



From code to connected device: Building and maintaining embedded Linux distributions

Josef Holzmayr

Head of Developer Relations, Mender.io

IT-S NOW, 06.06.2024

Hello, my name is Josef.



**I am a recovering
embedded developer.**



Some technical details about me



Street credibility

Yocto Project Community Manager & Ambassador

OpenEmbedded Social Media Manager

Kernel contributor (yup, really!)

Fame

 <https://www.linkedin.com/in/josef-holzmayr>

 josef.holzmayr@northern.tech

 <https://fosstodon.org/@theyoctojester>



Important note concerning this presentation!

Every form of interaction will be rewarded
... until I run out of chocolate.

Some ideas:

- Good: Tell me what you like.
- Better: Tell me what you don't like.
- Best: Tell me where I am wrong.
- Helpful: Ask for a clarification.
- Practical: Ask for chocolate.



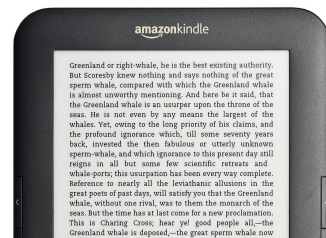
Image attribution: [User:-donald- - Wikimedia Commons](#)

I will not talk about these things:

- **When you should use Linux, or not**
- **What an embedded or connected device is**
- **Why you need OTA updates**
- ...

**All of this has been discussed many times already,
and will be many more.**

Embedded Linux is already ubiquitous.



https://wiki.yoctoproject.org/wiki/Project_Users



We now have about 25 minutes
to build and maintain a full Linux
distribution.

Disclaimer: therefore the shown command snippets are just **core concepts**



Things that you really want

License tracing

→ Shipping a Linux based device means that you are “conveying” copies as stated by most open source licenses, so you need to abide the rules.

SBOM generation

→ Keeping logs about what you shipped is a crucial first step for vulnerability management! Oh, and required by regulations in many cases too.

CVE checking

→ Don't hand out software which includes already *KNOWN* problems!

The toolbox



Step 1: Build and run

Raw poky 1 – build

```
$ mkdir itsnow && cd itsnow
$ git clone -b scarthgap git:// git.yoctoproject.org/poky.git
$ source poky/oe-init-build-env
$ time bitbake core-image-minimal
```

(you might want to watch `top` or `htop` now 😊)

<snip/>

```
real    61m12.114s
user    0m16.621s
sys     0m4.017s
$
```

Raw poky 2 - run

```
$ runqemu nographic slirp  
<boot, scroll, boot, scroll/>
```

```
Poky (Yocto Project Reference Distro) 5.0.1 qemu86-64 /dev/ttyS0
```

```
qemu86-64 login: root
```

```
WARNING: Poky is a reference Yocto Project distribution that should be used  
for  
testing and development purposes only. It is recommended that you create  
your  
own distribution for production use.
```

```
root@qemu86-64:~#
```

Step 2: Customize

Example: `systemd` & `bc`

Most things can be customized through variables, such as the init manager choice, the selection of packages to be installed or the filesystem of the resulting system image.

```
$ cat <<EOF >> conf/local.conf
INIT_MANAGER = "systemd"
IMAGE_INSTALL:append = " bc"
IMAGE_FSTYPES:append = " tar.bz2"
EOF
```

The key concept here is called metadata.

It defines all aspects of the build process, such as

- Source code URLs and revisions
- Configuration files and fragments
- Building, packaging and deployment stages
- ...

Example: layer, recipe and image

```
$ bitbake-layers create-layer ../meta-itsnow
$ bitbake-layers add-layer ../meta-itsnow
$ mkdir -p ../meta-itsnow/recipes-itsnow/images
$ cp ../poky/meta/recipes-core/images/core-image-minimal.bb \
    ../meta-itsnow/recipes-itsnow/images/core-image-itsnow.bb

$ bitbake core-image-itsnow
$ runqemu nographic slirp
```

Example: build for Raspberry Pi 4

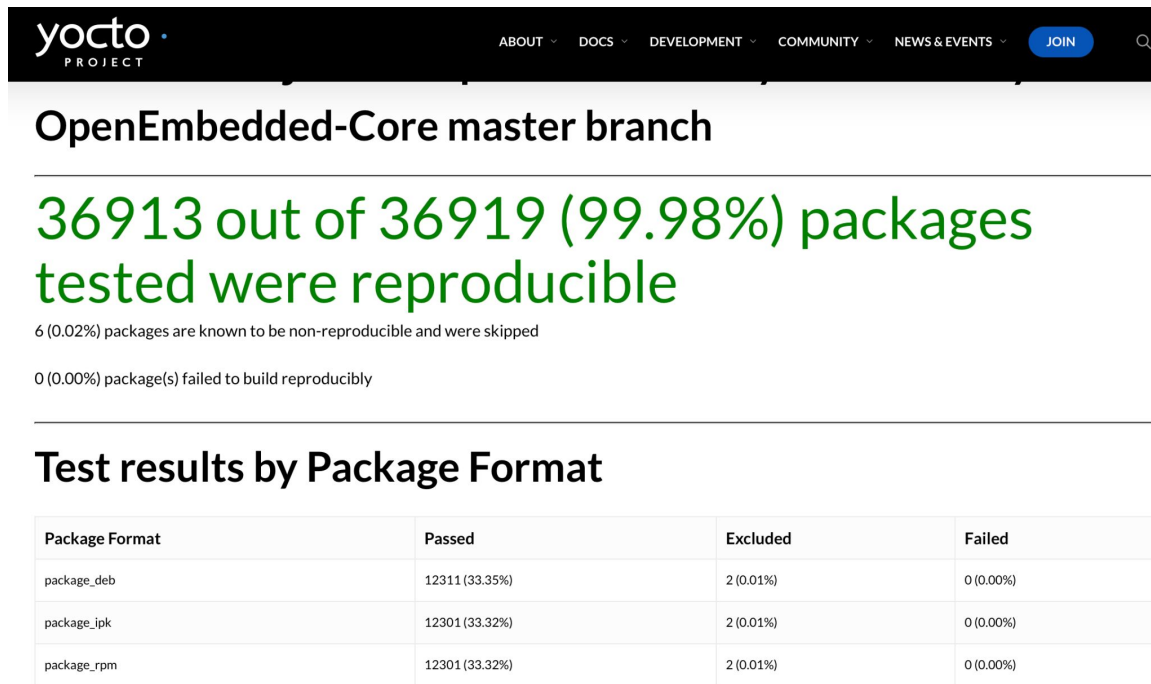
Board support packages for Yocto are usually provided as layers.

```
$ git clone -b scarthgap https://github.com/agherzan/meta-raspberrypi \  
  ../meta-raspberrypi  
$ bitbake-layers add-layer ../meta-raspberrypi  
$ cat <<EOF >> conf/local.conf  
  MACHINE = "raspberrypi4"  
  EOF
```

All of the relevant metadata should be under version control.

- **References and revisions of third party layers**
- **Custom layers including recipes**
- **MACHINE and DISTRO selection**
- **Eventual local.conf entries**

You get: reproducibility!



The screenshot shows the Yocto Project website with a navigation bar at the top. The main heading is "OpenEmbedded-Core master branch". Below it, a large green text block states "36913 out of 36919 (99.98%) packages tested were reproducible". Underneath, it notes "6 (0.02%) packages are known to be non-reproducible and were skipped" and "0 (0.00%) package(s) failed to build reproducibly". A section titled "Test results by Package Format" contains a table with the following data:

Package Format	Passed	Excluded	Failed
package_deb	12311 (33.35%)	2 (0.01%)	0 (0.00%)
package_ipk	12301 (33.32%)	2 (0.01%)	0 (0.00%)
package_rpm	12301 (33.32%)	2 (0.01%)	0 (0.00%)

Source: <https://www.yoctoproject.org/reproducible-build-results/> (as of 2024-06-02)

Step 3: Maintain

Yocto Magic 1: license compliance

Include the license texts and copyright notices, archive the sources used in the build.

```
$ cat <<EOF >> conf/local.conf
COPY_LIC_MANIFEST = "1"
COPY_LIC_DIRS = "1"
LICENSE_CREATE_PACKAGE = "1"

INHERIT += "archiver"
ARCHIVER_MODE[src] = "original"
EOF
```

Yocto Magic 2: SBOM generation

Create an SPDX-style SBOM, including descriptions of the individual files used in the build.

```
$ cat <<EOF >> conf/local.conf
  INHERIT += "create-spdx"

  SPDX_INCLUDE_SOURCES = "1"
EOF
```

Yocto Magic 3: enable CVE checking

Yocto offers a built-in CVE check at build time. Note that this might require fine tuning per use case!

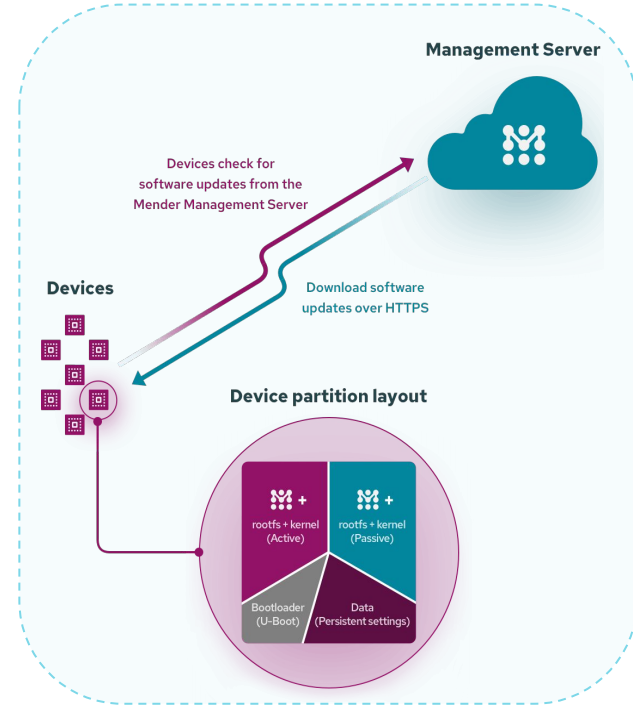
```
$ cat <<EOF >> conf/local.conf
  INHERIT += "cve-check"
EOF
```


Step 4: Deploy and manage

Mender – overview

Integrated solution

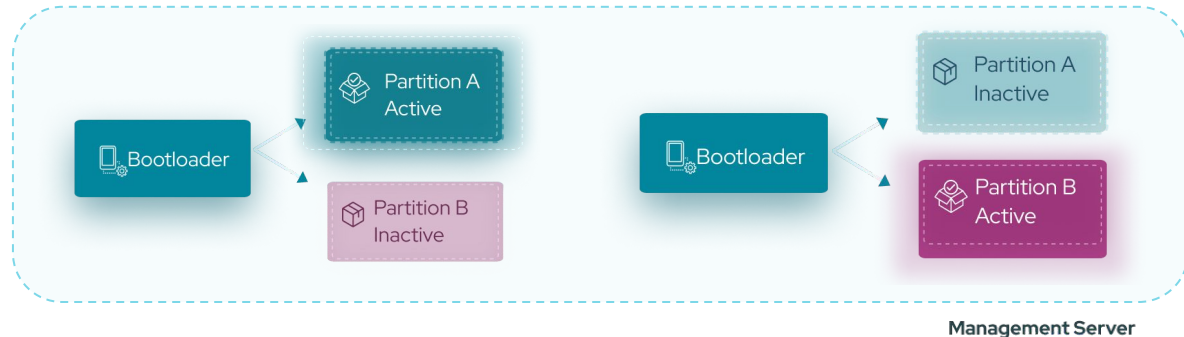
Both client- and server-side are to aligned to provide industrial-grade fleet and device lifecycle management.



Mender – on connected device

OS updates

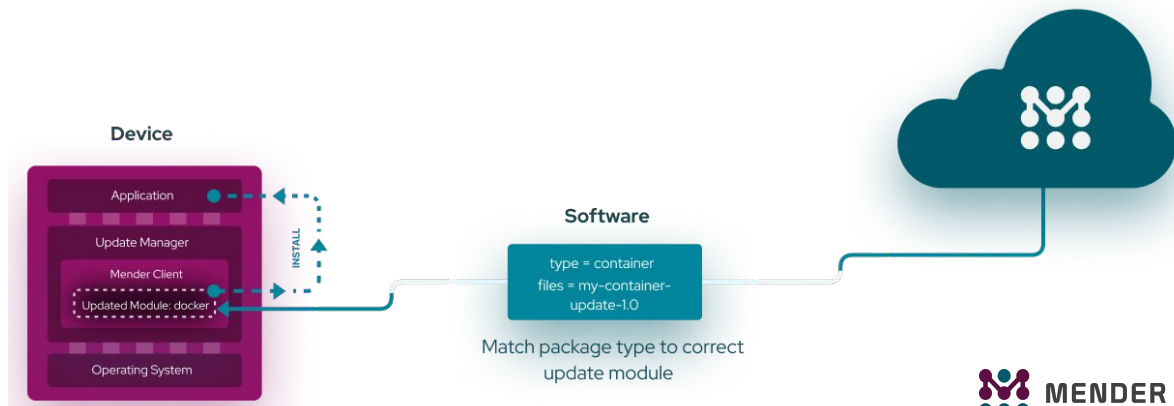
Provides full image updates with robust and failsafe mechanisms should an update fail for any reason.



Application updates

Highly customizable:

- containers
- binary applications and assets
- sub-controller firmware.



Enabling Mender

Mender supports a streamlined integration process through metadata layers.
Example board: Raspberry Pi 4

```
$ git clone https://github.com/mendersoftware/meta-mender \  
  ../meta-mender  
$ git clone https://github.com/mendersoftware/meta-mender-community \  
  ../meta-mender-community  
$ bitbake-layers add-layer ../meta-mender/meta-mender-core  
$ bitbake-layers add-layer ../meta-mender-community/meta-mender-raspberrypi  
$ cat <<EOF >> conf/local.conf  
  INHERIT += "mender-full"  
  EOF
```

Building Mender

This configures the build to generate an initial image and an artifact.

```
$ bitbake core-image-minimal
$ ls tmp/deploy/images/raspberrypi4
<snip/>
core-image-full-cmdline-raspberrypi4.mender
core-image-full-cmdline-raspberrypi4.sdimg
<snip/>
```

.sdimg: flash to SD card

.mender: upload to your Hosted Mender account to deploy

Things we skipped

- **Automated build setup: kas, git submodules,...**
- **MACHINE and DISTRO setup**
- **Release cadence & LTS**
- **Mender account and deployment process**
- **CI/CD pipeline**
- **Build time optimization and caching**
- **...**

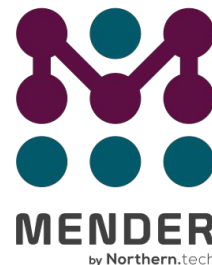
Summary

Plan for maintenance and sustainability

Creating a sustainable Linux distribution for a connected device and maintaining it is neither witchcraft nor rocket science!

The only really important rule is: understand your requirements, and then plan accordingly.

The [Yocto Project](#) and [Mender](#) are two powerful building blocks you can use and rely on.



Learn more

 contact@mender.io

 mender.io

 [@mender_io](https://twitter.com/mender_io)

 [company/northern.tech](https://www.linkedin.com/company/northern.tech)

Get started now

docs.mender.io/getting-started

Join the Mender Hub community

hub.mender.io

Mender on Github

github.com/mendersoftware

Q & A

Josef Holzmayr

Head of Developer Relations, Mender & Community
Manager, The Yocto Project

<https://github.com/TheYoctoJester/>

<https://www.linkedin.com/in/josef-holzmayr/>



MENDER
by Northern.tech

Thank you

Contact us

mender.io/contact-us