Composing systems in an automated way with Ansible, Podman, and bootc

Fabio Alessandro "Fale" Locati
Senior Principal Architect, Red Hat

Red Hat

TOC

About me

- Working in IT since 2004, mostly in operations roles
- Active in open source communities
- Fedora core developer since 2010
- Ansible user since 2013
- Immutable linux user since 2016
- Author of 5 books, 4 of which on Ansible
- Senior Principal Architect @ Red Hat

Disclaimers

- Everything we are discussing is fully open source (but also available with Enterprise support)
- Everything we are discussing is architecture independent (x86_64, aarch64, s390x, ppc64le)
- Linux is required (distro does not matter, as long as it has systemd and podman)

Why?

Why not Kubernetes?

- Heavy infrastructure overhead
- Steep learning curve
- Operational complexity

Red Hat

Kubernetes shaped problems

- Provide CaaS to others
- Deployments horizontal autoscaling
- Container auto-placement

Red Hat

Bootc

Red Hat

What is bootc?

- Tooling to turn OCI container images into bootable operating systems.
- Bridges container build workflows and real machines (VMs/bare-metal).
- Supports atomic updates & rollbacks of the whole system image.
- Leverages familiar container registries as distribution channels.
- Fits CI/CD: versioned artifacts, tests, promotions.

How does immutable Linux work?

- OS filesystem is (mostly) **Read-Only**.
- OS updates are **atomic**.
- The OS filesystem can be **reverted** to previous states.
- **User environments and applications** run in isolated, layered containers.

Red Hat

Architecture

- **Input:** Dockerfile/Containerfile → OCI image.
- **bootc:** converts image layers into a bootable rootfs.
- **Artifacts:** disk images (qcow2/raw/vmdk), ISO, or direct install.
- **Runtime:** systemd-managed services, read-mostly system.
- **Lifecycle:** pull new image, switch on reboot, rollback if needed.

Minimal base (Containerfile)

- Start from scratch.
- Start from a bootc-ready base (kernel, initramfs, systemd included).

```
FROM quay.io/fedora/fedora-bootc:latest
```

  - **AlmaLinux**: `https://github.com/AlmaLinux/bootc-images`
  - **Fedora**: `https://gitlab.com/fedora/bootc/base-images`
  - **CentOS**: `https://gitlab.com/redhat/centos-stream/containers/bootc`
  - **Arch**: `https://github.com/bootcrew/arch-bootc`
  - **Debian**: `https://github.com/bootcrew/debian-bootc`
  - **LinuxMint**: `https://github.com/bootcrew/linuxmint-bootc`
  - **OpenSUSE**: `https://github.com/bootcrew/opensuse-bootc`
  - **Ubuntu**: `https://github.com/bootcrew/ubuntu-bootc`

Adding packages

- Use familiar package managers during *image build*, not at runtime.
- Clean caches to keep layers lean and deterministic.
- Example:

```
RUN dnf -y install \
    nebula \
    neovim \
    && dnf -y clean all
```

- Note: this is not an interactive session (`-y` mandatory).

WARNING

- Never use:
    - `dnf -y update`
    - `dnf -y upgrade`
- Ok: single package upgrade

Red Hat

Adding services

- Define systemd units as part of the image.
- Example:

```
COPY myDaemon.service /etc/systemd/system/
RUN systemctl enable myDaemon.service
```

Adding users

- Leverage Systemd sysuser.
- sysuser-fale.conf

```
#Type  Name  ID  GECOS  HomeDirectory  Shell
u fale 1000 "Fale" /home/fale /bin/bash
g wheel - -
m fale wheel
```

- Containerfile

```
COPY sysuser-fale.conf /usr/lib/sysusers.d/fale.conf
```

- https://www.freedesktop.org/software/systemd/man/latest/sysusers.d.html

WARNING

```
RUN useradd -m demo && echo 'demo:demo' | chpasswd
```

Any invocation of `useradd` or `groupadd` that does not allocate a fixed UID/GID may be subject to drift in subsequent rebuilds by default.

Adding users files

- Leverage Systemd sysuser.
- tmpfiles-fale.conf

```
#Type Path           Mode User Group Age Argument...
d /var/home/fale 0700 fale fale -
d /var/home/fale/.ssh 0700 fale fale -
f+ /var/home/fale/.ssh/authorized_keys 0600 fale fale - ssh-rsa AAAAB....CWw==
Z /var/home/fale - - - -
```

- Containerfile

```
COPY tmpfiles-fale.conf /etc/tmpfiles.d/fale.conf
```

- https://www.freedesktop.org/software/systemd/man/latest/tmpfiles.d.html

Linting

```
RUN bootc container lint
```


Red Hat

Building the container

- Container build produces the canonical artifact.
- Keep tags semantic (e.g., 1.2.0) for safe rollouts.
- Example:

```
sudo podman build -t localhost/myos:1.0.0 .
```

Squashing

```
podman build --squash --pull-always .
```

Publishing updates

- New image = new OS version; hosts update atomically.
- Exactly like any other container image:

```
podman push localhost/myos:1.0.0
```

Building an ISO

```
mkdir output
sudo podman run --rm -it --privileged --pull=newer \
    --security-opt label=type:unconfined_t \
    -v ./output:/output \
    -v /var/lib/containers/storage:/var/lib/containers/storage \
    quay.io/centos-bootc/bootc-image-builder:latest \
    --type iso \
    --chown 1000:1000 \
    --rootfs btrfs \
    localhost/myos:1.0.0
```

https://github.com/osbuild/bootc-image-builder

Building a bootable image

```
mkdir output
sudo podman run \
    --rm \
    -it \
    --privileged \
    --pull=newer \
    --security-opt label=type:unconfined_t \
    -v ./output:/output \
    -v /var/lib/containers/storage:/var/lib/containers/storage \
    quay.io/centos-bootc/bootc-image-builder:latest \
    --type qcow2 \
    --use-librepo=True \
    --rootfs btrfs \
    localhost/myos:1.0.0
```

https://github.com/osbuild/bootc-image-builder

Run a bootable image

```
qemu-system-x86_64 \
    -M accel=kvm \
    -cpu host \
    -smp 2 \
    -m 4096 \
    -bios /usr/share/OVMF/OVMF_CODE.fd \
    -serial stdio \
    -snapshot output/qcow2/disk.qcow2
```

Installing bootc OS

```
podman run --rm \
           -v /dev:/dev \
           -v /var/lib/containers:/var/lib/containers \
           -v /:/target \
           --privileged \
           --pid=host \
           --security-opt label=type:unconfined_t \
           quay.io/fale/server:stable \
             bootc install to-existing-root \
               --root-ssh-authorized-keys /target/root/.ssh/authorized_keys
```

Atomic updates & rollback

- Updates are transactional; system switches entirely on reboot.

  ```
  bootc upgrade --apply
  ```

- Rollback path is symmetrical and fast.

  ```
  bootc rollback --apply
  ```

- No partial upgrades or dependency hell on production hosts.
- Possible to switch to a different image:

  ```
  bootc switch --apply quay.io/fedora/fedora-bootc:43
  ```

Some suggestions

- Base OS image + application layer(s).
- Keep image single-purpose (appliance mindset).
- Prefer deterministic package sets and configs.
- Automate!

Automate containers with Ansible, Podman, systemd

Red Hat

What is Podman?

- A daemonless, rootless alternative to Docker
- Donated to the CNCF in November 2024
- Key features
  - Compatible with Docker CLI
  - Native support for OCI containers
  - Native support for Kubernetes objects

What is systemd?

- A system and service manager for Linux (aka PID1)
- Controls system processes, services, and dependencies
- Replaces older init systems (SysV, Upstart)
- Interesting features
  - Manages long-running services efficiently
  - Supports dependency management and auto-restarts
  - Provides robust logging and monitoring with journald
  - Allows extensions for custom kind of resources
- Why Use systemd for container management?
  - Enables native service control for containers
  - Simplifies startup, shutdown, and auto-restart of containers

What is Quadlet?

- A systemd helper for Podman
- Simplifies systemd unit file creation for containers
- Allows easy deployment and management of containerized services
- Technically, Quadlet does not exists (anymore)

Quadlet key features?

- Uses declarative configuration for container management
- Supports auto-restarts and dependencies
- Enables seamless integration with systemd services

Red Hat

Why Quadlet?

- Removes complexity from managing Podman containers via systemd
- Reduces the need for manual unit file configurations
- Ideal for persistent containerized applications

Quadlet base example

```
[Container]
ContainerName=myservice
Image=docker.io/my/service:1.0.0
[Install]
WantedBy=default.target
```

Strategy

- Place a file
- Reload systemd daemons
- Start and enable daemon

Ansible code example

```yaml
- name: Ensure the container launcher is up to date
  ansible.builtin.copy:
    src: myservice.container
    dest: /etc/containers/systemd/myservice.container
    owner: root
    group: root
    mode: '0644'
  register: systemd_daemons
  notify: Restart myservice
- name: Reload systemd daemons if needed
  ansible.builtin.systemd:
    daemon_reload: true
  when: systemd_daemons.changed
- name: Ensure services are started and enabled
  ansible.builtin.service:
    name: myservice
    state: started
    enabled: true

- name: Restart myservice
  ansible.builtin.service:
    name: myservice
    state: restarted
```

Dependencies

```
[Unit]
After=local-fs.target nebula.service
```

Environment variables

```
[Container]
Environment=SECRET_KEY=YOUR_SECRET_KEY
```

Port publishing

```
[Container]
PublishPort=8080:80/tcp
```

Volumes

```
[Container]
Volume=/opt/mysrv:/etc/myservice
```

Wrapping up

Wrapping up

- Kubernetes is good for Kubernetes-shaped problems
- bootc offers a reliable, secure, and stable operating environment
- It is easy to create distros with bootc
- Ansible and Podman can be great to run containers
- Ansible and Podman is a very straightforward solution

Questions?

Email: fale@redhat.com

Fediverse: @fale@fale.io

Red Hat

Links

- https://podman.io/docs/
- https://podman.io/blogs/2023/04/quadlet-tutorial.html
- https://docs.ansible.com/ansible/latest/
- https:
  //fale.io/blog/2023/12/31/share-volumes-between-podman-systemd-services
- https://bootc-dev.github.io/bootc/
- https://docs.fedoraproject.org/en-US/bootc/
- https://fedoramagazine.org/building-your-own-atomic-bootc-desktop/