Practical Experience with Linux A/B Upgrades

Leon Anavi
Konsulko Group
leon.anavi@konsulko.com
leon@anavi.org
Embedded Linux Conference 2021

Konsulko Group

- Services company specializing in Embedded Linux and Open Source Software
- Hardware/software build, design, development, and training services
- Based in San Jose, CA with an engineering presence worldwide
- http://konsulko.com/

Agenda

- Strategies and open source solutions for updating embedded Linux devices
- The Yocto Project
- Mender
- RAUC
- Integration of containers with A/B upgrades
- Conclusions

Things to Consider for Software Updates (1/2)

- Are there any limitations of the disk space?
- Are there any limitations of the network bandwidth for the data transfer?
- How do you manage applications?
- Do you need a container-based solution?
- Do you need A/B or binary delta updates?
- How to upgrade: over the air, cable, USB stick, etc?
- Is the device mission critical?

Things to Consider for Software Updates (2/2)

Konsulko Group

- What distribution and build system do you use?
- Is there Yocto/OpenEmbedded BSP for the hardware you use?
- Is software update technology compatible with the YP, OE and the BSP?
- Which Yocto Project released do you need for your product?
- How to update fleet of devices?

Common Embedded Linux Update Strategies

- A/B updates (dual redundant scheme)
- Delta updates
- Container-based updates
- Combined strategies

A/B Upgrades

Konsulko Group

- Dual A/B identical rootfs partitions
- Data partition for storing any persistent data which is left unchanged during the update process
- Typically a client application runs on the embedded device and periodically connects to a server to check for updates
- If a new software update is available, the client downloads and installs it on the other partition
- Fallback in case of update failure

Combined Strategies

- Container technology has changed the way application developers interact with the cloud and some of the good practices are nowadays applied to the development workflow for embedded devices and IoT
- Containers make applications faster to deploy, easier to update and more secure through isolation
- There are use cases on powerful embedded devices where containers are combined with A/B updates of the base custom embedded Linux distribution

Popular open source solution for updates

- **Mender**
- **RAUC**
- SWUpdate
- Swupd
- UpdateHub
- Balena
- Snap
- OSTree
- Aktualizr
- Aktualizr-lite
- QtOTA
- Torizon
- FullMetalUpdate
- Rpm-ostree (used in Project Atomic)

Build Frameworks for Embedded Linux Distro

Popular open source build systems fro custom embedded Linux distributions:

- Yocto/OpenEmbedded
- Buildroot
- PTXdist
- OpenWRT
- Other

Can I just use Debian?

- Debian is a stable full distribution with tens of thousands of packages available as binary files for installation without the need to cross-compile from source
- Numerous Debian derivatives exist for embedded devices
- Debian or Yocto Project? Which is the Best for your Embedded Linux Project? Chris Simmonds, Embedded Linux Conference Europe 2019 https://www.youtube.com/watch?v=iDllXa8SzUgr

The Yocto Project

- Open source collaborative project of the Linux foundation for creating custom Linux-based systems for embedded devices using the OpenEmbedded Build System
- OpenEmbedded Build System includes BitBake and OpenEmbedded Core
- Poky is a reference distribution of the Yocto Project provided as metadata, without binary files, to bootstrap your own distribution for embedded devices
- Bi-annual release cycle
- Long term support (LTS) release covering two-year period

The Yocto Project

Konsulko Group

Codename	Version	Release Date	Support Level
Honister	3.4	October 2021	Planning
Hardknott	3.3	April 2021	Stable
Gatesgarth	3.2	October 2020	EOL
Dunfell	**3.1**	**April 2020**	**Long Term Stable**
Zeus	3.0	October 2019	EOL
Warrior	2.7	April 2019	EOL
Thud	2.6	November 2018	EOL

Yocto Override Syntax Change

- In release 3.4 Honister (scheduled for October 2021), the Yocto Project override syntax changes the **:** character replacing the use of **_** previously, for example:

```
IMAGE_INSTALL:append = " docker-ce"
```

- To help with migration of layers OE-Core provides a script:

```
<oe-core>/scripts/contrib/convert-overrides.py <layerdir>
```

- For details: http://docs.yoctoproject.org/next/migration-guides/migration-3.4.html#release-3-4-honister

Mender

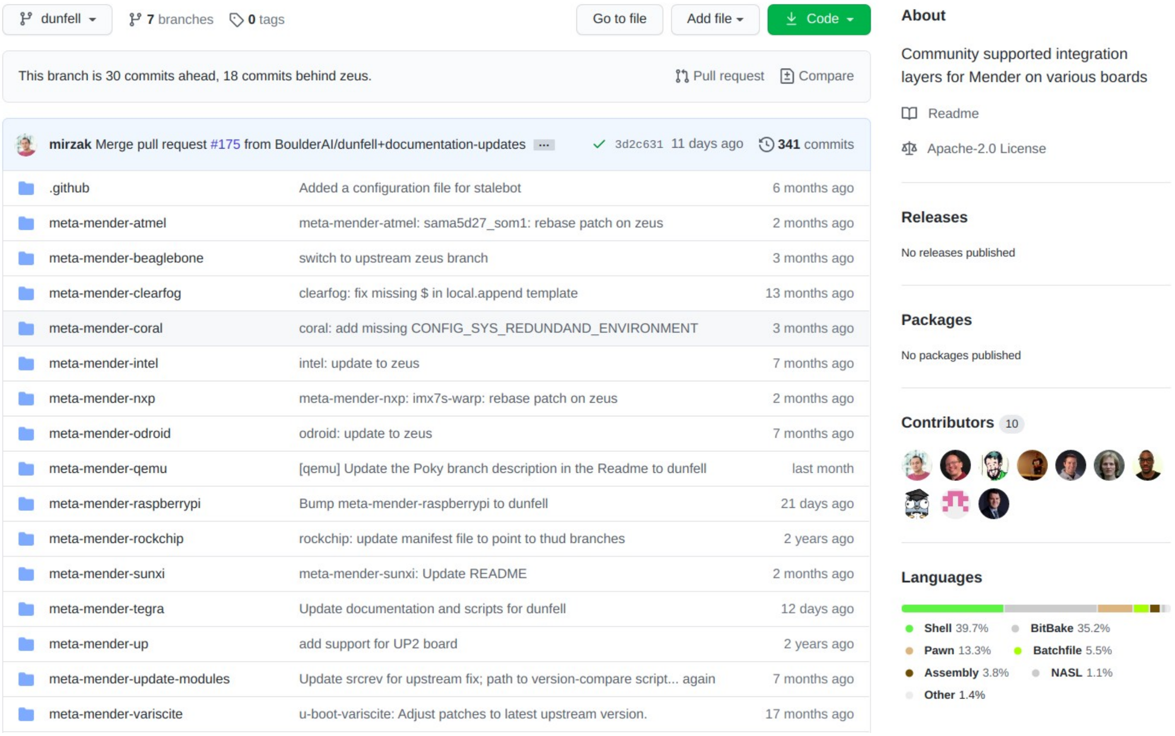
- Available as a free open source or paid commercial and enterprise plans
- A/B update scheme for open source and all plans as well as delta updates for professional and enterprise plans
- Back-end services (Hosted Mender)
- Written in Go, Python, JavaScript
- Yocto/OE integration through meta-mender and extra BSP layers: https://github.com/mendersoftware/meta-mender
- Source code in GitHub under Apache 2.0

Mender Supported Devices

The following hardware platforms and development boards are supported:

- Raspberry Pi
- BeagleBone
- Intel x86-64
- Rockchip
- Allwinner
- NXP
- And more in: https://github.com/mendersoftware/meta-mender-community

meta-mender-community



Mender

Mender A/B updates supports two client modes:

- Managed (default) - client running as a daemon polls the server for updates
- Standalone - updates are triggered locally which is suitable for physical media or any network update in pull mode

```
SYSTEMD_AUTO_ENABLE_pn-mender = "disable"
```

```
$ cd tmp/deploy/images/raspberrypi4
$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

```
$ mender -install http://example.com:8000/core-image-base-raspberrypi4.mender
```

Mender Data Partition

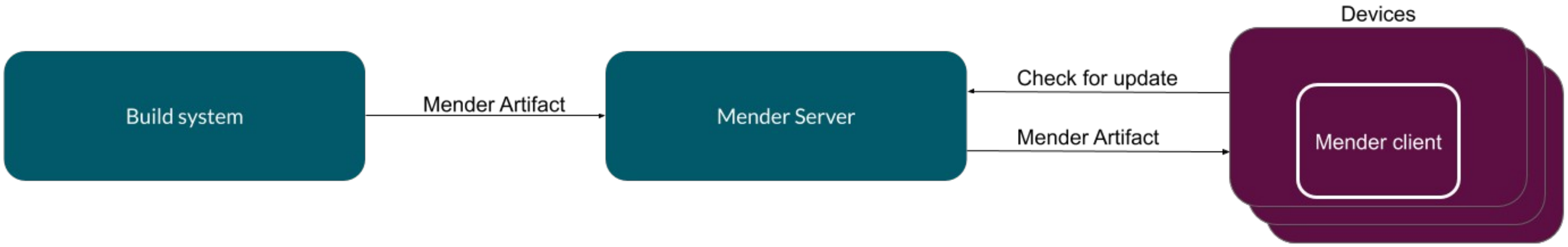
- Mender creates a **/data** partition to store persistent data, preserved during Mender updates
- the Mender client on the embedded devices uses **/data/mender** to preserve data and state across updates
- Variable **MENDER_DATA_PART_SIZE_MB** configures the size of the **/data** partition. By default it is 128 MB. If enabled, mender feature **mender-growfs-data** which relies on **systemd-growfs** tries to resize on first boot with the remaining free space
- It is possible to create an image for the data partition in advance with bitbake:

```
IMAGE_FSTYPES_append = " dataimg"
```

Mender

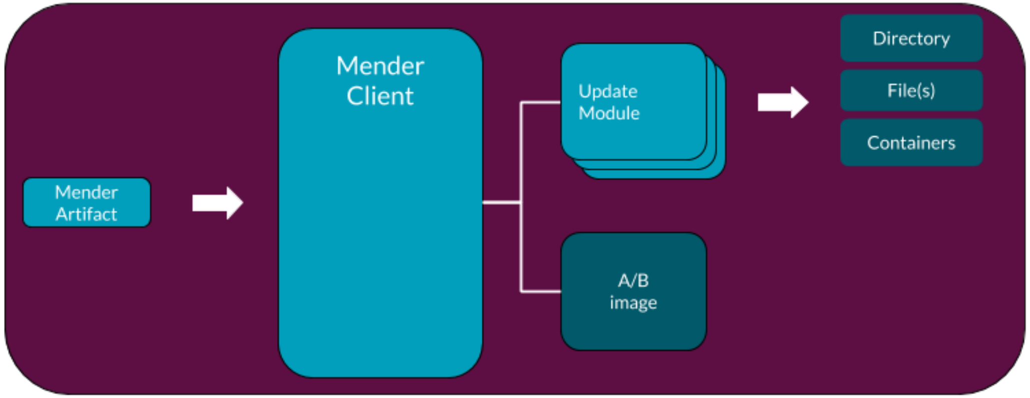
Steps to install Mender A/B update on embedded Device:

- Apply update
- Reboot
- On the first boot after a successful update, the Mender client will commit the update.



Mender Single File Artifact

- Deployment of a single file, directory or even a container image is possible through “Application updates”



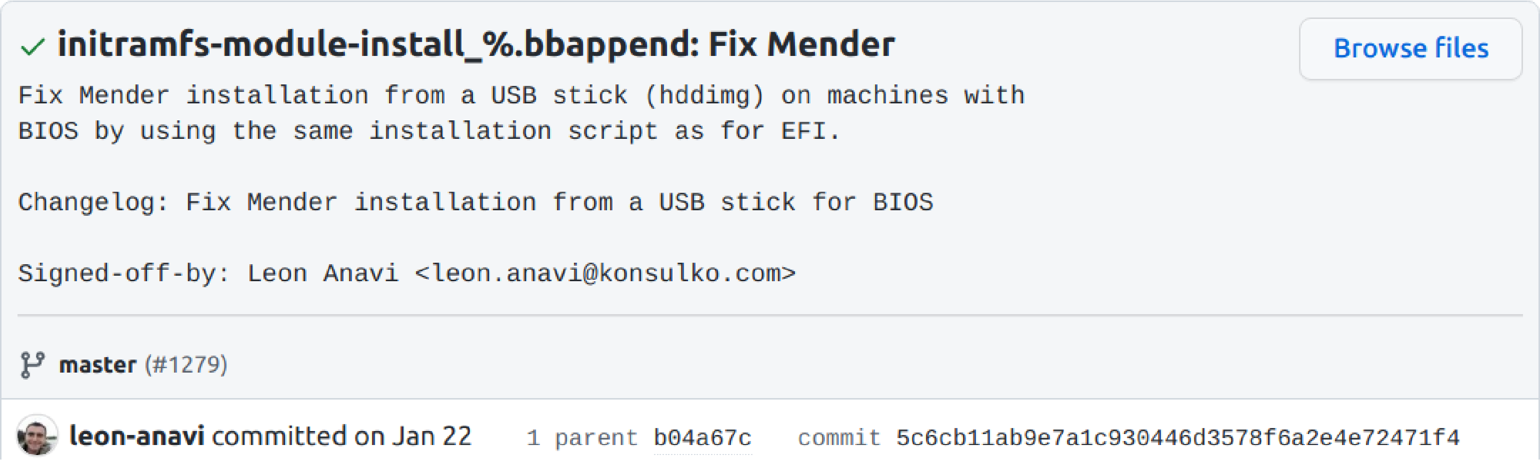
Mender add-ons

Mender supports several add-ons:

- **Remote Terminal** - interactive shell sessions with full terminal emulation
- **File Transfer** - upload and download files to and from a device
- **Port Forward** - forward any local port to a port on a device without opening ports on the device
- **Configure** - apply configuration to your devices through a uniform interface

Mender with x86-64 support

- Mender added support for x86-64 machines through GRUB in 2018
- Initial installation of the distribution is most commonly done using a live image on a USB stick


initramfs-module-install_%.bbappend: Fix Mender

Browse files

Fix Mender installation from a USB stick (hddimg) on machines with
BIOS by using the same installation script as for EFI.

Changelog: Fix Mender installation from a USB stick for BIOS

Signed-off-by: Leon Anavi <leon.anavi@konsulko.com>

master (#1279)

leon-anavi committed on Jan 22   1 parent b04a67c   commit 5c6cb11ab9e7a1c930446d3578f6a2e4e72471f4

RAUC

- A lightweight update client that runs on an Embedded Linux device and reliably controls the procedure of updating the device with a new firmware revision
- Supports multiple update scenarios
- Provides tool for the build system to create, inspect and modify update bundles
- Uses X.509 cryptography to sign update bundles
- Compatible with the Yocto Project, PTXdist and Buildroot

RAUC Licenses

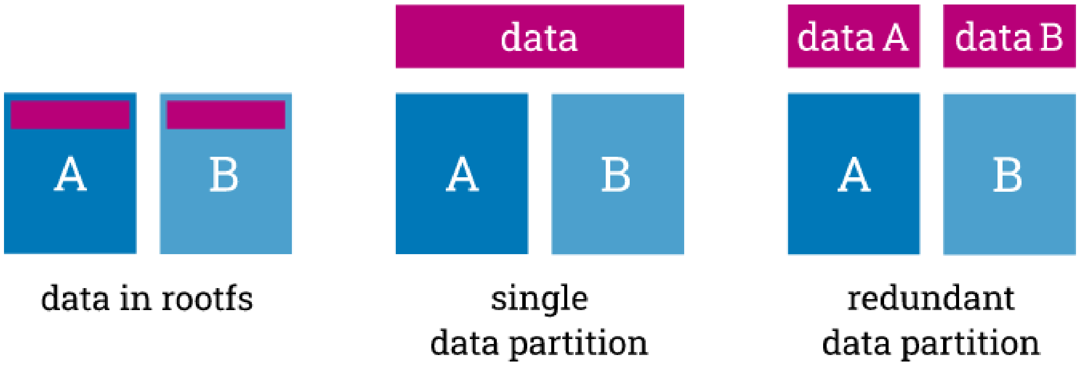
- RAUC – LGPLv2.1
  https://github.com/rauc/rauc
- meta-rauc - MIT
  https://github.com/rauc/meta-rauc
- rauc-hawkbit – LGPLv2.1
  https://github.com/rauc/rauc-hawkbit
- rauc-hawkbit-updater – LGPLv2.1
  https://github.com/rauc/rauc-hawkbit-updater

RAUC Integration Steps

- Select an appropriate bootloader
- Enable **SquashFS** in the Linux kernel configurations
- **ext4** root file system (RAUC does not have an ext2 / ext3 file type)
- Create specific partitions that matches the RAUC slots
- Configure Bootloader environment and create a script to switch RAUC slots
- Create a certificate and a keyring to RAUC’s **system.conf**

RAUC Data Partition

- Supports single and redundant data partitions
- For redundant data partitions the active rootfs slot has to mount the correct data partition dynamically, for example with a udev rule



meta-rauc-community

- Yocto/OE layer with examples how to integrate RAUC on various machines
- Started in 2020
- Moved to the RAUC organization in GitHub in 2021
- Currently supports Raspberry Pi through **meta-raspberrypi** and Sunxi (Allwinner) devices through **meta-sunxi**
- https://github.com/rauc/meta-rauc-community/

Contributions are always welcome as GitHub pull requests!

RAUC Example with Raspberry Pi 4

Konsulko Group

- Integration layer (branch **Dunfell**): https://github.com/rauc/meta-rauc-community/tree/master/meta-rauc-raspberrypi
- Add layers to **bblayers.conf** and add the following configuration to **local.conf**:

```
MACHINE = "raspberrypi4"
DISTRO_FEATURES_append = " systemd"
VIRTUAL-RUNTIME_init_manager = "systemd"
DISTRO_FEATURES_BACKFILL_CONSIDERED = "sysvinit"
VIRTUAL-RUNTIME_initscripts = ""
IMAGE_INSTALL_append = " rauc"
IMAGE_FSTYPES="tar.bz2 ext4 wic.bz2 wic.bmap"
SDIMG_ROOTFS_TYPE="ext4"
ENABLE_UART = "1"
RPI_USE_U_BOOT = "1"
PREFERRED_PROVIDER_virtual/bootloader = "u-boot"
WKS_FILE = "sdimage-dual-raspberrypi.wks.in"
```

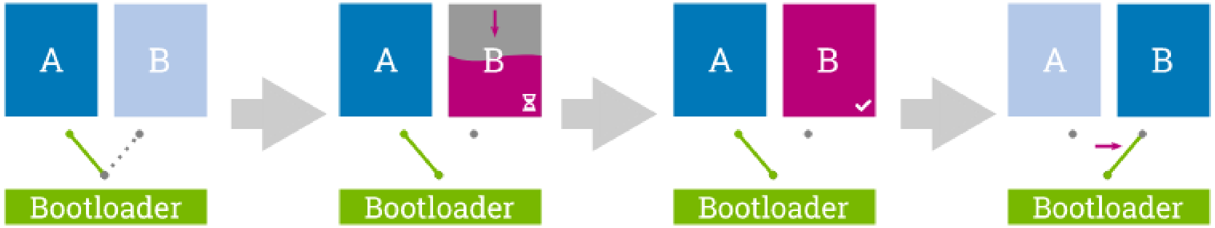
Manual RAUC Update of Raspberry Pi 4

- On the build system:

```
cd tmp/deploy/images/raspberrypi4/
python3 -m http.server
```

- On the embedded device:

```
wget http://example.com:8000/update-bundle-raspberrypi4.raucb -P /tmp
rauc install /tmp/update-bundle-raspberrypi4.raucb
reboot
```



Read-only Root Filesystem

Yocto and OpenEmbedded offer two options to create a read-only root filesystem:

- Thought the image’s recipe file:

```
IMAGE_FEATURES += "read-only-rootfs"
```

- Alternatively, through **local.conf**:

```
EXTRA_IMAGE_FEATURES = "read-only-rootfs"
```

- Beware, there might be packages in the image that expect the root filesystem to be writable and might not function properly. A solution is to move these files and directories to the data partition.

Combined Strategies with Containers

- Yocto/OE layer **meta-virtualization** provides support for building Xen, KVM, Libvirt, docker and associated packages necessary for constructing OE-based virtualized solutions
- **virtualization** has to be added to the **DISTRO_FEATURES**:

```
DISTRO_FEATURES_append = " virtualization"
```

- For example adding Docker to the embedded Linux distribution is easy:

```
IMAGE_INSTALL_append = " docker-ce"
```

- There are use cases on powerful embedded devices where contains are combined with A/B updates of the base Linux distribution built with Yocto/OE

Conclusion

- There are numerous things to consider when implementing an upgrade mechanism for an embedded Linux device
- Use open source software for upgrade mechanism instead of another proprietary homegrown solution
- Mender and RAUC are powerful solutions for A/B upgrades with excellent Yocto/OpenEmbedded integration as well as for alternative build frameworks
- Combined strategies for A/B upgrades with containers for applications are increasingly popular nowadays
- Real-world implementations of A/B upgrades very often require a data partition for storing any persistent data which is left unchanged during the update process

Thank You!

Useful links:

- https://www.yoctoproject.org/
- https://mender.io/
- https://rauc.io/
- https://git.yoctoproject.org/cgit/cgit.cgi/meta-virtualization/
- https://www.konsulko.com/building-platforms-with-secure-over-the-air-updating/
- https://www.konsulko.com/how-mender-works/
- https://www.konsulko.com/getting-started-with-rauc-on-raspberry-pi-2/