bootc 101: dalla container image alla tua distro personale

Fabio Alessandro “Fale” Locati

October 25, 2025

EMEA Principal Specialist Solutions Architect, Red Hat

TOC

History

How does it work?

Creating a Containerfile for bootc images

Build and run bootc images

Wrapping up

About me

- IT user since 1996.
- Working in IT since 2004.
- Fedora core developer since 2010.
- Immutable linux user since 2016.
- Fedora Engineering Steering Committee (FESCo) member since 2024.
- EMEA Principal Specialist Solution Architect @ Red Hat.

History

History

- 1988 - POSIX added support “read-only” file systems.
- 2003 - Eelco Dolstra started Nix as a research project.
- 2006 - Armijn Hemel presented NixOS as the result of his Master’s thesis at Utrecht.
- 2013 - Docker made popular the idea of immutable containers.
- 2013 - Alex Polvi created CoreOS.
- 2014 - Red Hat created Project Atomic.
- 2015 - The NixOS Foundation was founded.
- 2018 - Red Hat acquired CoreOS.
- 2024 - Red Hat announced bootc.
- 2025 - bootc ownership was moved from Red Hat to CNCF.

How does it work?

Definitions

- **Container**: a lightweight, standalone, executable package that includes everything needed to run an application—code, runtime, libraries, and dependencies.
- **OCI container**: containers that adhere to a set of standards defined by the Open Container Initiative. The OCI was established in 2015 to standardize container technology to improve compatibility, portability, and interoperability across different environments.
- **Snap**: A universal package format developed by Canonical (the makers of Ubuntu) that allows applications to run in an isolated environment across different Linux distributions.
- **Flatpak**: A framework for building, distributing, and running sandboxed desktop applications on Linux.

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.

Different kinds of immutable Linux

- NixOS
- CoreOS
- Project Atomic
- Fedora Atomic
- Bootc

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.

Architecture in one slide

- **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.

Creating a Containerfile for bootc images

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

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

- Examples

```
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`

Sudoers config

- Leverage sudo config folder ability.
- tmpfiles-fale.conf

```
fale ALL=(ALL) NOPASSWD: ALL
```

- Containerfile

```
COPY --chmod=440 sudoers-fale /etc/sudoers.d/fale
```

Firewalld

- Leverage sudo config folder ability.
- firewalld-public.xml

```
<?xml version="1.0" encoding="utf-8"?>
<zone>
  <short>Public</short>
  <description>For use in public areas.</description>
  <service name="dhcpv6-client"/>
  <service name="nebula"/>
  <service name="ssh"/>
  <forward/>
</zone>
```

- Containerfile

```
RUN dnf install -y firewalld
```

Linting

```
RUN bootc container lint
```

Build and run bootc images

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!
- No, really, automate!

Wrapping up

Wrapping up

- bootc offers a reliable, secure, and stable operating environment at the cost of flexibility.
- bootc is very convinient for big deployment of similar systems (kiosks, labs, cloud, ...).
- It is easy to create distros with bootc.

Questions?

Email: mail@fale.io
Fediverse: @fale@fale.io

Links

- https://bootc-dev.github.io/bootc/
- https://docs.fedoraproject.org/en-US/bootc/
- https://fedoramagazine.org/building-your-own-atomic-bootc-desktop/