Building ISOs from OCI containers

Ondřej Budai, Image Builder @ Red Hat, ondrej@budai.cz

Background

People like using container images as the source of truth for their machines:

- Universal Blue (🕹️ Bazzite, Bluefin, Aurora)
- HeliumOS
- RHEL Image Mode
- Kairos
- Elemental (by Rancher)

The problem is:

No good tooling for ISOs

Existing solutions

(bootc-)image-builder ISOs

- Very opinionated
- Only generates anaconda-based installers
- Tied to Fedora and friends
- Creates the root tree from scratch
- Almost **no** customizations 🤯
- Fragile
- I kinda hate it*

(*) It was my idea to implement the ISOs this way.

🐍 Titanoboa

- Made by ublue folks
- Tied to Fedora and friends
- Separation of concerns is *strange*
- Modifies the container image and uses it as the tree
- There's no container as a source of truth

What do we really want?

Let’s consider bootc and building a disk image as a case study.

The process consists of:

- Building a container image, i.e. defining all content
- Converting it to a disk image, i.e. exploding a tarball into a partitioned disk in **the right way**™

➡️ **The container image is always the source of truth.**

Even the partitioning and bootloader setup can be specified by the container!

What do we really want?

The ISO building process consists of:

1. Building a container image
2. Converting it to an ISO in **the right way™**

The right way:

- squashfs the container image
- copy kernel
- copy bootlader
- copy initramfs
- configure the bootloader
- set the ISO label

We need a *contract* between these 2 steps!

Introducing Container-native ISO Contract

- a tiny specification that allows a container image to be the source of truth for ISO peculiarities
- based on how bootc containers are laid out
- opinionated only when there are pragmatic reasons
  - El Torito hybrid ISO setup + GRUB2 is good for majority of use cases

- the most important bits come from the container image
  - shim + GRUB2
  - kernel + args
  - initramfs
  - subset of GRUB2 settings
  - label
  - squashfs (of the whole container image)

Introducing Container-native Contract v0.1.0

- Only x86_64 is currently supported.
- The kernel is expected to be in `/usr/lib/module/*/vmlinuz`. If there are multiple kernels, the behavior is unspecified. This is to be specified in a future version of this contract. The kernel is put in `/images/pxeboot/vmlinuz` in the ISO.
- The initramfs is expected to be next to the kernel with the filename `initramfs.img`. The initramfs is put in `/images/pxeboot/initrd.img`.
- The UEFI vendor is specified by a directory name in `/usr/lib/efi/shim/*/EFI/$VENDOR`. If there are multiple directories, the behavior is unspecified. The `BOOT` directory is always ignored.
- Shim and grub2 EFI binaries (`shimx64.efi`, `mmx64.efi`, `gcdx64.efi`) are expected to be in `/boot/efi/EFI/$VENDOR`.
- GRUB2 modules are expected to be in `/usr/lib/grub/i386-pc`.
- Required executables are `podman`, `mksquashfs`, `xorriso`, `implantisomd5`, `grub2-mkimage`, and `python`.
- The container image is converted to a squashfs archive and put into `/LiveOS/squashfs.img` in the ISO.
- Additional configuration can be written into `/usr/lib/bootc-image-builder/iso.yaml` in YAML format. The file currently supports 2 top-level keys:
  - `label` (string): Label of the ISO
  - `grub2` (object): GRUB2 configuration, supports the following keys:
    - `default` (string): Default menu item
    - `timeout` (string): Default timeout (in seconds)
    - `entries` (array of objects): GRUB2 menu entries with the following keys (all are required):
      - `name` (string): Name of the entry
      - `linux` (string): Path to the kernel + kernel arguments (the path is always `/images/pxeboot/vmlinuz` in this version of this spec)
      - `initrd` (string): Path to the initramfs (the path is always `/images/pxeboot/initrd.img` in this version of this spec)

Introducing Container-native Contract v0.1.0

`image-builder` and its `bootc-generic-iso` (*) image type implement the conversion step for the contract

> Project placement time!
> image-builder is an open source project for building disk images, ISOs and more of Fedora, and CentOS Stream and its derivatives. We support both mutable and bootc-based systems.
> Find the project at https://github.com/osbuild/image-builder-cli

(*) The name needs a discussion.

Let’s build a container fulfilling the contract

```
FROM ghcr.io/ublue-os/bazzite:latest
RUN dnf install -y dracut-live livesys-scripts grub2-efi-x64-cdboot xorriso isomd5sum && dnf clean all

RUN sh -c 'kernel=$(kernel-install list --json pretty | jq -r ".[] | select(.has_kernel == true) | .version"); \
    DRACUT_NO_XATTR=1 dracut -v --force --zstd --reproducible --no-hostonly \
    --add "dmsquash-live dmsquash-live-autooverlay" \
    "/usr/lib/modules/${kernel}/initramfs.img" "${kernel}"' && \
    mkdir -p /boot/efi && cp -av /usr/lib/efi/*/*/EFI /boot/efi/

RUN sed -i "s/^livesys_session=.*/livesys_session=kde/" /etc/sysconfig/livesys && \
    systemctl enable livesys.service livesys-late.service

RUN <<EOF
mkdir -p /usr/lib/bootc-image-builder
cat > /usr/lib/bootc-image-builder/iso.yaml << 'YAML'
label: "Bazzite-Live"
grub2:
  timeout: 10
  entries:
    - name: "Bazzite Live ISO"
      linux: "/images/pxeboot/vmlinuz quiet rhgb root=live:CDLABEL=Bazzite-Live enforcing=0 rd.live.image"
      initrd: "/images/pxeboot/initrd.img"
```

Let’s build a container fulfilling the contract

```
# podman build -t bazzite-live
# image-builder build --bootc-ref localhost/bazzite-live bootc-generic-iso
```

Quite simple, right?

image-builder is also available as a container image, so you just need podman.

Current state

Upstream: https://github.com/ondrejbudai/bootc-isos

Feedback welcome!

Matrix: #image-builder:fedoraproject.org

Future plans

- alignment with the latest bootupd changes
- custom buildroots (imagine minimal live ISOs)
- multiple kernels support
- squashfs options
- erofs
- different bootloaders

Current state

Upstream: https://github.com/ondrejbudai/bootc-isos

Feedback welcome!

Matrix: #image-builder:fedoraproject.org

❤️ Big thanks to:

- Simon de Vlieger
- Michael Vogt
- The Image Builder team
- The Universal Blue community