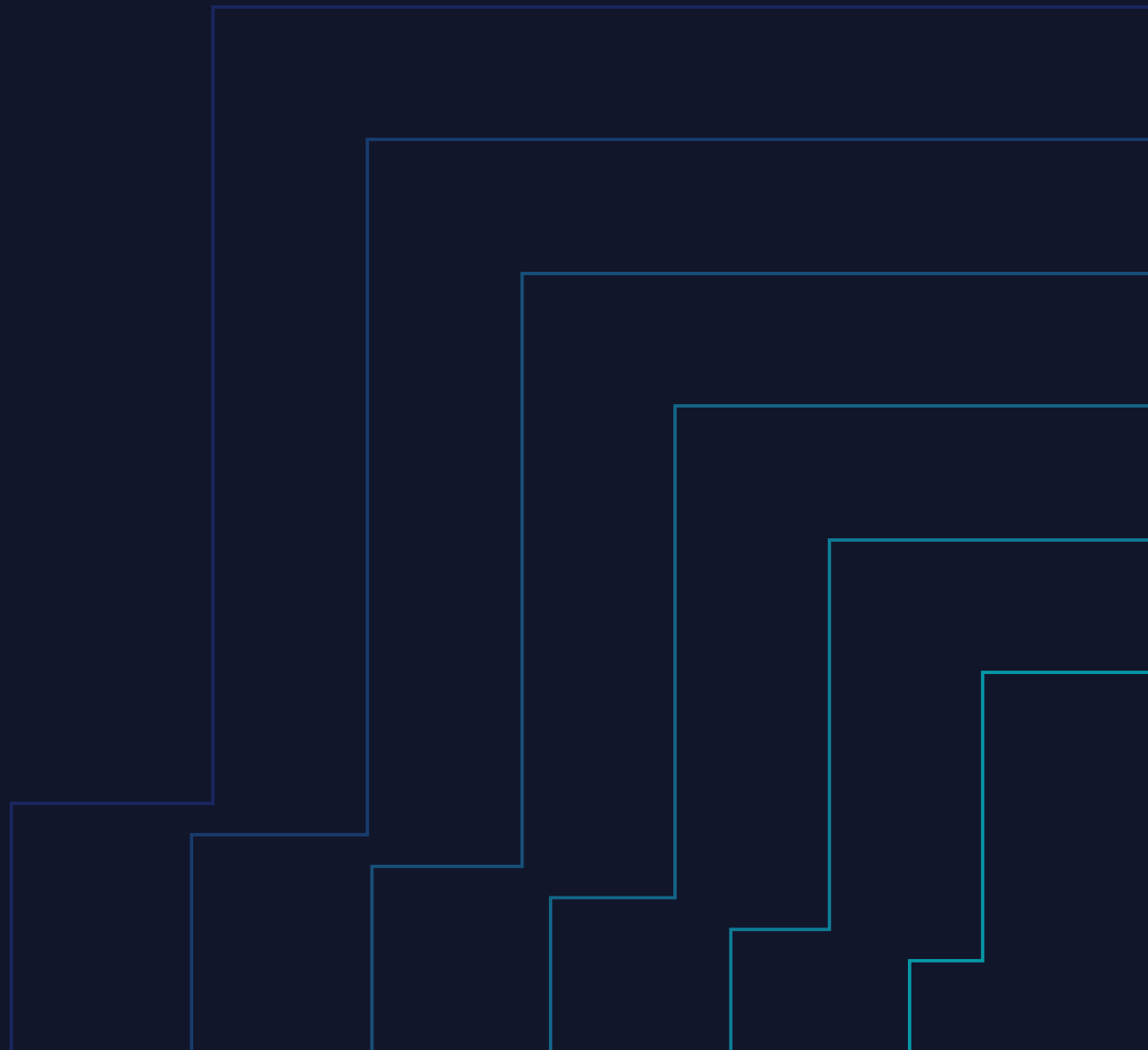
Deploy software with systemd-sysext

Introduction to OS extensions with sysext images

FrOSCon - 21 August 2022



I'm Kai,

one of the Flatcar Container Linux maintainers

Kai Lüke

Senior Software Engineer, Microsoft

Email: kailuke@microsoft.com

GitHub: pothos

flatcar.org

Agenda

Problem Statement

Approaches

systemd-sysext intro

Use cases:

- User OS extension – custom Docker binaries
- OS component selection – Docker or containerd
- Official OS extensions – Cloud vendor tools, K8s

Deploying Software on Linux… without Packages

- Traditional packages are not always desired or possible
  - Many distros → many different packages
  - Package installation introduces varying OS state, can be fragile
  - Some distros have no package manager, e.g., Flatcar
- Numerous solutions seem to exist
  - Containers: Docker, containerd, Podman, cri-o, lxc, systemd-nspawn
  - Ubuntu has Snap
  - For the desktop we have Flatpak
  - systemd has Portable Services
  - Good old static binaries

Containers

- Reduce system dependencies, isolate from the host
- Integration into host possible by starting container from systemd unit, and tweaking the isolation to some extend
- Problem: Make no CLI binaries available to the host, e.g., CNIs like Cilium provide them as extra static binaries
- Need to keep track of scattered systemd unit files and CLI binaries when updating

Snap

- Single image file
- Can ship system services and CLI tools
- However, mostly only used on Ubuntu…

Flatpak

- Mainly meant for desktop GUI applications
- No system services
- CLI usage possible but not nice at all

  PATH="$PATH":~/.local/share/flatpak/exports/bin

Systemd Portable Services

- Reduce system dependencies, isolate from the host as needed but always have their own filesystem tree
- Good integration with the host, systemd unit files get copied to the host
- Can be layered internally with extension images
- Problem: Make no CLI binaries available to the host

Static Binaries

- Often not a bad solution
- But a bit difficult to keep track of systemd unit files, binaries, resource files, etc. when updating

What is systemd-sysext?

- Overlay filesystem images for /usr/ or /opt/
- Mainly meant for CLI/GUI binaries (for now)
- Useable to extend a Portable Service's internal filesystem
- Instead of using it in addition to a container or Portable Service, a small workaround makes shipping system services possible → upstream solution could be to create the overlay earlier
- General note: Be careful not to overlay system files such as libraries, generic images should use static binaries

Systemd-sysext Details

- Allowed formats: .raw filesystem image (squashfs, ext4, ...), .raw GPT partition image, directory, or btrfs subvolume
- Must have a /usr/lib/extension-release.d/extension-release.$NAME file in it with ID/VERSION_ID matching the host
- Stored in /etc/extensions/, /run/extensions/, /var/lib/extensions/, /usr/lib/extensions/, ... first one wins (masking possible)
- Loaded by systemd-sysext.service or systemd-sysext merge/unmerge/refresh (--force)

Systemd-sysext Matching Logic

- Depends on what is set in /etc/os-release
- Strong coupling to OS version is possible for dynamic linking or other type of dependency on host
- Can use self-defined SYSEXT_LEVEL instead of version
- In progress: Make OS ID optional, add ARCH matching
- No semver comparisons, just equality for now

Systemd-sysext Workaround For Services

- Flatcar ships a service to reload units from sysext images:

```
[Unit]
BindsTo=systemd-sysext.service
After=systemd-sysext.service
DefaultDependencies=no
ConditionDirectoryNotEmpty=|/etc/extensions
…
ConditionDirectoryNotEmpty=|/usr/lib/extensions
[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/usr/bin/systemctl daemon-reload
ExecStart=/usr/bin/systemctl restart --no-block sockets.target timers.target multi-user.target
[Install]
WantedBy=sysinit.target
```

Systemd-sysext Updates

- Replacing the file and reloading live should work
- Maybe better is to have a version suffix
- Accompanying systemd-sysupdate tool can be used
  - Local update configuration (“What goes where when?”)
  - Remote HTTP server with manifest file and new versions
  - For example, let it download to /var/myextension/myextension-VERSION.raw and have it update the /var/myextension/myextension-current.raw symlink, while you have set up a static symlink /etc/extensions/myextension.raw pointing to it

Use Cases

Flatcar Container Linux

- Image-based A/B auto updates and rollbacks for the read-only /usr/ partition
- Declarative first-boot configuration with Ignition JSON (can be transpiled from CLC/Butane YAML) applied from initramfs

Deploy a Custom Docker Version

- Flatcar has Torcx to swap out Docker versions but now users can use systemd-sysext
- Helper script in github.com/flatcar-linux/sysext-bakery to download official Docker release binaries, add systemd units, and sysext metadata file

```
tar -xf "docker-1.2.3.tgz" -C mydocker
mv mydocker/docker/* mydocker/usr/bin/
create /usr/lib/systemd/system/ and /usr/share/containerd/config.toml files ...
{ echo ID=flatcar ; echo SYSEXT_LEVEL=1.0 ; } > mydocker/usr/lib/extension-release.d/extension-release.mydocker
mksquashfs mydocker mydocker.raw -all-root
```

Container Runtime Selection

- Flatcar also used Torcx provide Docker+containerd together but now we plan to use systemd-sysext to split them up
- Users could, e.g., disable Docker from their Ignition config by “masking” the sysext that Flatcar provides

```
storage:
  directories:
    - path: /etc/extensions/docker-flatcar
```

Cloud Vendor Tool Updates

- Flatcar currently has cloud vendor tools on an OEM partition that isn't A/B auto-updated like the rest of the OS
- We plan to provide sysext images and store them on the OEM partition with a migration path
- The update service and an early-boot service would manage them to delete old ones and active the current one
- We get much better integration because the OEM tools appear under /usr/ and can use dynamic linking safely

Future Ideas

- Optional Flatcar extensions such as Kubernetes (Kubelet and CNI binaries)
  - Can be stand alone and independent from Flatcar version, updated with systemd-sysupdate
  - Or could benefit from strong OS coupling by integrating it into the update service, and being covered in our release tests

Flatcar sysext docs:

flatcar.org/docs/latest/provisioning/sysext/

Thank you

Kai Lüke

Senior Software Engineer, Microsoft

Email: kailuke@microsoft.com

GitHub: pothos