Showcase - Mender, an end-to-end OTA solution

Josef Holzmayr
Yocto Project Ambassador
& Head of Developer Relations for Mender.io

Yocto Project Summit, 2022.05

Deploy Software Updates for Linux Devices

About me

Josef Holzmayr

Head of Developer Relations at mender.io

Yocto Project Ambassador
OpenEmbedded Social Media Manager
Gitpod.io Community Hero

Guilty of most sins in this presentation myself

Contact Me

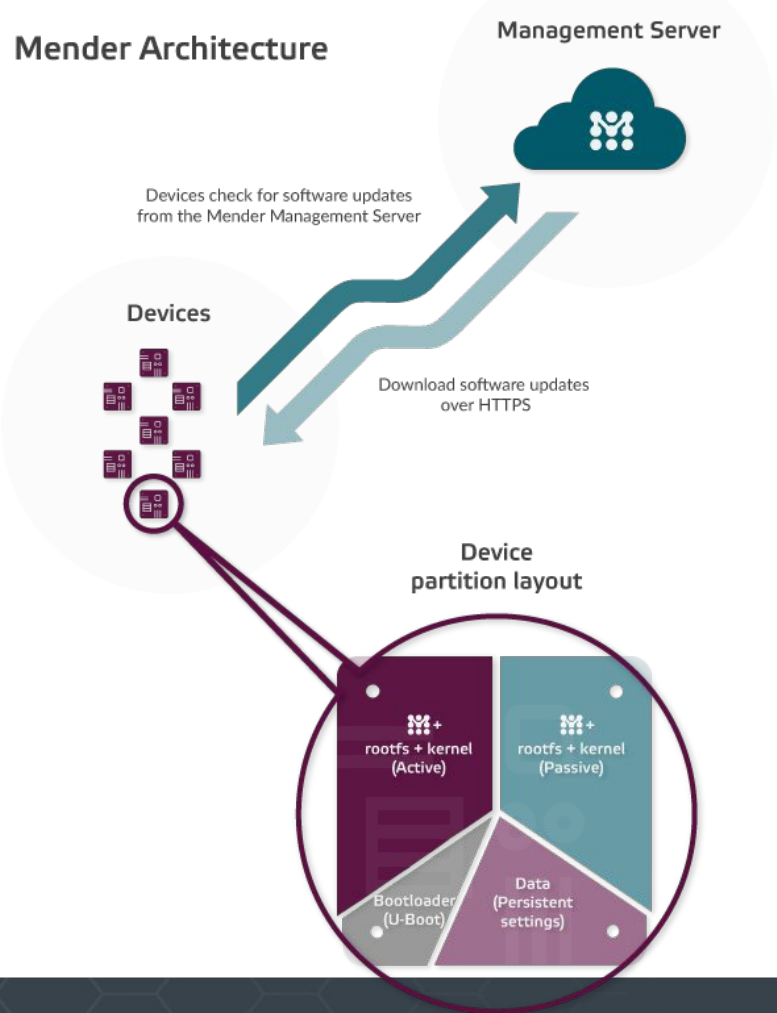✉ josef.holzmayr@northern.tech

🐦 @TheYoctoJester
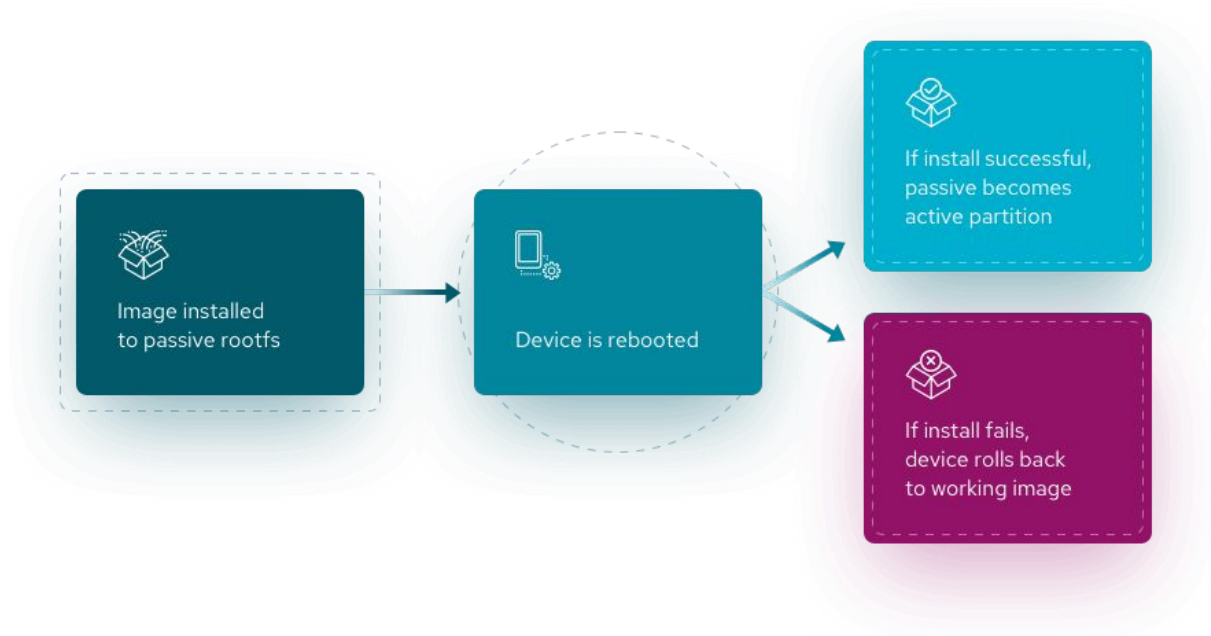
OTA requirements checklist

✓ Failsafe so it never renders device unusable ("bricked"), even when losing power

✓ Capable of atomic updates to avoid half-done installations

✓ Do integrity verification to avoid corruption of updates

✓ Able to do code signing of image updates to ensure control over updates

Mender Architecture

Management Server

Devices check for software updates
from the Mender Management Server

Devices

Download software updates
over HTTPS

Device
partition layout

rootfs + kernel
(Active)

rootfs + kernel
(Passive)

Bootloader
(U-Boot)

Data
(Persistent
settings)

A/B system updates – flow

Image installed to passive rootfs

Device is rebooted

If install successful, passive becomes active partition

If install fails, device rolls back to working image

A/B system updates – partition persistence

Device

Bootloader

OS A
(active)

OS B
(inactive)

Persistent data

OTA requirements checklist

✓ Failsafe so it never renders device unusable ("bricked"), even when losing power
✓ Capable of atomic updates to avoid half-done installations
✓ Do integrity verification to avoid corruption of updates
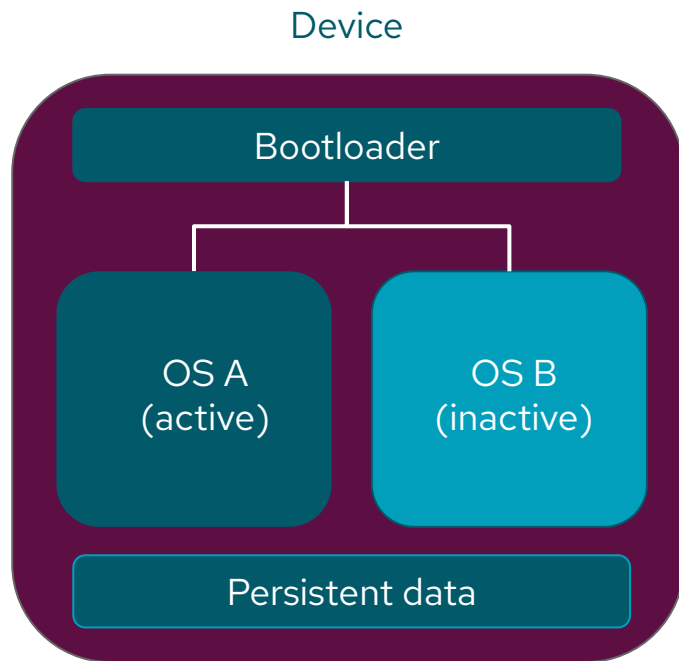✓ Able to do code signing of image updates to ensure control over updates

What artifacts do we need?

- Full Images
  - partition layout
  - bootloader
  - payload system / application
  - initial persistent data
    → `.sdimg`, `.uefiimg`

- System images
  - only payload system
  - doesn't change persistent data
    → `.mender`

[1] https://docs.mender.io/overview/artifact

What goes into a Yocto build, from where

- Machine configuration
  - comes from the BSP and triggers everything necessary to make a specific set of hardware work

- Distro configuration
  - defines the API of the linux system

- Image recipe
  - declares the actual application payload that you ship

- Local configuration
  - "transient", per build respectively build setup

How does this fit together?

- Machine configuration
  This defines the identifier that the OTA pipeline uses!
  Sets what mender needs on that specific hardware
  Example:
  - MENDER_BOOT_PART_SIZE_MB = "40"

- Distro configuration
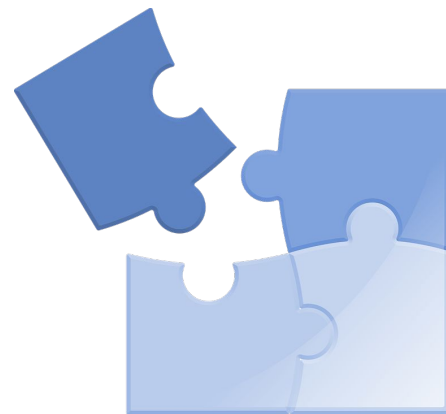 Applies the project-wide mender integration
  - INHERIT += "mender-full"

- Local configuration
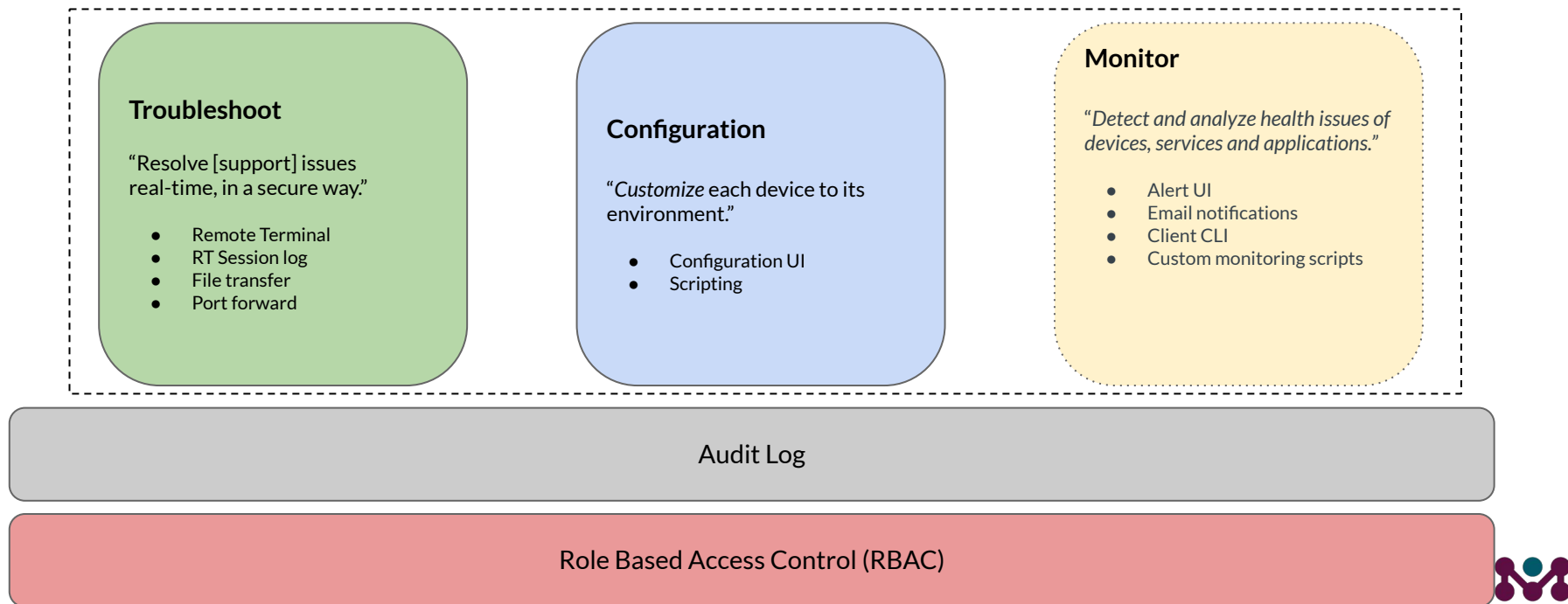 Configures how the build will be seen as an update
  - MENDER_ARTIFACT_NAME = "release-1"

Mender – Beyond OTA – Add ons

Current Device Management add-ons in Mender

Troubleshoot

"Resolve [support] issues real-time, in a secure way."

- Remote Terminal
- RT Session log
- File transfer
- Port forward

Configuration

"*Customize* each device to its environment."

- Configuration UI
- Scripting

Monitor

"*Detect and analyze health issues of devices, services and applications.*"

- Alert UI
- Email notifications
- Client CLI
- Custom monitoring scripts

Audit Log

Role Based Access Control (RBAC)

Demo time!

Learn more

Get started now
docs.mender.io/getting-started

Join the Mender Hub community
hub.mender.io

Mender on Github:
github.com/mendersoftware/

contact@mender.io

mender.io

@mender_io

company/mender.io