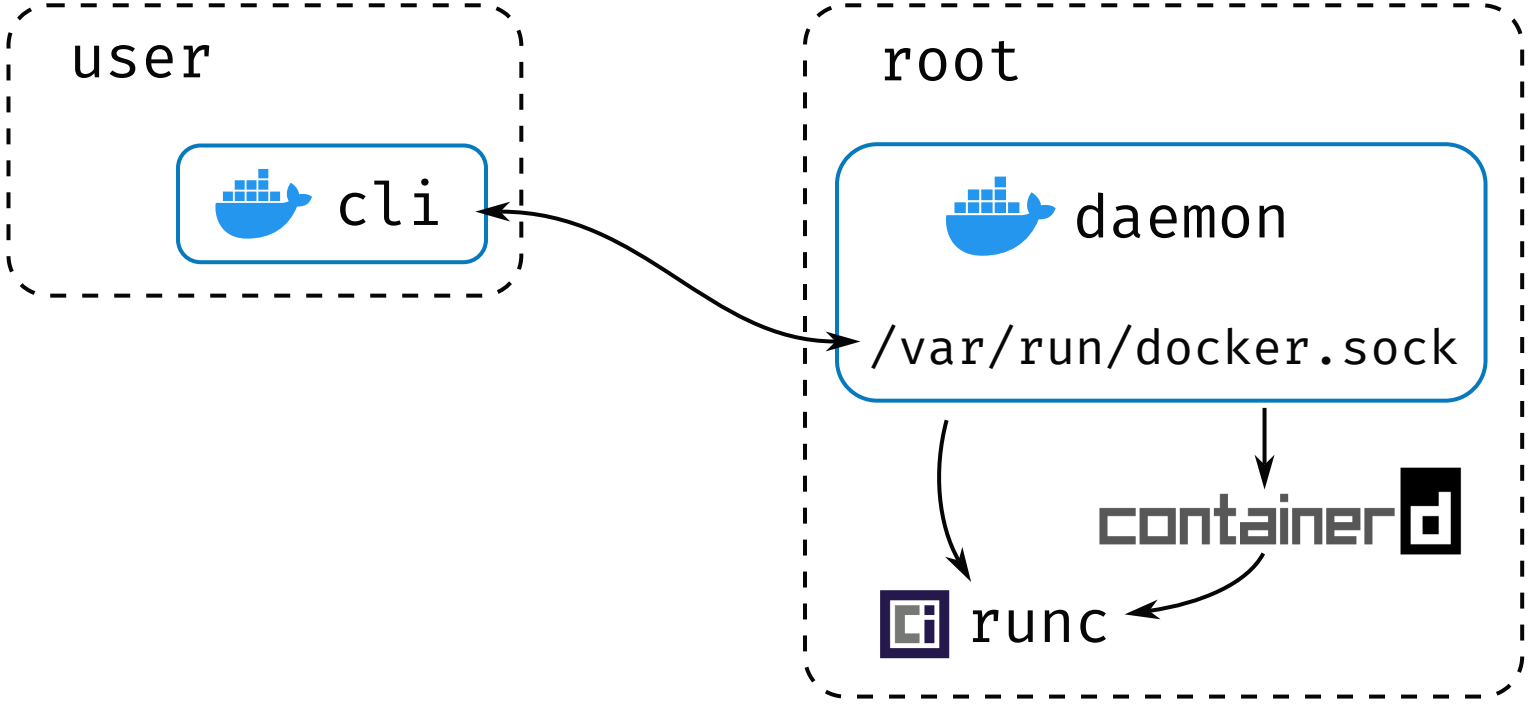
Podman

Podman

Actually Docker

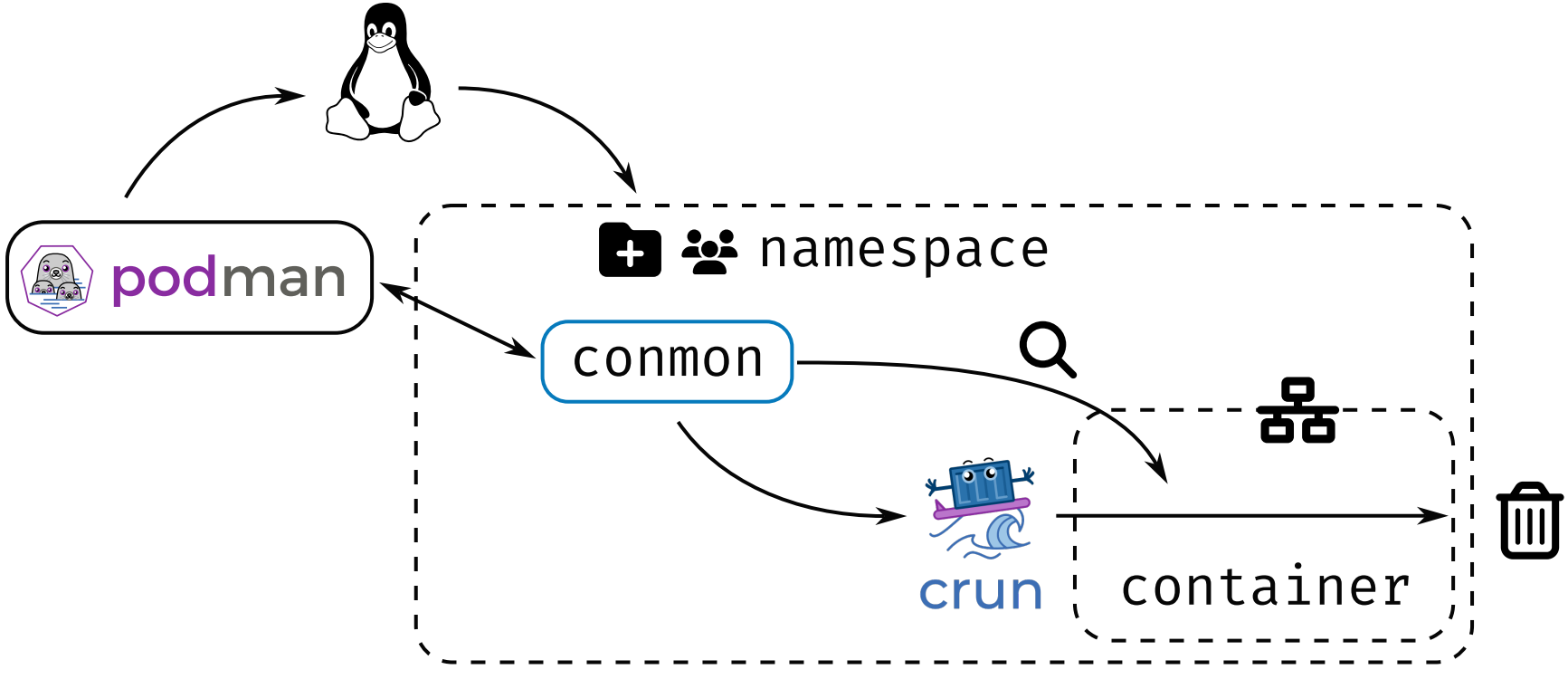
Podman

Actually Docker



Podman

Podman



Rootless Containers

Rootless Containers

- container runs as non-root or a sub-uid of your user

Rootless Containers

- container runs as non-root or a sub-uid of your user
- rootless networking runs in userspace

Security

Security

- container potentially as privileged as the user running it

Security

- container potentially as privileged as the user running it
- container breakout attacks exist

Security

- container potentially as privileged as the user running it
- container breakout attacks exist
- SELinux is your friend

When to use

When to use

- Single binary

When to use

- Single binary
- Cloud Native Deployment

When to use

- Single binary
- Cloud Native Deployment
- Testing other Distributions

When to use

- Single binary
- Cloud Native Deployment
- Testing other Distributions
- Reproducible Dev/Test/Build Environment

When to use

- Single binary
- Cloud Native Deployment
- Testing other Distributions
- Reproducible Dev/Test/Build Environment
- Base OS doesn't matter

When not to use

When not to use

- Complex multi binary legacy code

When not to use

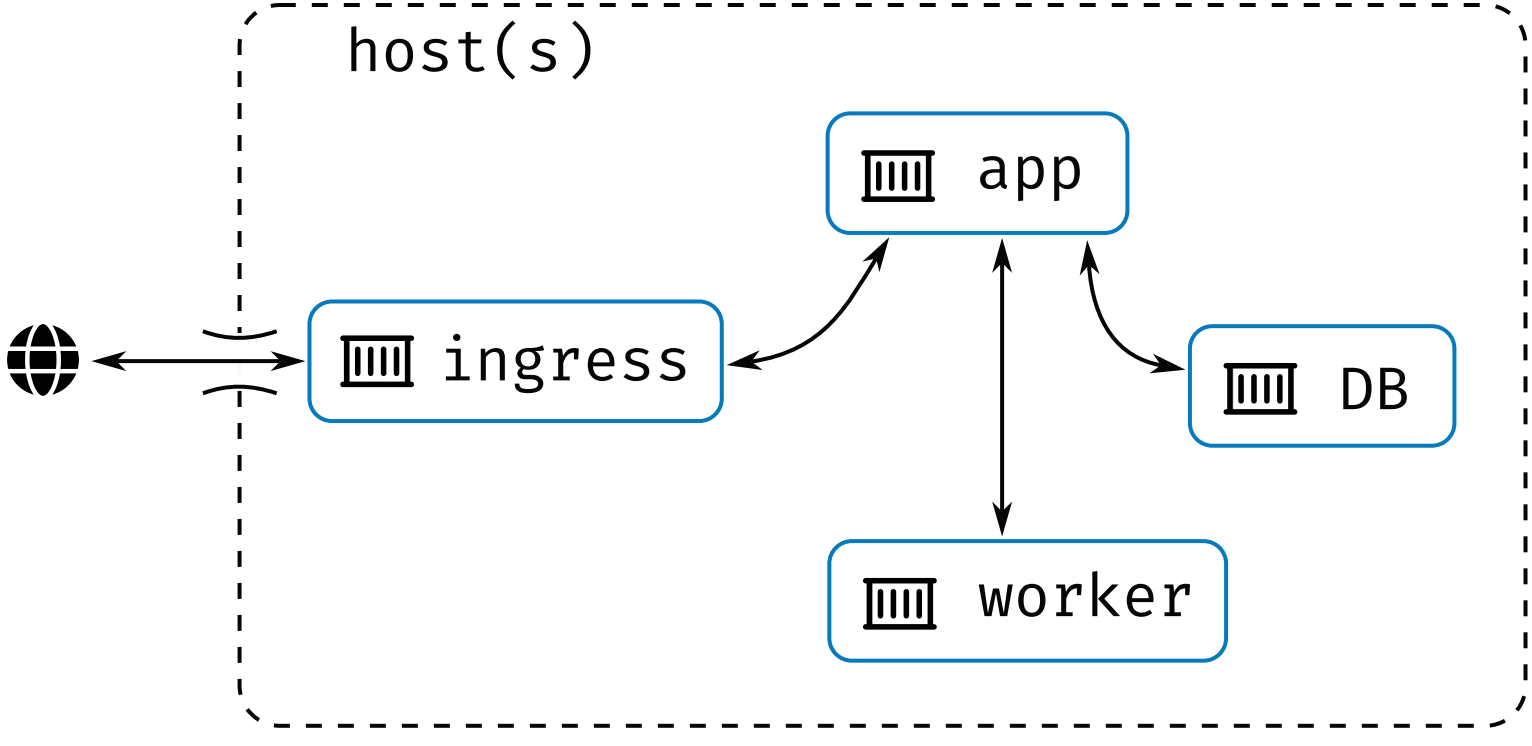
- Complex multi binary legacy code
- High-Performance Code

When not to use

- Complex multi binary legacy code
- High-Performance Code
- Base OS matters

Container Orchestration

Container Orchestration



docker-compose

docker-compose

```
services:
  app:
    build: .
    ports:
      - "8080:8080"
    volumes:
      - .:/src
    depends_on:
      db:
        condition: service_healthy
  db:
    image: registry.opensuse.org/opensuse/mariadb
    environment:
      - MARIADB_ALLOW_EMPTY_ROOT_PASSWORD=1
```

docker-compose

```
services:
  app:
    build: .
    ports:
      - "8080:8080"
    volumes:
      - .:/src
    depends_on:
      db:
        condition: service_healthy
  db:
    image: registry.opensuse.org/opensuse/mariadb
    environment:
      - MARIADB_ALLOW_EMPTY_ROOT_PASSWORD=1
```

```
docker compose up
```

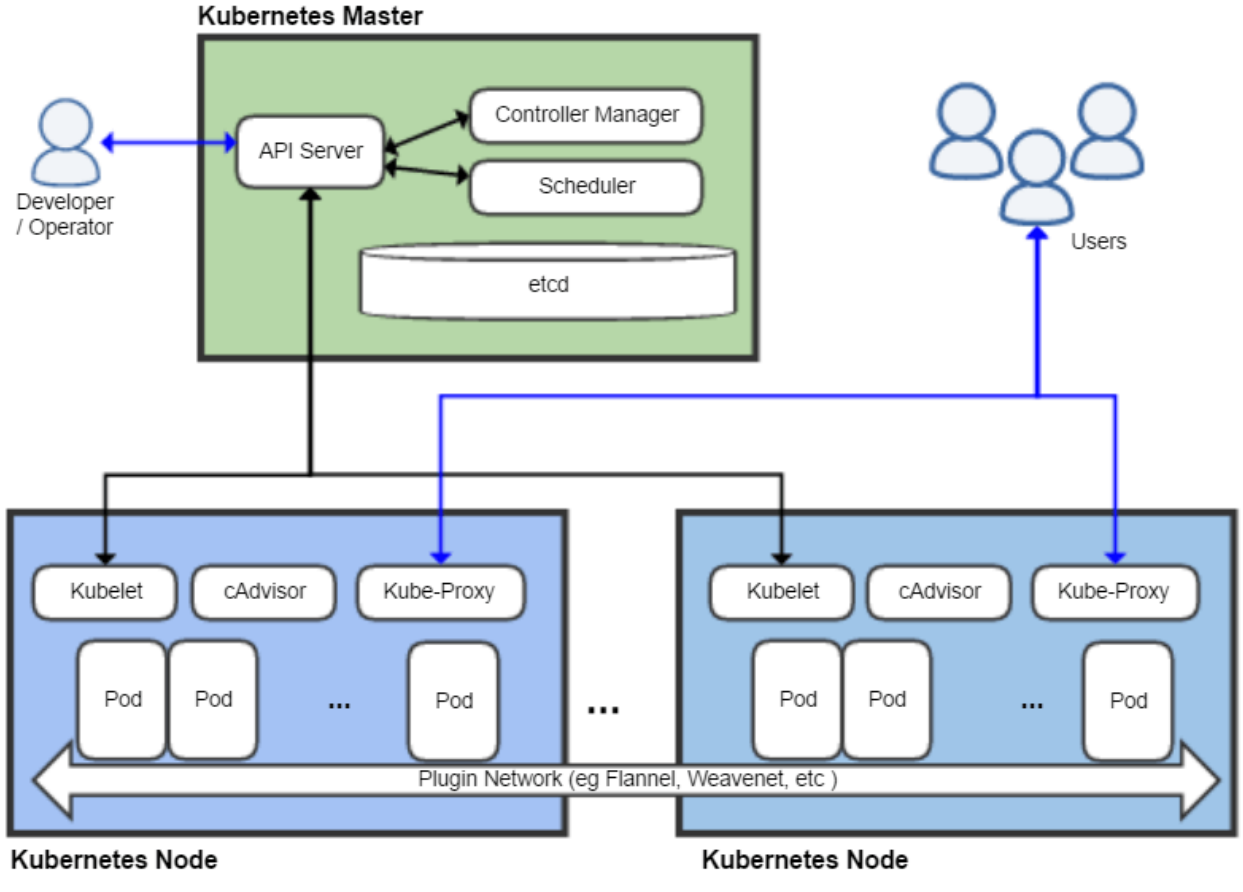
Quadlet / `podman generate systemd`

Quadlet / `podman generate systemd`

```
[Unit]
Description=TW container

[Container]
Image=registry.opensuse.org/opensuse/tumbleweed

# volume and network defined below in other configs
Volume=test.volume:/data
Network=test.network

Exec=sleep infinity

[Service]
Restart=always
TimeoutStartSec=900
```

Kubernetes

Kubernetes


Kubernetes Master
Developer
/ Operator
API Server
Controller Manager
Scheduler
etcd
Users
Kubelet
cAdvisor
Kube-Proxy
Pod
Pod
...
Pod
...
Kubelet
cAdvisor
Kube-Proxy
Pod
Pod
...
Pod
Plugin Network (eg Flannel, Weavenet, etc)
Kubernetes Node
Kubernetes Node

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-application
  labels:
    app: web
spec:
  replicas: 3
  selector:
    matchLabels:
      app: web
  template:
    metadata:
      labels:
        app: web
```

Should I use containers?

Should I use containers?

It depends

Questions?

dcermak.github.io/container-images