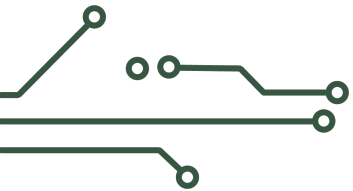
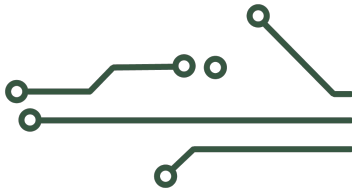
ASG
2025



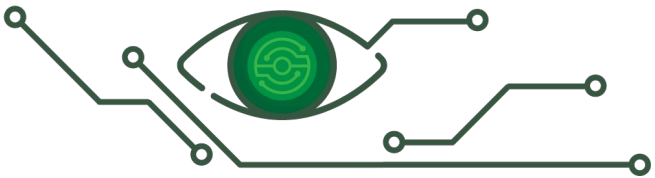
OS As a Service at Meta Platforms



Serge Dubrouski
Marcello Barnaba

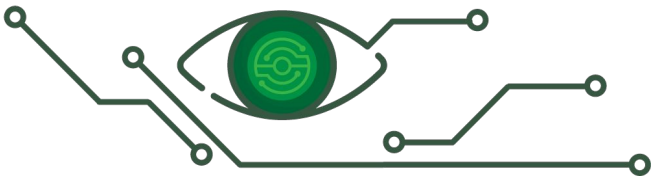
Why care about the OS (vs ~~Metaverse~~ AI)?

- OS enables services.
- OS is everywhere
  - Hosts, VMs, Service containers
- OS is a means to an end, not an end in itself.
- Need regular, reliable, and transparent OS lifecycle management.



OS Management Challenges

- Most OS software is developed and built externally.
- Release cadence is out of our control.
- Require regular OS updates (Compliance, features, fixes, etc)
- Avoid impacts to services / workloads.
- Need to accommodate major OS updates.
- Has to scale well

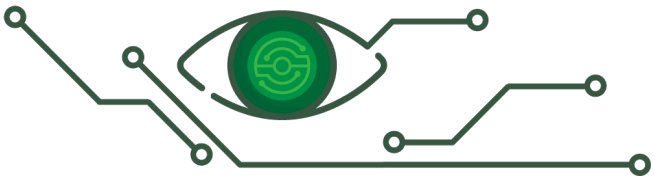


Strategies

- More OS instances / containers
- Image based deployments
- Constrain OS lifecycle management operations.
- Automate everything.

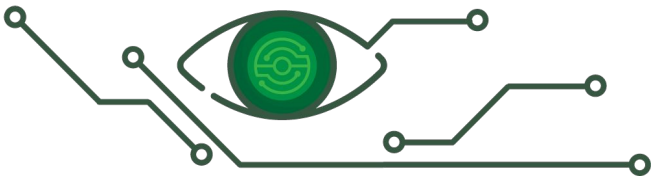
Tooling

- OS Snapshots
  - Rolling OS Updates (ROU)
- OS Image builds:
  - Antlir & Container Manifests (CMs)
- OS Deployments & Updates:
  - Within Containers
  - On bare-metal hosts



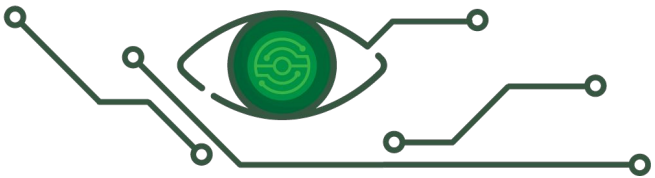
OS Snapshots (ROU): Why we need it

- Combination of upstream / external software.
  - CentOS Stream: Core OS rpms.
  - Vendors RPMs: Nvidia, HP, Google, Microsoft, etc.
  - CentOS Hyperscale SIG: Meta open source efforts.
- Can’t allow direct access to external software.
  - Need Third-Party Agreements: legal prerequisite.
  - Need to verify authenticity: check rpm signatures.
  - Need to control availability: upstream could delete things we depend on.
  - Need to test updates before deploying them.
- Ensure regular updates: Avoid one-off manual imports.



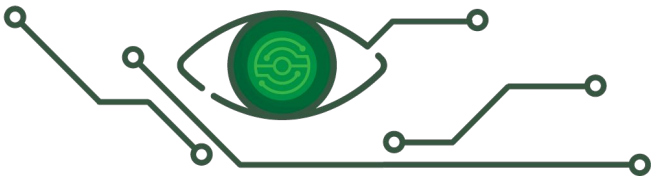
OS Snapshots (ROU): What it does

- Tooling to lifecycle manage upstream / external OS software.
- Defines major releases, ex: `centos9`
  - A set of upstream repos.
- Creates major release snapshots every 2-3 weeks.
  - Mirrors external repos internally to ensure software availability.
  - https://$SERVER/yum/centos/9/rollingupdates/$RELEASE-DATE/$REPO/
- Verifies rpm signatures.
- Tests new snapshots.
- Releases snapshots for use by Antlir and Chef.



OS Image Builds (Antlir): Why we need it

- Lots of existing build mechanisms: docker, podman, etc.
- Lots of existing configuration mechanisms: chef, puppet, etc.
- No existing image build mechanisms meet our requirement:
  - Combining OS Snapshots (rpms) with internal software.
  - Support Chef for configuration management.
  - Host, host services, and container deployments.
  - Source repo (fbsource) build determinism.
  - OS version agnostic builds.

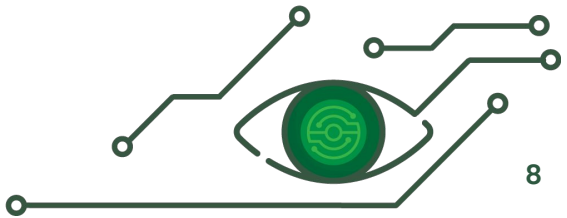


OS Image Builds (Antlir): What it does

- Powerful buck2 based image build and packaging system: Docker build on steroids.
- Provides bzl APIs to build images:
  - Install: RPMs, fbpkgs, and fbsource software.
  - Configure software / services in images.
  - Run chef solo bundles.
  - Layer and compose image artifacts.
- Provides bzl APIs to package images:
  - Provisioning images
  - Service images
  - Initrd/Bootloader/kernel images

```
image.layer(
    name = "ai-overlord",
    parent_layer = "//ai/overlord/base:image",
    features = [
        feature.rpms_install(rpms = “fb-metamate”),
        systemd.install_unit("fb-metamate.service"),
        feature.chef_solo(bundle = "fb_ai_overlord_bundle"),
        feature.clone(
            path = "/usr/local/alternate-reality",
            src_layer = "//metaverse/base:image",
        ),
        feature.install(
            src = ":empty",
            dst = "/usr/sbin/nologin",
        ),
        feature.user_add(
            home_dir = "/",
            primary_group = "systemd-network",
            shell = SHELL_NOLOGIN,
            username = "systemd-network",
        ),

    ],
)
```

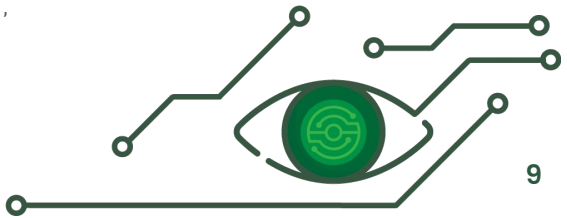


OS Image Builds (Antlir): What it does

- Builds are deterministic based on a code commit.
  - Snapshots rpm repositories state into code repo.
  - Snapshots fbpkg releases state into code repo.
- Image definitions are OS-agnostic.
  - Easily build images based on different OS releases / snapshots.
- Antlir is an open source:
  - https://github.com/facebookincubator/antlir
  - Can bult OCI images using skoper
  - Tightly coupled with Buck

```
# Repo, RPM, and fbpkg snapshots:
$ buck2 targets repo//bot_generated/antlir/...
...
repo//bot_generated/antlir/rpm/fast_snapshot/repos/centos/9/rollingupdates/20240912/BaseOS/x86_64/os:centos/9/rollingupdates/20240912/BaseOS/x86_64/os
...
repo//bot_generated/antlir/rpm/fast_snapshot/rpms/09/centos-stream-release:0-9.0-26.el9.noarch
...
repo//bot_generated/antlir/fbpkg/db/main_db/container.image.sendstream.base.c9/stable:stable

# OS-agnostic image builds
image.layer(
    name = "image",
    ...
)
antlir2_configured_alias(
    name = “image.centos9”,
    actual = “:image”,
    default_os = “centos9”,
)
antlir2_configured_alias(
    name = “image.centos10”,
    actual = “:image”,
    default_os = “centos10”,
)
```

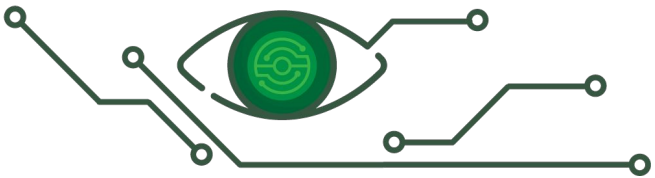


OS Deployments: MetalOS

🧭 **North star: What we build is what we test and what we run in prod**

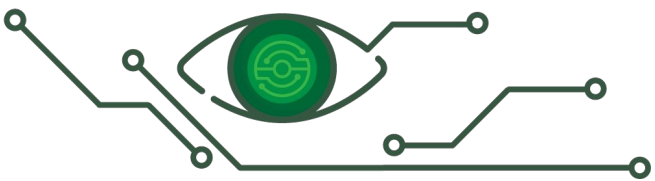
- Image-based CentOS designed to run Linux Containers
- Assembled from coarse 🧩 pieces
  - UKI Bootloader, kernel, initrd, rootfs, service containers
- Goals:
  - Well defined host blueprint and lifecycle
  - Minimize on-host updates
  - Be as **Read Only** and **stateless** as possible
  - Pre-build and CI/CD test OS pieces before deployment

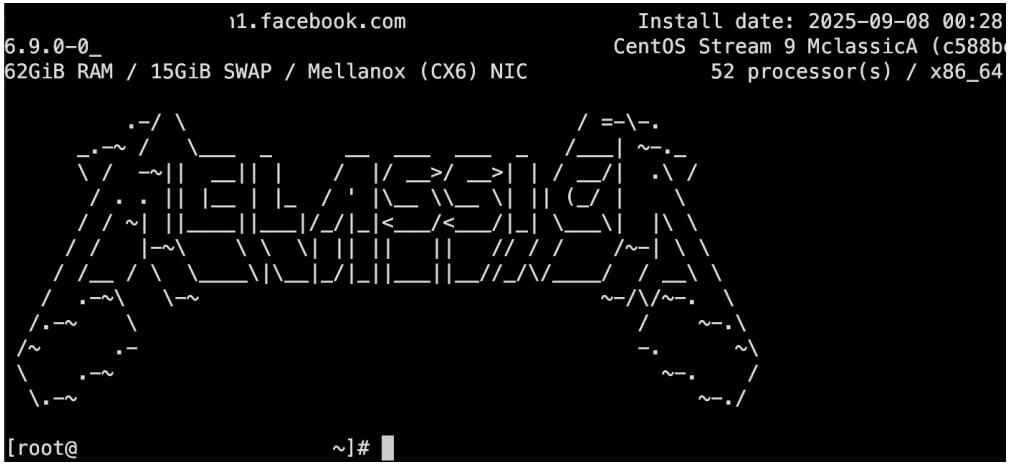
->> We can't migrate to this scheme overnight! 😰



OS Deployments: enter ~~sandman~~ M*classic*A

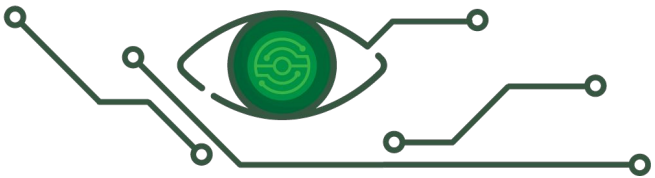
- 🎸 **Metal** frame for a 🎻 **Classic** OS
- Assembled from pieces defined in the host blueprint
- Lifecycle actuation: `metald`; OS assembling: `initrd`
- **Still R/W** and **stateful**, so allows Chef to install software and write configuration
- ⏩ Our migration strategy





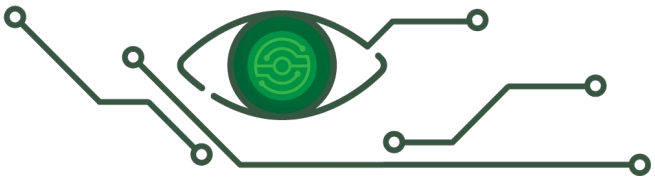
OS Updates: maintenance trains

- 🎅 visits with presents, 🚄 visit hosts with boxcars full of upgrades
- Every host is visited **every 45 days** and delivered fresh OS pieces
- Deliver safely non-critical fixes or high-impact upgrades
- Failure is contained
- Automated *handbrake* mechanisms



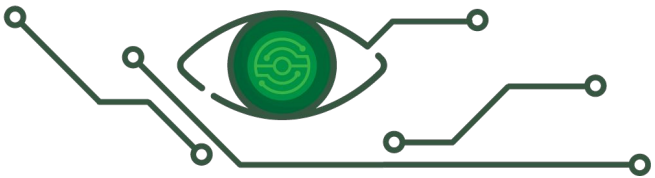
OS Updates: in place updates still required

- Even with a 45 days window, hosts require in-place updates.
- 👨‍🍳 `chef`: *legacy* - one-size-fits-all every-15-minutes-runs
- 🕵️ `host_agent`: workload-centric
- 🎸 `metald`: OS-centric
- Control planes roll-out changes instructing `host_agent` and `metald`



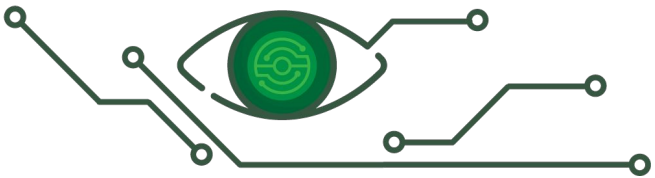
OS Updates: Where we're headed

- Move software and config management out of Chef
- Leverage `metald` and `host_agent` lifecycle and constraints
- Further reduce scope of in-place host updates
- **Stop** in-place Rolling OS Updates



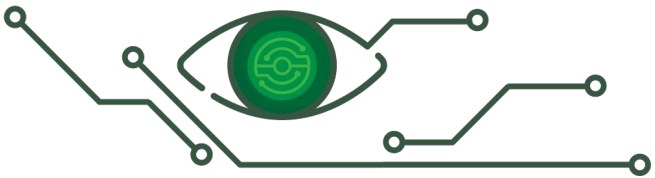
OS Updates: new major releases

- Occurs every couple of years
- Requires prep Antlir and Chef work for the new OS release
- Continuous provisioning cycles made the process smoother
  - C7->8 migration: 15 months 😰 C8->9 migration: 3 months 🥳
- Controlled Rollout via incremental switchover and baking
- Roll-back mechanisms always in place



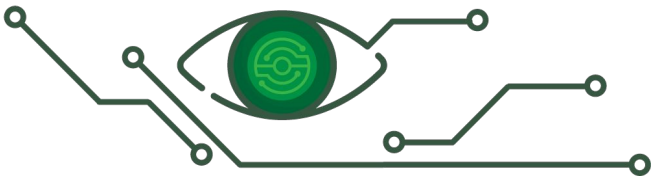
OS Updates: Challenges

- Reaching the 45 days goal required much coordination 🤝🏽
  - Make all stakeholders aware, ensuring visibility
  - Work together on regressions
  - Gently push back requests to delay the cycle, proposing alternatives
- Things don't always go as planned 🥺
  - Buffers are critical
  - Incidents stop upgrades
  - Manual maintenances hinder scheduling



OS Updates: Achievements

- MclassicA runs on 97% all Meta's fleet
- All hosts reimaged every 45 days
- CI/CD tested and released OS images
- Migrate almost all C8 servers to C9 via automated codemods
- Planning automated CentOS 10 rollout, at Meta Scale



Q&A