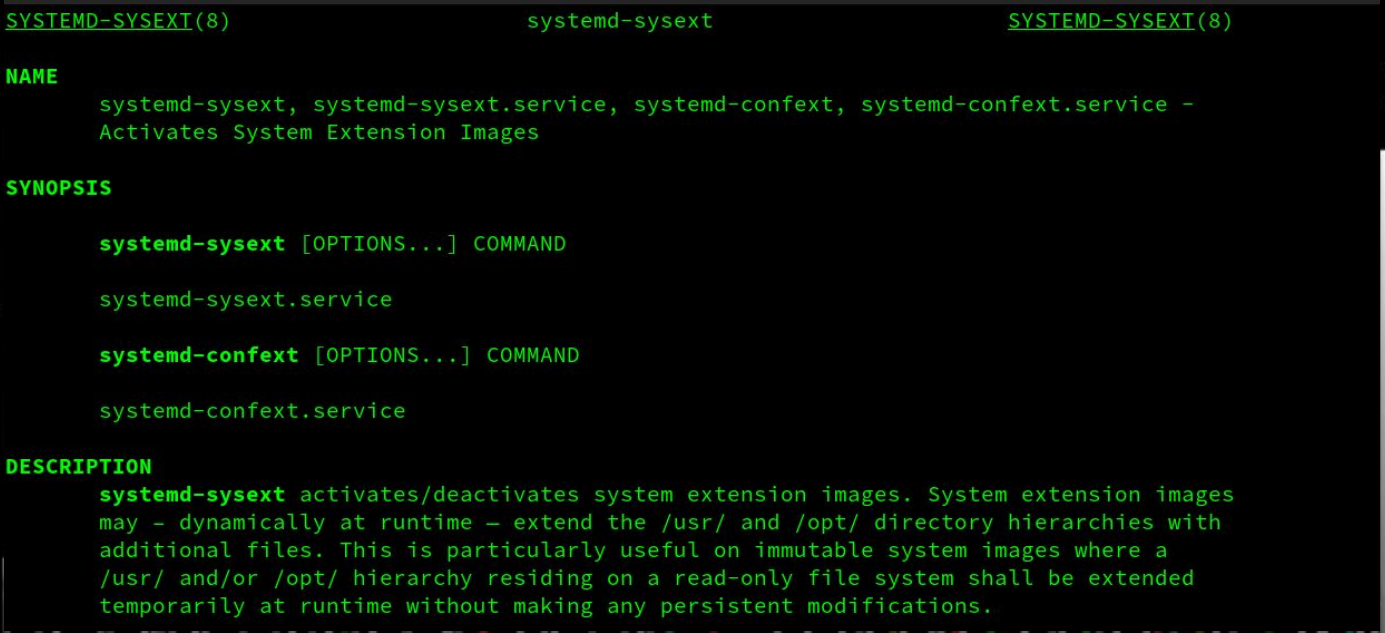


```
SYSTEMD-SYSEXT(8)                    systemd-sysext                    SYSTEMD-SYSEXT(8)

NAME
       systemd-sysext, systemd-sysext.service, systemd-confext, systemd-confext.service -
       Activates System Extension Images

SYNOPSIS

       systemd-sysext [OPTIONS...] COMMAND

       systemd-sysext.service

       systemd-confext [OPTIONS...] COMMAND

       systemd-confext.service

DESCRIPTION
       systemd-sysext activates/deactivates system extension images. System extension images
       may – dynamically at runtime — extend the /usr/ and /opt/ directory hierarchies with
       additional files. This is particularly useful on immutable system images where a
       /usr/ and/or /opt/ hierarchy residing on a read-only file system shall be extended
       temporarily at runtime without making any persistent modifications.
```

,

Room Friedrichshain III | 2024-06-18 | Speaker: Krish

Hi, I'm Krish

Krish Jain

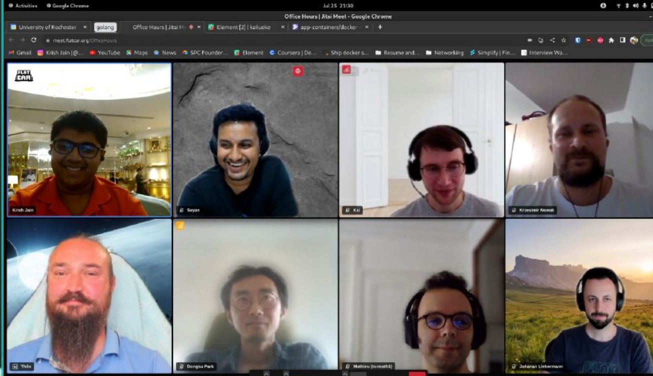
Previous intern working on Flatcar Linux project (a project out of Kinvolk, now Microsoft)

- this is what I will be talking about

Currently at Chainguard (backed by Sequoia Capital) securing the software supply chain.

LinkedIn: linkedin.com/in/krishjain02/

Email: krish.jain@rochester.edu



Flatcar Container Linux

(Fork of CoreOS Container Linux)

For context: why Flatcar Container Linux?

Minimal distribution for containers

Reduced dependencies

Less base software to manage

Reduced attack surface area

Secure, immutable file system

Read-only /usr partition

No package installation or modification of base OS files

Removes entire category of security threats (e.g., runc vulnerability CVE-2019-5736)

Automated, streamlined updates

Security patches

Atomic updates and rollbacks

Co-ordinated with Kubernetes control plane (update operator)

Declarative provisioning

First boot setup from declarative configuration

Immutable infrastructure (no custom per-node changes during production)

Repeatable deployment

Ignition Config

- ❑ JSON format. Declaration of files, systemdunits, sysext images, networks, users, filesystems, and partitions

- ❑ Referencing data from external resources.

- ❑ Applied from initramfs (first-boot flag file for GRUB sets kernel parameter)

- ❑ Compare to cloud-initwhich runs after the initramfs, and on every boot

Butane Config

- ❏ Friendlier YAML format with extras (octal permissions,
- ❏ variables for metadata)

Transpiled to Ignition JSON through transpiler"ct"

```
docker run --rm -i quay.io/coreos/butane:latest < your_config.yaml >
your_config.json
```

- ❏ Info: https://www.flatcar.org/docs/latest/provisioning/config-transpiler/

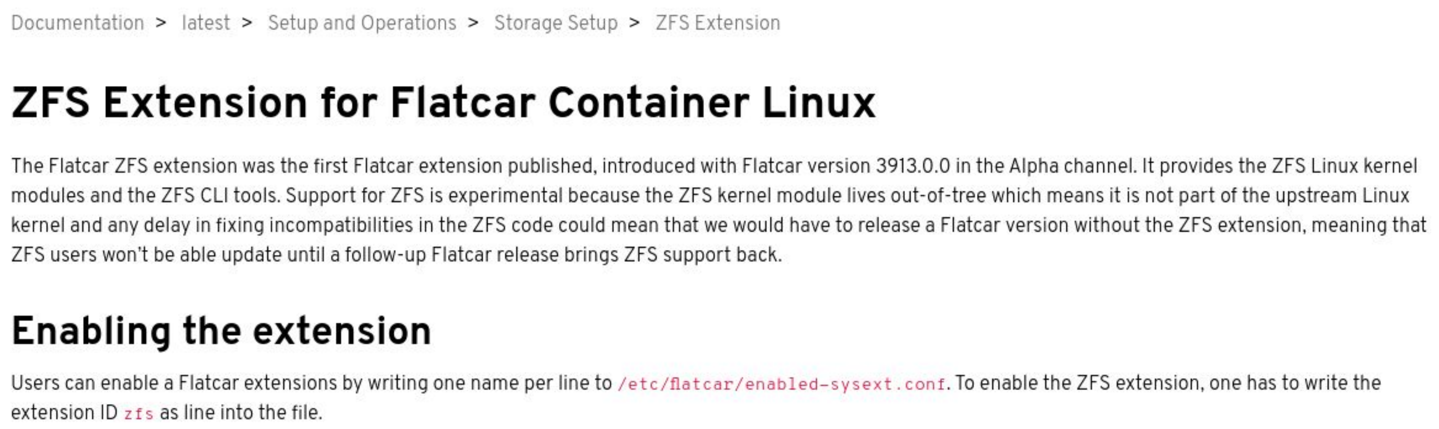
Butane Config Example

```
variant: flatcar
version: 1.0.0
storage:
  files:
    - path: /etc/extensions/mydocker.raw
      mode: 0644
      contents:
        source: https://myserver.net/mydocker.raw
    - path: /etc/systemd/system-generators/torcx-generator
  links:
    - path: /etc/extensions/docker-flatcar.raw
      target: /dev/null
      overwrite: true
    - path: /etc/extensions/containerd-flatcar.raw
      target: /dev/null
      overwrite: true
```

After boot you can see it loaded in the output of the systemd-sysext command: You can reload the sysext images at runtime by executing systemctl restart systemd-sysext

```
HIERARCHY EXTENSIONS SINCE
/opt      none      -
/usr      mydocker   Wed 2022-03-23 14:16:37 UTC
```

For podman/python we already have ebuilds within flatcar's repo.



Documentation > latest > Setup and Operations > Storage Setup > ZFS Extension

ZFS Extension for Flatcar Container Linux

The Flatcar ZFS extension was the first Flatcar extension published, introduced with Flatcar version 3913.0.0 in the Alpha channel. It provides the ZFS Linux kernel modules and the ZFS CLI tools. Support for ZFS is experimental because the ZFS kernel module lives out-of-tree which means it is not part of the upstream Linux kernel and any delay in fixing incompatibilities in the ZFS code could mean that we would have to release a Flatcar version without the ZFS extension, meaning that ZFS users won't be able update until a follow-up Flatcar release brings ZFS support back.

Enabling the extension

Users can enable a Flatcar extensions by writing one name per line to `/etc/flatcar/enabled-sysext.conf`. To enable the ZFS extension, one has to write the extension ID `zfs` as line into the file.

Immutable Infrastructure

>>Immutable"ness"<<

- Flatcar Container Linux has a strong focus on backwards compatibility

- Pros:
  - Reproducible and consistent configuration, e.g., matching a git repository. Flatcar ships a fixed set of software and users should rely on containers for the rest

  Cons
  - Since Flatcar ships a fixed set of software versions, users have to rely on containers for everything
  - Limiting if for instance you need to run a different version of docker/containerd or other OS level software
  - To run on clouds like AWS/Azure/GCP Flatcar needs the cloud vendor tools like Azure's WAAgent but we can't pack all of them into the base

Let’s break this down (and what the build system I created solves)

❏ User provides custom software

- ❏ While most software is deployed as containers, this is not possible for certain host-level software such as the container runtime itself
- ❏ One had to place binaries under /opt/bin and keep track of them for updating, or use Torcx to switch the inbuilt Docker/containerd version to a custom Torcx bundle
- ❏ Now we removed it because with systemd-sysext there is now a more generic solution for IT
- ❏ Flatcar’s inbuilt Docker/containerd versions are in fact systemd-sysext images already :) , so that they will fully disappear when disabled
- ❏ To help users extend Flatcar with systemd-sysext, we provide build recipes for common software projects and publish prebuilt extension images in the sysext-bakery repository . (Only static, I worked on the build system for this - build_sysext)
- ❏ Since the lifecycle of these extensions is decoupled from Flatcar OS updates, **user-provided extensions** should consist of static binaries instead of linking against OS libraries.
- ❏ Extensions can be updated with systemd-sysupdate , and the sysext-bakery repository provides the configuration to set it up.

❏ Cloud vendor tools

- ❏ To make Flatcar work on the various clouds we often need the OEM images to contain integration software provided by the cloud vendor. Adding these to the base image would waste disk space for all users and the old approach was to put these binaries on the Flatcar OEM partition. The problem was that there was no update/rollback mechanism for the scattered files and the custom location was also not ideal for a good integration due to diverging from an expected standard path.
- ❏ Using my build system - build_sysext
- ❏ We are already updating of OEM specific tools
- ❏ Now the cloud vendor tools in Flatcar are layered on top of the /usr partition through systemd-sysext images. They are covered by the Flatcar A/B update/rollback mechanism and provided as additional update payloads by our update server . The extensions are coupled to the OS version to ensure that they are compatible and, therefore, can make use of dynamic linking to save disk space.
- ❏ Having established a mechanism for A/B-updated extensions that are bound to the OS version, Flatcar has become more modular. In the past we had to find a compromise between user demands and the image size. The first optional Flatcar extension we introduced provides the kernel drivers and CLI utilities for the ZFS out-of-tree filesystem. We plan to make more CLI tools available such as htop or tmux and cover more use cases with a Podman and Incus extension. The NVIDIA kernel driver is also a candidate for a Flatcar extension. At the same time we can look into reducing the base image size by splitting out some less common parts such as sssd and Kerberos into extensions, likely pre-enabled for backwards compatibility.

Much todo?

❏ Extension Loading and System Boot-Up: Extensions currently load late during the boot-up process, requiring workarounds to apply necessary settings. Proposing to mount extension overlays during the initrd stage for a fully configured system at boot.

Stability and Integrity of Extensions: Issues with overlay mounts disappearing during extension reloads will be addressed using the new Linux mount beneath API. Additionally, using dm-verity to ensure the integrity of extension images with more granular enforcement policies.

Systemd-sysupdate and Downgrade Support: Implementing systemd-sysupdate to run on first boot from initrd for downloading missing extensions. Introduction of downgrade support in the manifest format to retract updates if needed.

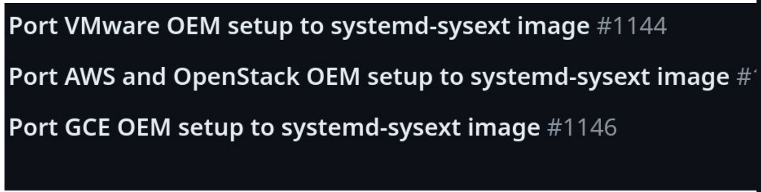
Systemd-confext and Mutable Overlay Mode: Introduction of a mutable overlay mode in systemd-confext and systemd-sysext to manage configuration changes more flexibly, accommodating both traditional and image-based OS requirements.

Flatcar Innovations and Community Involvement: Flatcar is advancing with new features available in the Stable, Alpha, and Beta channels, aiming to split into composable OS layers. Encouragement for community participation in systemd-sysext feature testing and contribution to the sysext-bakery repository.

Thinking Back About The Problem

- ❑ We have a solution to extend Flatcar that provides a robust update mechanism and and integrates well with base OS.
- ❑ With systemd-sysext we can overlay extensions on top of the read-only /usr partition. “New Package Request”’s issues: fail2ban, podman, incus, kata-containers etc
- ❑ Allows us to address long-standing feature requests and find new solutions outside of previous compromises.
- ❑ The team has mentioned systemd-sysext in many conference talks. Now after my work and the work following that by the team it works :)

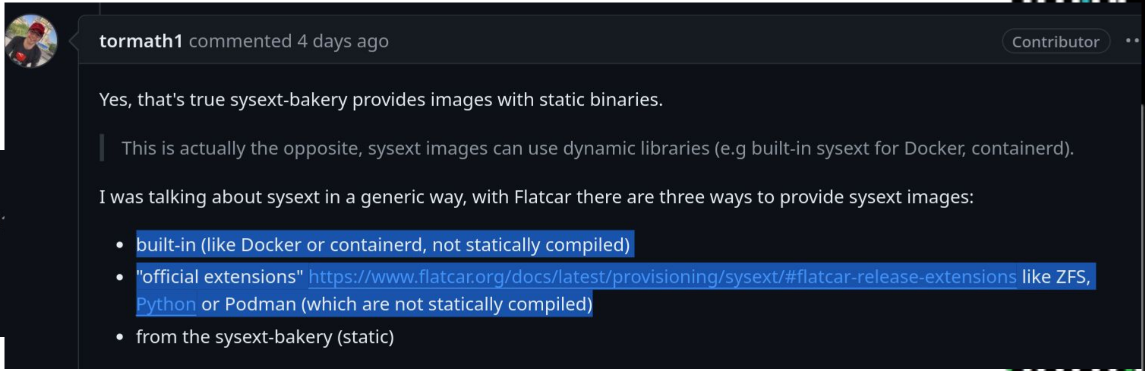
What I worked on ↓->



Port VMware OEM setup to systemd-sysext image #1144

Port AWS and OpenStack OEM setup to systemd-sysext image #

Port GCE OEM setup to systemd-sysext image #1146



tormath1 commented 4 days ago

Yes, that's true sysext-bakery provides images with static binaries.

> This is actually the opposite, sysext images can use dynamic libraries (e.g built-in sysext for Docker, containerd).

I was talking about sysext in a generic way, with Flatcar there are three ways to provide sysext images:

- built-in (like Docker or containerd, not statically compiled)
- "official extensions" https://www.flatcar.org/docs/latest/provisioning/sysext/#flatcar-release-extensions like ZFS, Python or Podman (which are not statically compiled)
- from the sysext-bakery (static)

Demo (we will work through the docs)

- build_sysext is to build OS dependent sysexts (like docker, vendor tools, official Flatcar extensions (zfs, Incus is one), kmods, GUI's etc). Contrary to user-supplied sysexts, these need careful integration with the base OS. build_sysext is not meant to be a generic packaging tool; ebuilds built into sysexts with this tool will always need some adoption.

  Todo:
  The build_sysext tool is now used for the OEM and the internal Docker/containerd systemd-sysext image.
  For Docker and containerd we need to make sure that the files are correctly labeled for SELinux to work in enforcing mode.

Summary

- ❑ Immutable Infra possible even for stateful
- ❑ systems Flatcar Container Linux already simplifies
- ❑ OS maintenance through immutable A/B updates and systemd sysext
- ❑ Choose your strategy for bundling packages onto base OS.

Thank you!

Krish Jain

LinkedIn: linkedin.com/in/krishjain02/

Email: krish.jain@rochester.edu

Internship Blog Post: https://www.flatcar.org/blog/2023/07/summer-2023-my-internship-experience/

Project Website: flatcar.org

GitHub Repos: flatcar

Matrix Room: flatcar:matrix.org to chat about sysexts!