# Complex Application in Container

Jan Pazdziora
Sr. Principal Software Engineer
Identity Management Special Projects, Red Hat


5th October 2015

# Linux containers

Different containers can use different subsets of:

- Namespaces
  - Mount (filesystems hierarchy) (`docker run -v ...`)
  - Network (devices, IP addresses, routing) (`docker run --net=...`)
  - Process IDs (`docker run --pid=...`)
  - User and group IDs (currently not used by Docker)
  - UTS (hostname, domainname) (`docker run`'s `--uts` and `-h`)
  - IPC (SysV IPC, message queues) (`docker run --ipc=...`)
- Control groups (cgroups) — setting limits
- SELinux (use `--selinux-enabled` with Docker daemon)
- iptables (use `--icc=false` with Docker daemon)

# Building image

- Dockerfile:

```
FROM fedora
RUN dnf -y install httpd && dnf clean all
RUN dnf -y install /usr/bin/ps /usr/sbin/ip && dnf clean all
RUN echo "Test Server" > /var/www/html/index.html
CMD [ "/usr/sbin/httpd", "-DFOREGROUND" ]
```

- Build image:

```
host$ docker build -t httpd .
Sending build context to Docker daemon 2.048 kB
Step 0 : FROM fedora
[...]
Successfully built 911aa3a3937c
```

```
host$ docker images
REPOSITORY        TAG     IMAGE ID      CREATED       VIRTUAL SIZE
httpd             latest  911aa3a3937c  2 minutes ago 203.7 MB
docker.io/fedora  latest  ded7cd95e059  4 months ago  186.5 MB
```

# Running container

- Start (run) new container from this image:

```
host$ docker run --name httpd httpd &
[1] 1556
host$ AH00558: httpd: Could not reliably determine the server's fully qu
```

```
host$ docker ps
CONTAINER ID IMAGE COMMAND                  CREATED         STATUS
9312c1c40a92 httpd "/usr/sbin/httpd -DFO" 41 seconds ago Up 40 seconds
```

```
host$ docker inspect -f '{{ .NetworkSettings.IPAddress }}' httpd
172.17.0.4
host$ curl http://172.17.0.4/
Test Server
```

- Just the one process (plus the worker children) is run.

# Namespaces

- PID namespace:

```
host$ docker exec httpd ps ax
  PID TTY      STAT   TIME COMMAND
    1 ?        Ss     0:00 /usr/sbin/httpd -DFOREGROUND
    8 ?        S      0:00 /usr/sbin/httpd -DFOREGROUND
    9 ?        S      0:00 /usr/sbin/httpd -DFOREGROUND
   10 ?        S      0:00 /usr/sbin/httpd -DFOREGROUND
   11 ?        S      0:00 /usr/sbin/httpd -DFOREGROUND
   12 ?        Rs     0:00 ps ax
```

- Network namespace:

```
host$ docker exec httpd ip route
default via 172.17.42.1 dev eth0
172.17.0.0/16 dev eth0  proto kernel  scope link  src 172.17.0.4
```

- View namespace transitions on the host:

```
host# pstree -S | grep docker
        |-docker(mnt)-+-httpd(ipc,mnt,net,pid,uts)---4*[httpd]
        |              `-12*[{docker}]
```

# Filesystems and volumes

- The image is mounted as root:

```
host$ docker exec httpd findmnt -n /
/ /dev/mapper/docker-253:0-396043-bb73e8f8e3c3e06d7480d4d0f7dd890ad
8e7eda186b541cb79a70ba2eba3cc8a[/rootfs] ext4   rw,relatime,context
="system _u:object_r:svirt_sandbox_file_t:s0:c224,c877",stripe=16,d
ata=ordered
```

- We get data / configuration to the container by bind-mounting volumes:

```
host$ mkdir /tmp/data
host$ echo "Test serving data from volume" > /tmp/data/index.html
host$ docker run --name httpd -v /tmp/data:/var/www/html:Z httpd &
host$ docker inspect --format '{{ .HostConfig.Binds }}' httpd
[/tmp/data:/var/www/html:Z]
host$ ls -aZ /tmp/data | cut -d ' ' -f 1,4,5
drwxr-xr-x. system_u:object_r:svirt_sandbox_file_t:s0:c206,c497 .
drwxrwxrwt. system_u:object_r:tmp_t:s0       ..
-rw-r--r--. system_u:object_r:svirt_sandbox_file_t:s0:c206,c497 index.ht

host$ curl http://172.17.0.8/
Test serving data from volume
```

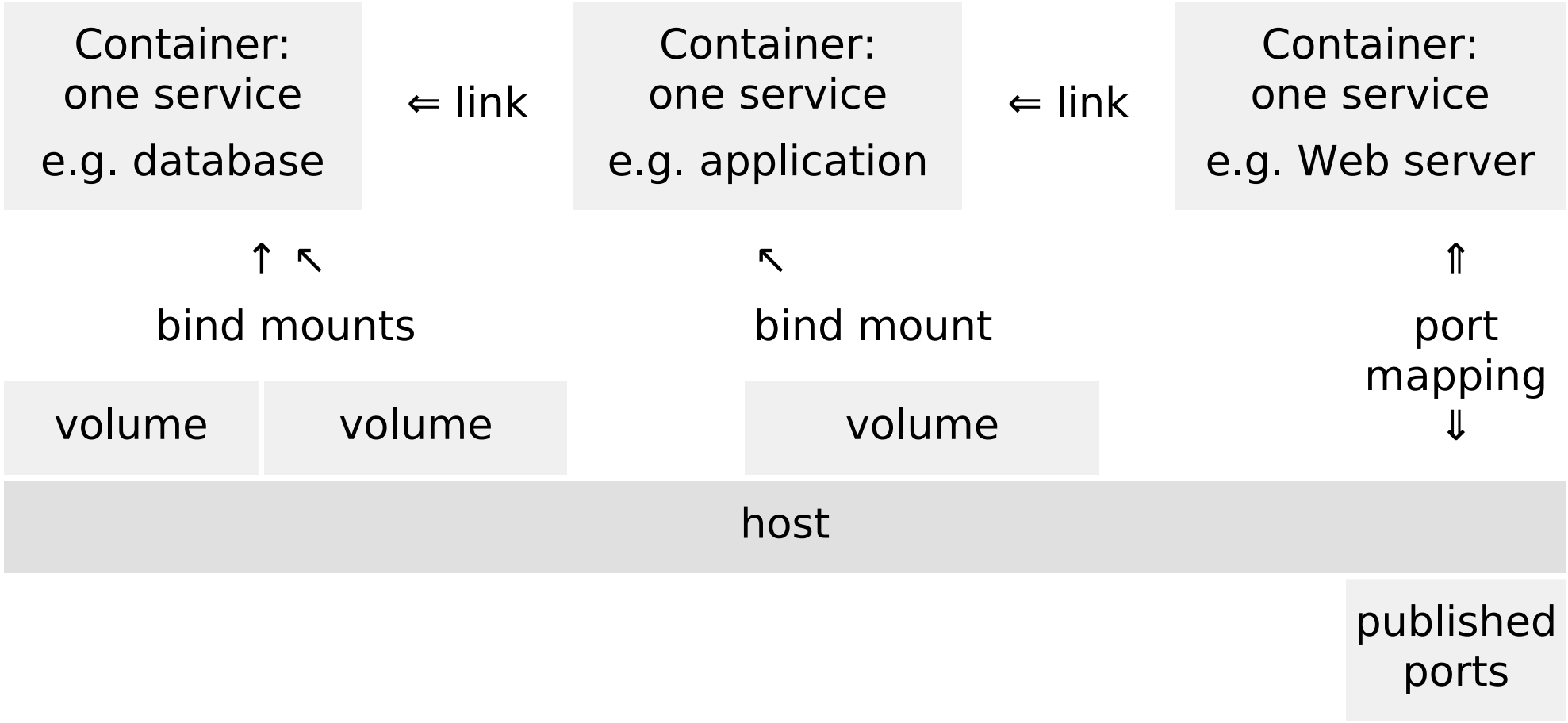
# Getting application to container

Usual advice:

- One service / daemon per container.
- Start the daemon / process directly.
- Connect containers to form desired setup (`docker run --link ...`).
- Bind-mount volumes with configs / data to containers.
- Configuration in build-time.

# Containerized setup

Usual setup:



# Multi-container setup

Concerns with multi-container setups:

- Starting the individual components (container) manually is error-prone.
  - There may be dozen of them in the setup.
- Components may expect Unix sockets and not support TCP.
  - The external layer has to provide those shared socket locations.
- There is wealth of knowledge in
  - SysV init scripts;
  - systemd unit files;
  - config / setup tools that assume everything on one machine.
- Extensive initial setup might be needed, in run-time.

# Complex application

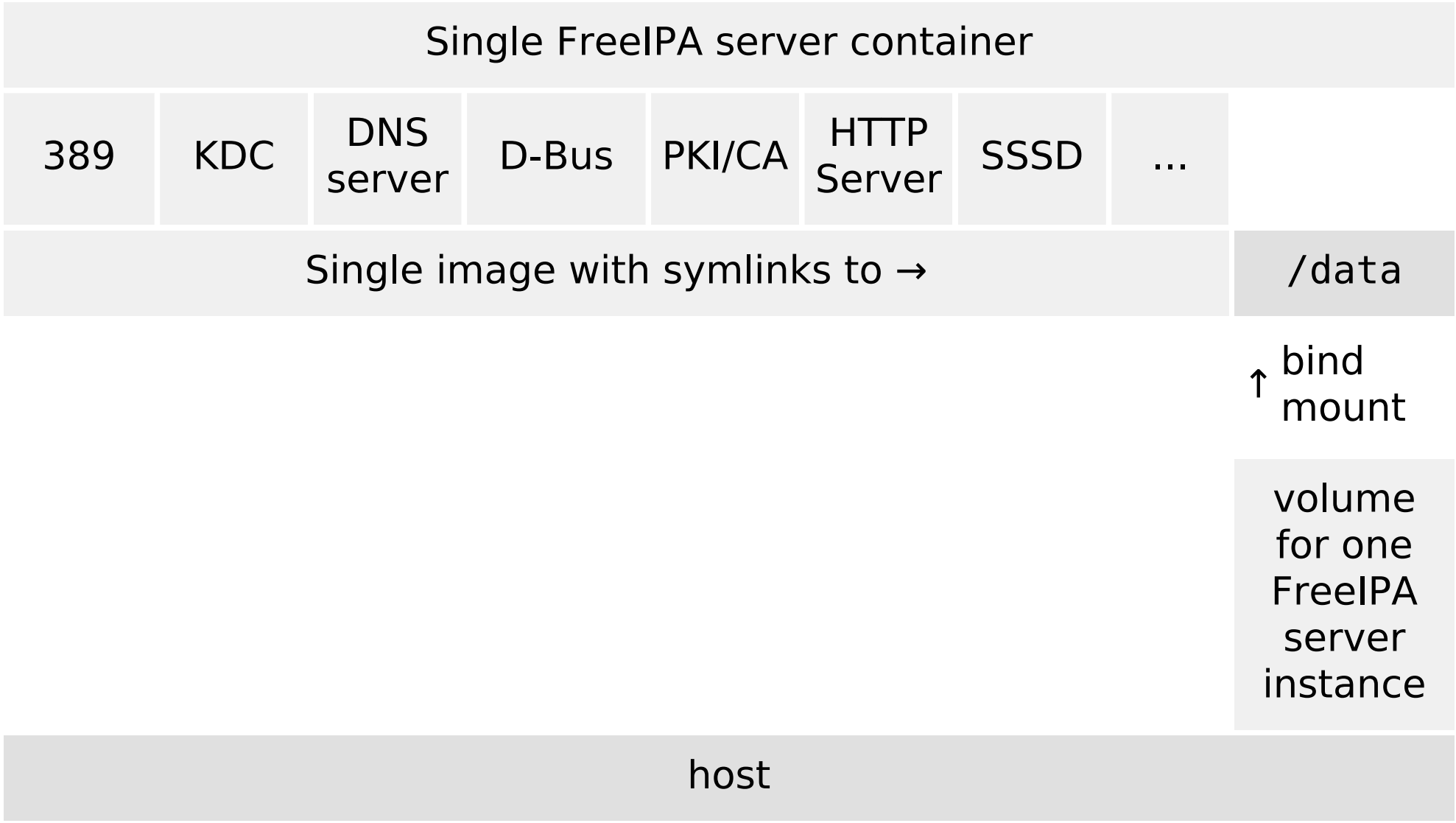
FreeIPA server:

- Dozen of daemons.
  - They form an integrated solution.
  - They expect to run on a single machine.
  - Their libraries share config files (nss, krb5, ...).
- The `ipa-server-install` or `ipa-replica-install` setup tools play important role.
  - They currently do not support multi-host setups.
- The domain / realm for which FreeIPA is reflected in LDAP schemas or directory and file names.
  - Only known in run-time.

# Approach

- All parts in one container.
- Minimize number of volumes to one: `/data`.
- In build time:
  - Install software (`yum install -y freeipa-server`).
  - Move directories and files that will be populate fresh volume aside, to `/data-template`.
  - On image, create symlinks pointing to `/data-template`.
- During first run, when empty volume is detected:
  - Populate the volume, mounted to `/data`, with the template content.
  - Run `ipa-server-install` (the setup tool).
- During subsequent runs, just start the services.

# Layout


Single FreeIPA server container
389
KDC
DNS server
D-Bus
PKI/CA
HTTP Server
SSSD
...
Single image with symlinks to →
/data
↑ bind mount
volume for one FreeIPA server instance
host

# Initial instance configuration

```
  PID TTY STAT TIME COMMAND
    1 ?   Ss   0:00 /bin/bash /usr/sbin/ipa-server-configure-first
   43 ?   S    0:00 xargs /usr/sbin/ipa-server-install -U
   44 ?   S    0:01  \_ /usr/bin/python2 -E /usr/sbin/ipa-server-install -U --d
   74 ?   S    0:00      \_ /usr/bin/perl /usr/sbin/setup-ds.pl --silent --logf
   89 ?   S    0:00          \_ sh -c /var/lib/dirsrv/scripts-EXAMPLE-COM/ldif2
   90 ?   S    0:00              \_ /bin/sh /var/lib/dirsrv/scripts-EXAMPLE-COM
   91 ?   S    0:00                  \_ /bin/sh ./ldif2db -n userRoot -i /var/l
  119 ?   Sl   0:00                      \_ /usr/sbin/ns-slapd ldif2db -D /etc/
   66 ?   Ss   0:00 /usr/sbin/ntpd -u ntp:ntp -g -x
```

# FreeIPA container running

```
 PID TTY STAT TIME COMMAND
   1 ?   Ss   0:00 /bin/bash /usr/sbin/ipa-server-configure-first
1470 ?   Ss   0:00 /bin/dbus-daemon --system --fork
1479 ?   Ss   0:00 /usr/sbin/certmonger -S -p /var/run/certmonger.pid -n
2010 ?   Ss   0:00 /usr/sbin/kadmind -P /var/run/kadmind.pid
2020 ?   Ssl  0:00 /usr/bin/memcached -d -s /var/run/ipa_memcached/ipa_memcach
2043 ?   Ss   0:00 /usr/bin/perl /bin/systemctl-socket-daemon /var/run/krb5kdc
2225 ?   Sl   0:01 /usr/sbin/ns-slapd -D /etc/dirsrv/slapd-EXAMPLE-COM -i /var
2274 ?   Ss   0:00 /usr/sbin/krb5kdc -P /var/run/krb5kdc.pid
2502 ?   Ss   0:00 sh -c export TOMCAT_CFG_LOADED="1"; export TOMCATS_BASE="/v
2503 ?   S    0:00  \_ /usr/sbin/runuser -g pkiuser -u pkiuser -- /usr/libexec
2504 ?   Sl   0:11      \_ /usr/lib/jvm/jre/bin/java -DRESTEASY_LIB=/usr/share
2635 ?   Ssl  0:00 /usr/sbin/named-pkcs11 -u named
2645 ?   Ss   0:00 sh -c export LANG=C; /usr/sbin/httpd $OPTIONS -DFOREGROUND
2646 ?   S    0:00  \_ /usr/sbin/httpd -DFOREGROUND
2647 ?   S    0:00      \_ /usr/libexec/nss_pcache 458756 off /etc/httpd/alias
2648 ?   Sl   0:01      \_ /usr/sbin/httpd -DFOREGROUND
2649 ?   Sl   0:01      \_ /usr/sbin/httpd -DFOREGROUND
2650 ?   S    0:00      \_ /usr/sbin/httpd -DFOREGROUND
2651 ?   S    0:00      \_ /usr/sbin/httpd -DFOREGROUND
2652 ?   S    0:00      \_ /usr/sbin/httpd -DFOREGROUND
```

# FreeIPA container running (cont'd)

```
2653 ?   S    0:00      \_ /usr/sbin/httpd -DFOREGROUND
2654 ?   S    0:00      \_ /usr/sbin/httpd -DFOREGROUND
2685 ?   S    0:00      \_ /usr/sbin/httpd -DFOREGROUND
2733 ?   Ss   0:00 /usr/sbin/sssd -D -f
2738 ?   S    0:00  \_ /usr/libexec/sssd/sssd_be --domain example.com --uid 0
2740 ?   S    0:00  \_ /usr/libexec/sssd/sssd_nss --uid 0 --gid 0 --debug-to-f
2741 ?   S    0:00  \_ /usr/libexec/sssd/sssd_sudo --uid 0 --gid 0 --debug-to-
2742 ?   S    0:00  \_ /usr/libexec/sssd/sssd_pam --uid 0 --gid 0 --debug-to-f
2743 ?   S    0:00  \_ /usr/libexec/sssd/sssd_pac --uid 0 --gid 0 --debug-to-f
```

# Implementation

- Iterative process.
  - Used `docker diff` to check changes confined to volume.
- FreeIPA's setup tools use `systemctl` heavily.
  - No `systemd` in the container.
  - `systemctl` replacement written, tailored to FreeIPA's services.
- Multiple services on multiple ports:

```
EXPOSE 53/udp 53 80 443 389 636 88 464 88/udp 464/udp 123/udp 7389 9443
```

  - Exposed by mapping to host's ports (`docker run -p ...`).

# Security implications

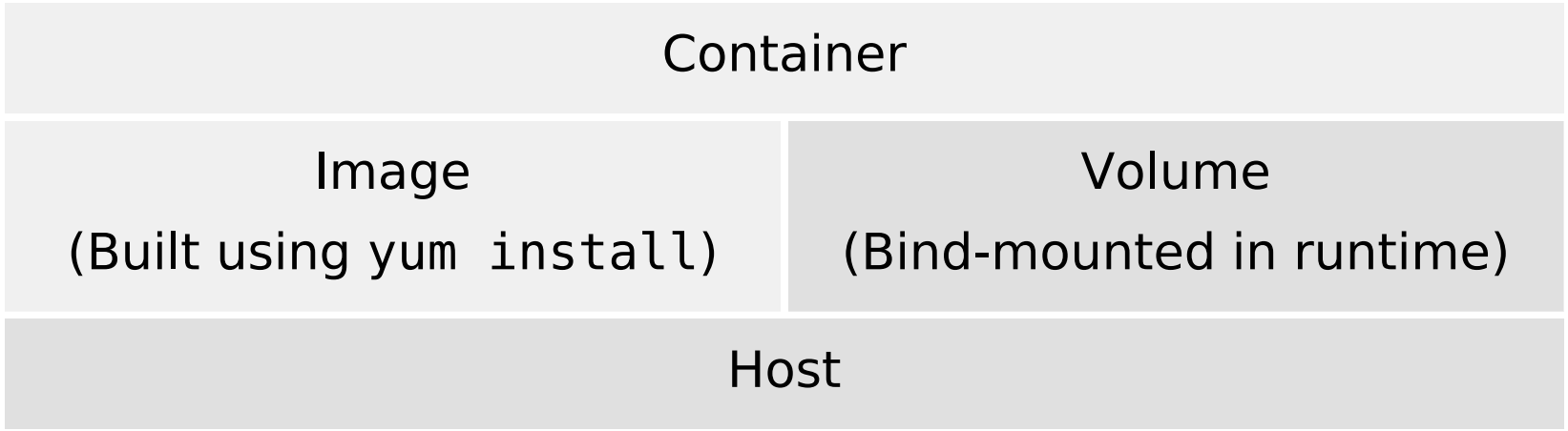
- All processes in the container share one SELinux domain.
  - Host's SELinux policy is used.
  - No isolation of components.
- Capabilities need to be given do the initial process.
- NTP example:
  - FreeIPA can setup and run NTP server.
  - Capability needed (`docker run --cap-add=SYS_TIME`).
  - Custom SELinux policy needed to allow `sys_time` capability to `svirt_lxc_net_t`.

# Naming, host, and localhost

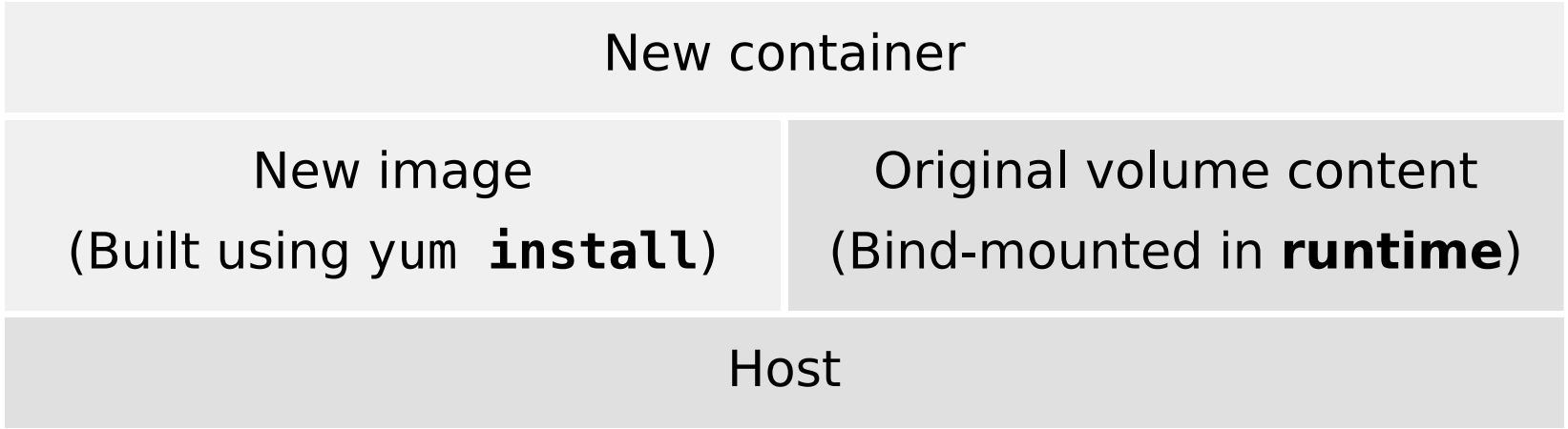
FreeIPA can include DNS server as well.

- It needs to set its own (public) A record in the zone.
- What is the public IP address of service in container?
  - Cannot determine from inside of the container.
  - Be explicit, pass the value to the container.
- Forwarder to host's `127.0.0.1`?
- Or `resolv.conf` pointing to host's 127.0.0.1?
  - Bind to bridge address on the host, or to public interface.

# Upgrades in containers




- Build new image (with `yum install`).
- Remove the old container and run a new one:



# Approach to upgrades

- Since software was installed and not upgraded, upgrade (postinstall) scriptlets in rpms never kick in.
- Config files or data still need to be brought up-to-date.
- Generate `/etc/build-id` in image.
  - Copy it to `/data`
  - Upon next run, detect that different image is used.
- When upgrade is needed during container start:
  - If standalone upgrade tool is available in the project, use it.
  - Parse rpm scriptlets and just run them.
  - They really need to be idempotent.
  - Populate missing (new) locations in volume that new image expects.

# Future work

- Revisit the possibility of `systemd` in the FreeIPA container.
- In FreeIPA, support running some components on separate hosts.
  - Needs to be done in upstream.
  - Orchestration needed.

# Side note: complex service

- Even single-daemon service can be complex.
- SSSD (System Security Services Daemon): client-side to FreeIPA.
- Data on host, mounted to numerous locations:

```
-v /etc/ipa/:/etc/ipa/:ro \
-v /etc/krb5.conf:/etc/krb5.conf:ro \
-v /etc/krb5.keytab:/etc/krb5.keytab:ro \
-v /etc/nsswitch.conf:/etc/nsswitch.conf:ro \
-v /etc/openldap/:/etc/openldap/:ro \
-v /etc/pam.d/:/etc/pam.d/:ro \
-v /etc/passwd:/etc/passwd.host:ro \
-v /etc/pki/nssdb/:/etc/pki/nssdb/:ro \
-v /etc/ssh/:/etc/ssh/:ro \
-v /etc/sssd/:/etc/sssd/:ro \
-v /etc/systemd/system/sssd.service.d:/etc/systemd/system/sssd.service.d:ro \
-v /etc/sysconfig/authconfig:/etc/sysconfig/authconfig:ro \
-v /etc/sysconfig/network:/etc/sysconfig/network:ro \
-v /etc/sysconfig/sssd:/etc/sysconfig/sssd:ro \
-v /etc/yp.conf:/etc/yp.conf:ro \
-v /var/cache/realmd/:/var/cache/realmd/ \
-v /var/lib/authconfig/last/:/var/lib/authconfig/last/:ro \
-v /var/lib/ipa-client/sysrestore/:/var/lib/ipa-client/sysrestore/:ro \
-v /var/lib/samba/:/var/lib/samba/ \
-v /var/lib/sss/:/var/lib/sss/ \
-v /var/log/sssd/:/var/log/sssd/ \
-v /var/run/dbus/system_bus_socket:/var/run/dbus/system_bus_socket \
```

# Side note: complex service (cont'd)

- Configures multiple libraries on the machine.
- Setup in runtime: `ipa-client-install`.
- Special SELinux domain needed to bridge the host/container boundary.
- Separation of install and runtime.
  - The `atomic` tool uses `LABEL INSTALL`/`LABEL RUN` to capture the glue (`docker run` options) needed.

# Conclusion

- Running multiple services in one container is possible.
- Easy testing of wild upgrade scenarios.
  - Even across operating systems.
  - Just pair the image with data.
- The state of the volume can drive the behaviour.
  - Install when empty.
  - Upgrade when created with different image.
- Unless you can depend on orchestration, minimize number of containers and volumes.
- Use native init system or work around it.

# References

- github.com/adelton/docker-freeipa
- www.freeipa.org/page/Docker
- github.com/fedora-cloud/Fedora-Dockerfiles/tree/master/sssd