USB Updates

Challenges, Approaches and Practical Tips

Anna-Lena Marx
München, 22. Februar 2024

inovex

Anna-Lena Marx

Anna-Lena Marx

anna-lena.marx@inovex.de

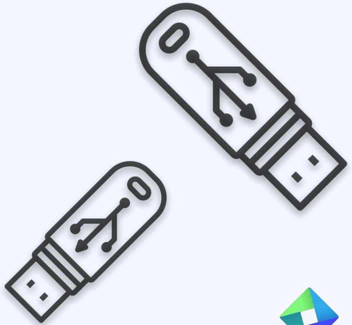
marx.engineer

Embedded Systems Developer

- with inovex since 2015
- has a master's degree in Embedded Systems
- studies Electrical Engineering as a hobby

Focus Points

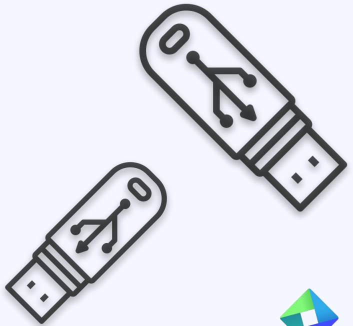
- Embedded Systems
- Yocto Linux
- Linux Kernel
- AOSP/AAOS
- IoT



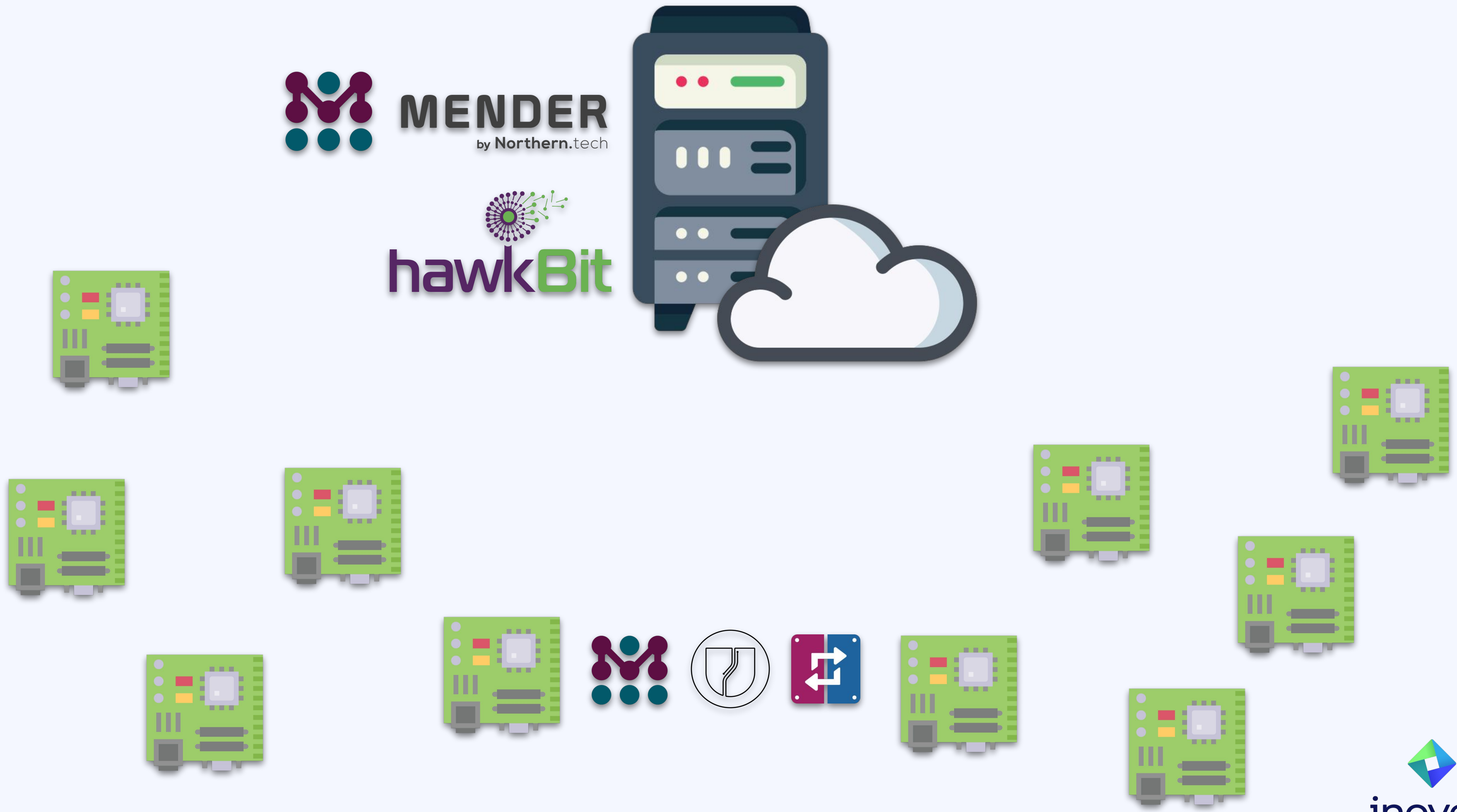
inovex

USB Updates

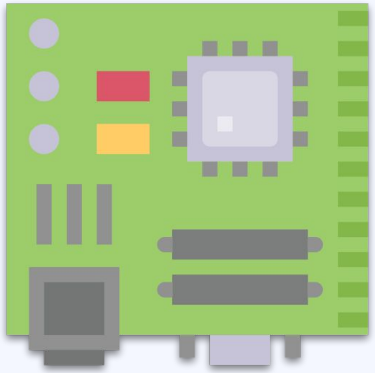
When to Consider? Why?



inovex



inovex



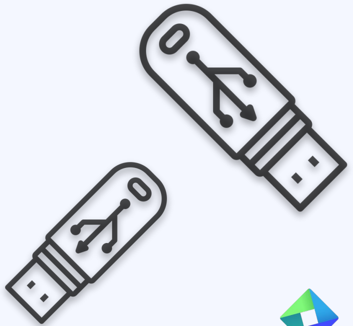
Why do USB Updates when Over-the-Air is standard?

- in early phases/start-up
  - few devices in field
  - shying away from cost and effort of maintaining the update server
  - *fast and easy* way to updates
  - at least a *low effort* way to update devices

- back-up mechanism
  - unreliable network connection

Update Mechanisms

Key Requirements and Factors



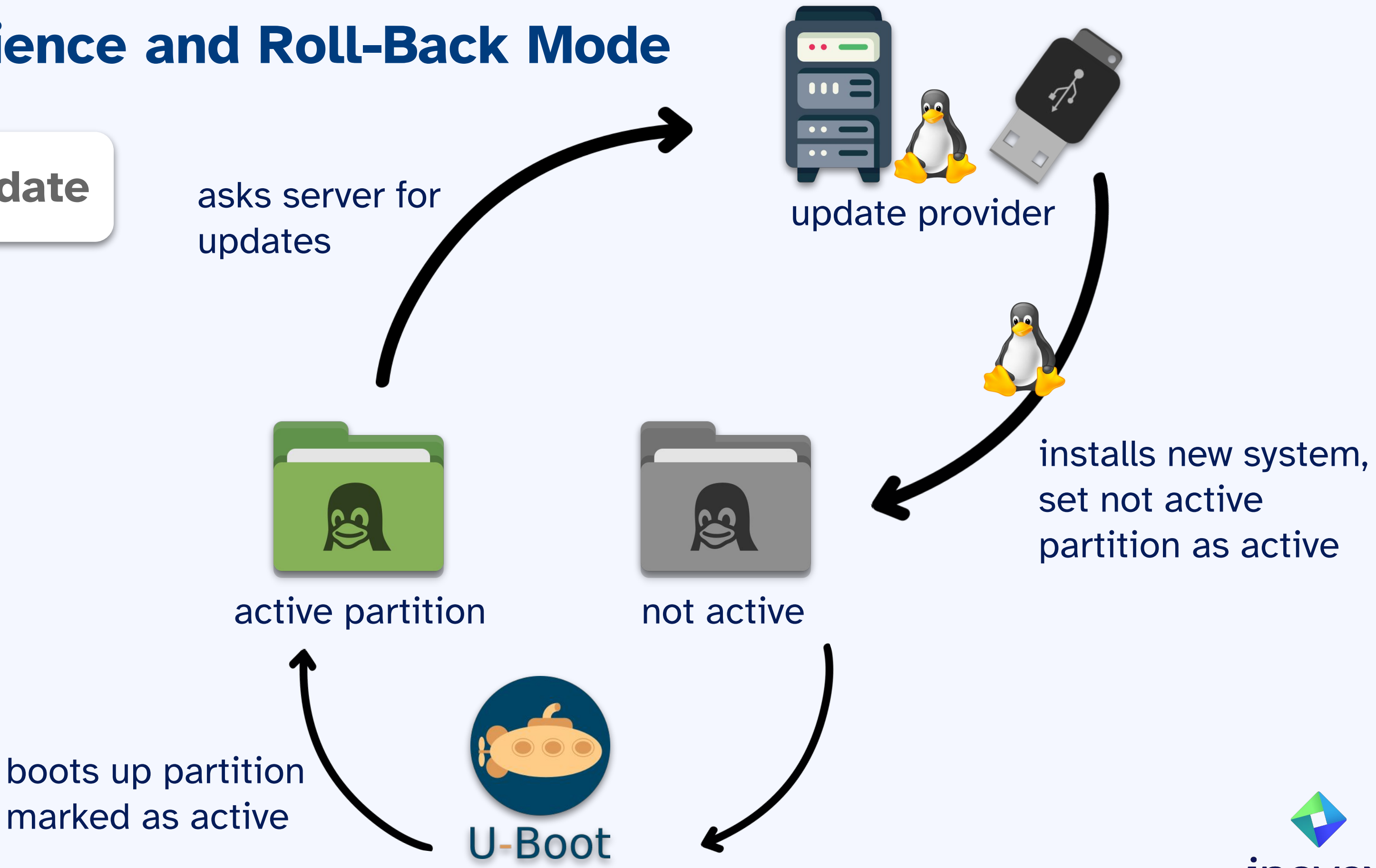
inovex

Update Mechanism - Design Considerations

- block-based vs. file based
- storage layout
- root filesystem (read-only vs. read/write)
- update data source
- granularity (update files vs. whole system)
- integration with build systems and bootloader
- fault resilience
- security (signing, encryption, ...)
- license

Fault Resilience and Roll-Back Mode

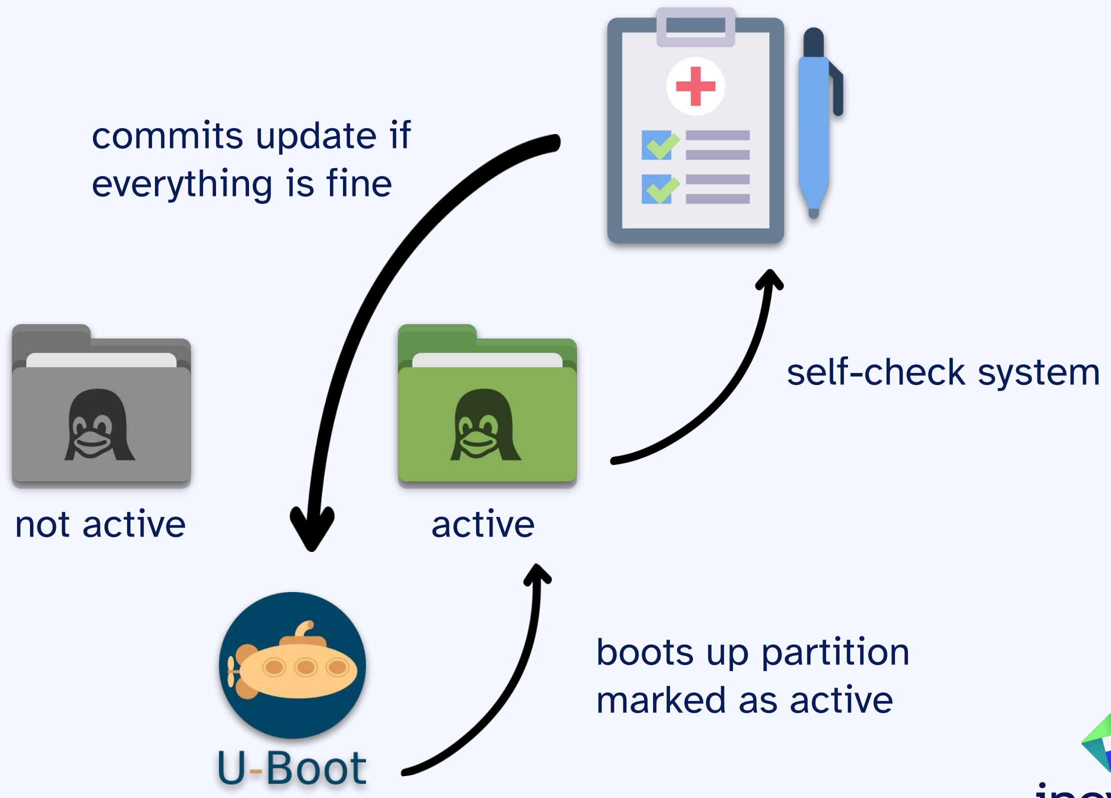
Install New Update



asks server for updates
update provider
installs new system, set not active partition as active
active partition
not active
boots up partition marked as active
U-Boot

Fault Resilience and Roll-Back Mode

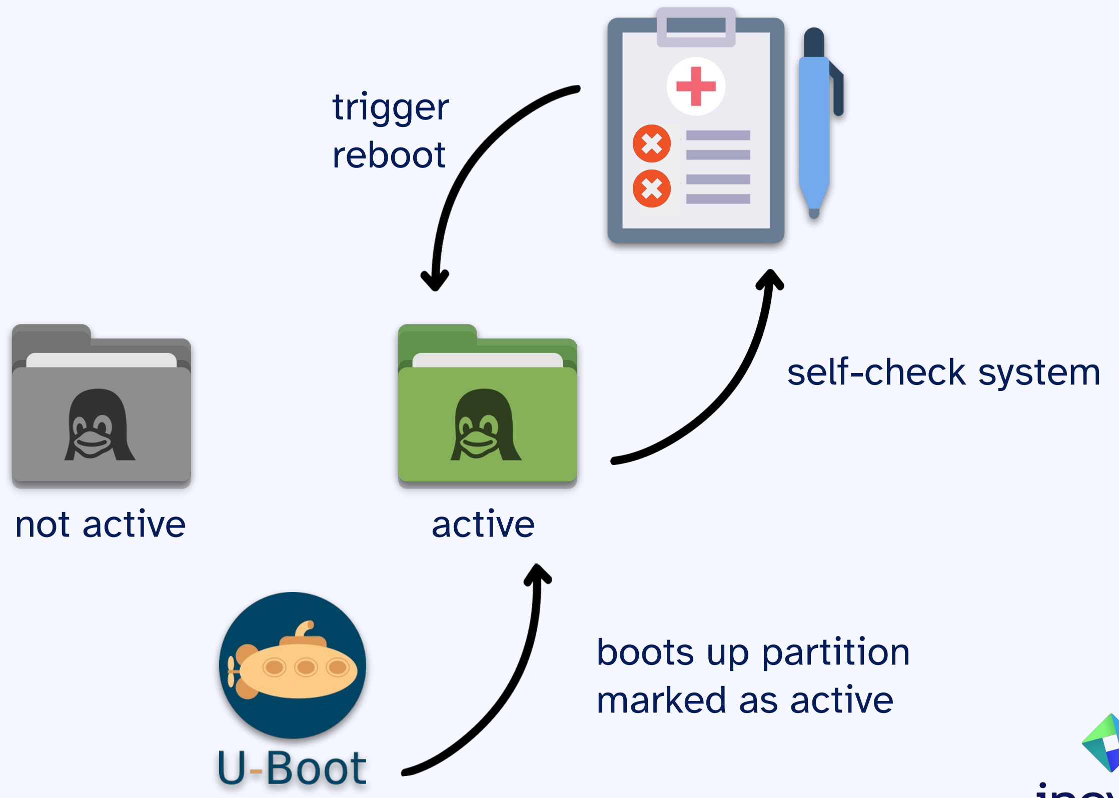
Healthy System


commits update if everything is fine
self-check system
not active
active
U-Boot
boots up partition marked as active

inovex

Fault Resilience and Roll-Back Mode

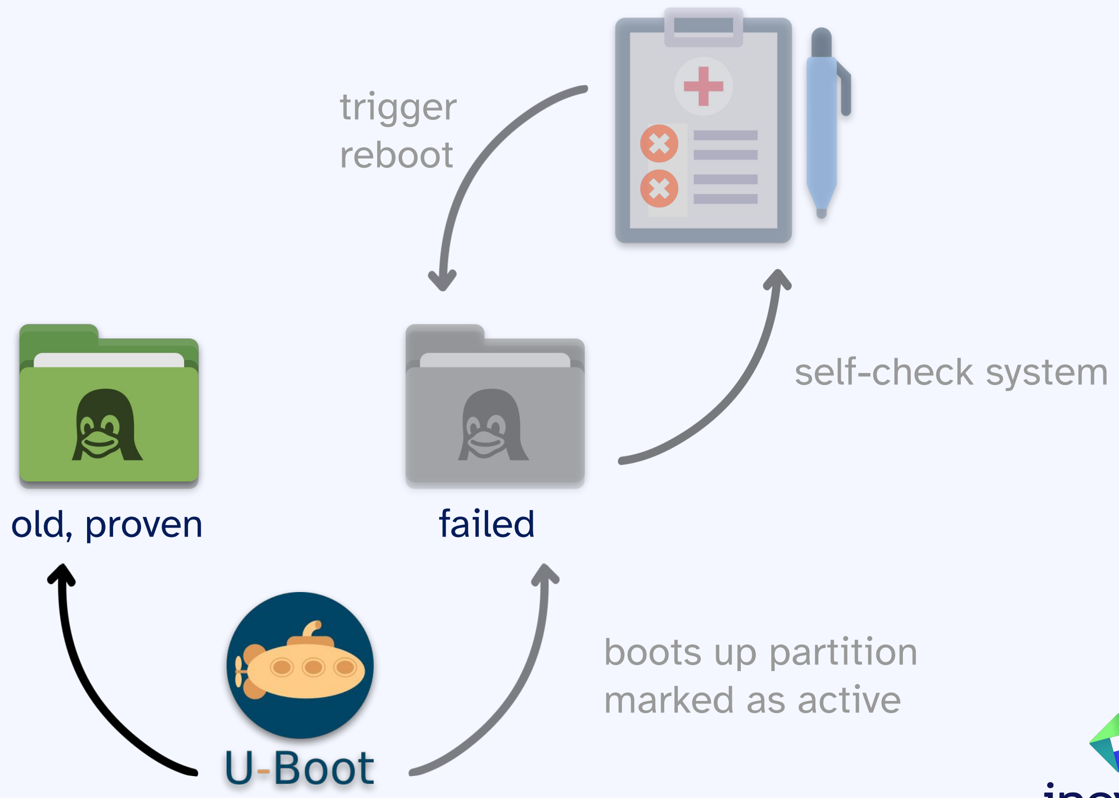
Faulty System



inovex

Fault Resilience and Roll-Back Mode

Faulty System

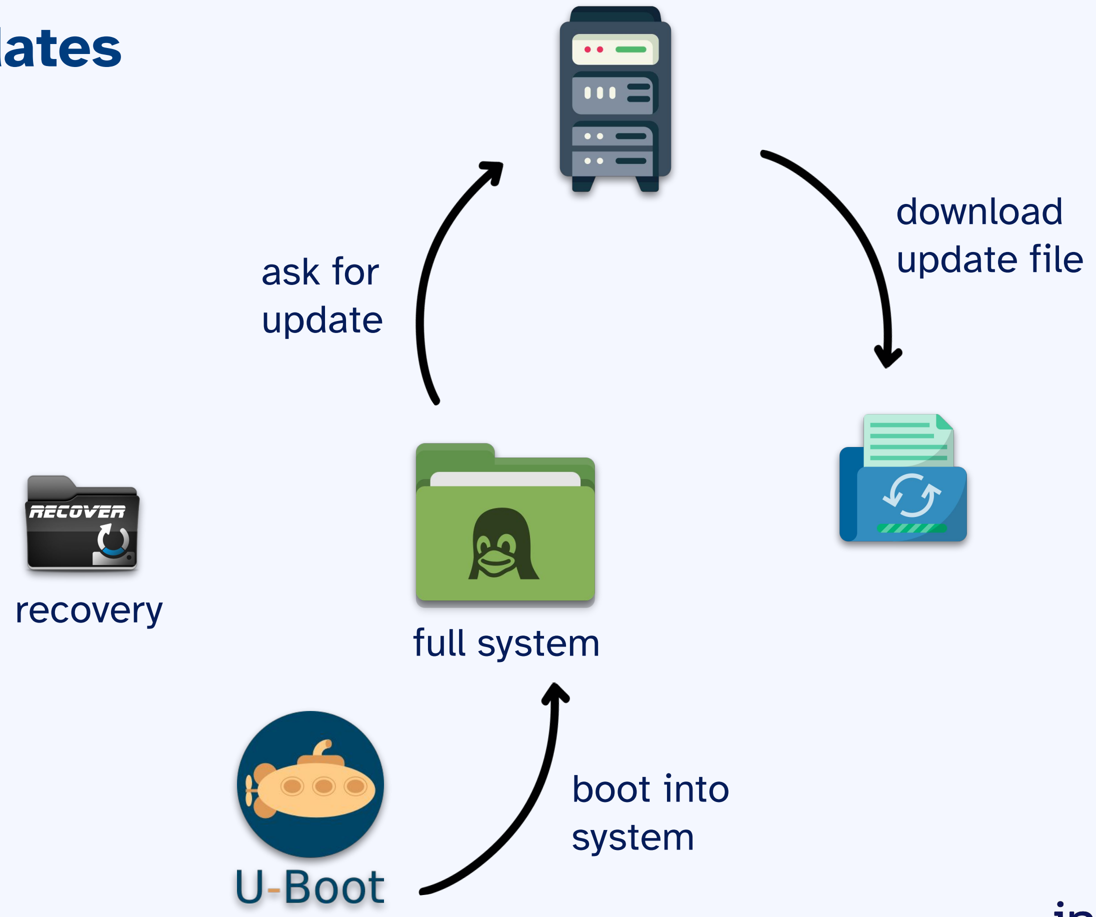


Update was not marked as ok, boot old one!

inovex

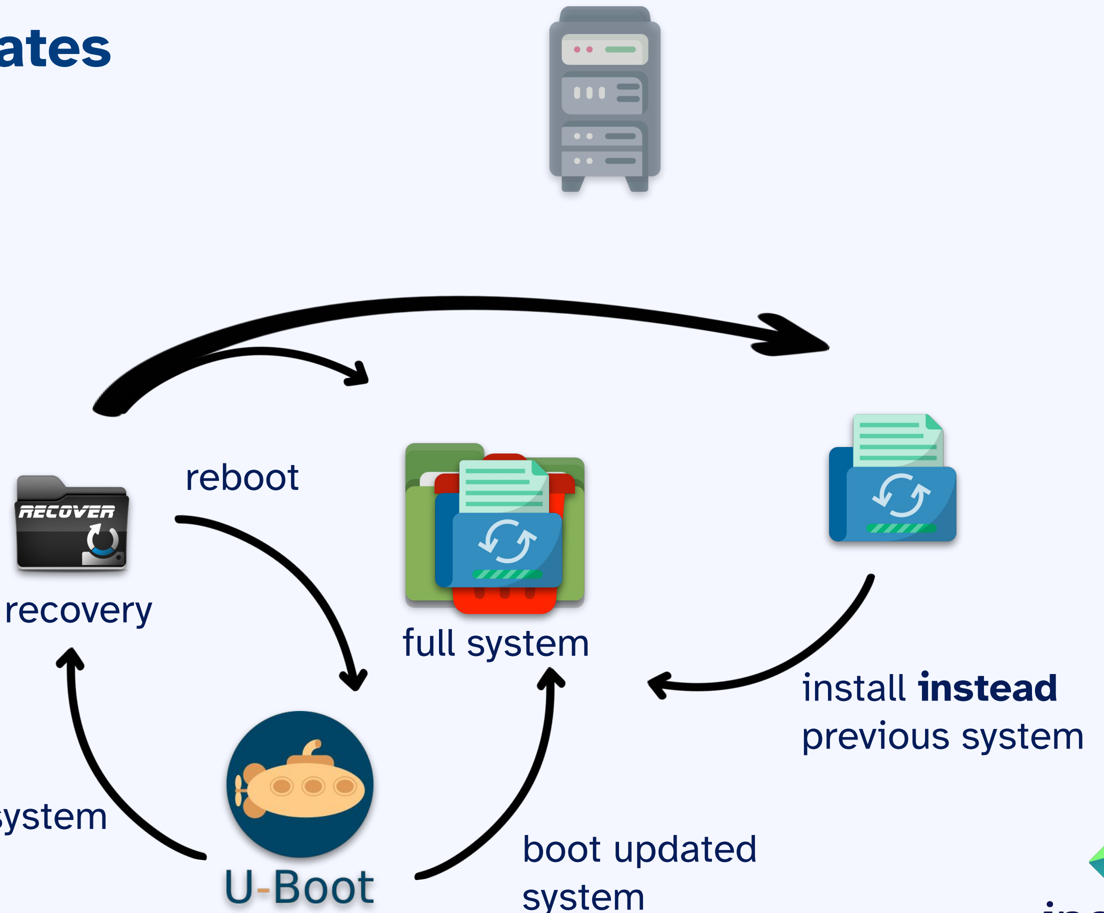
Recovery-based Updates

Download



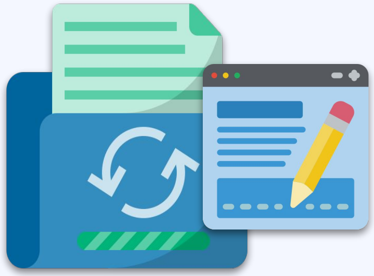
Recovery-based Updates

Install & check



inovex

Security

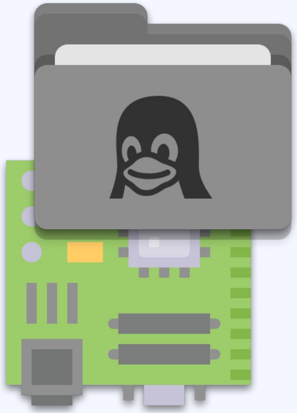


sign updates!

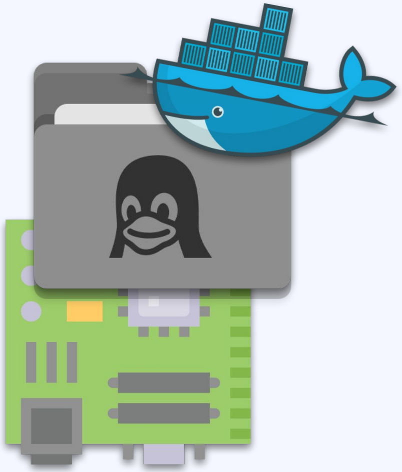
encrypt as needed

inovex

Granularity



rootfs and kernel



containers

file-based

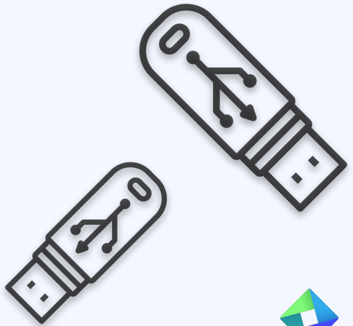
Server-Side

Are Over-the-Air Updates Planned for Later?

- decide for a client capable of both
  - will you be able to host and maintain hawkBit?
  - is a managed Mender server an option?
- already check what is needed to enable OTA
- have a plan, know what’s to do!

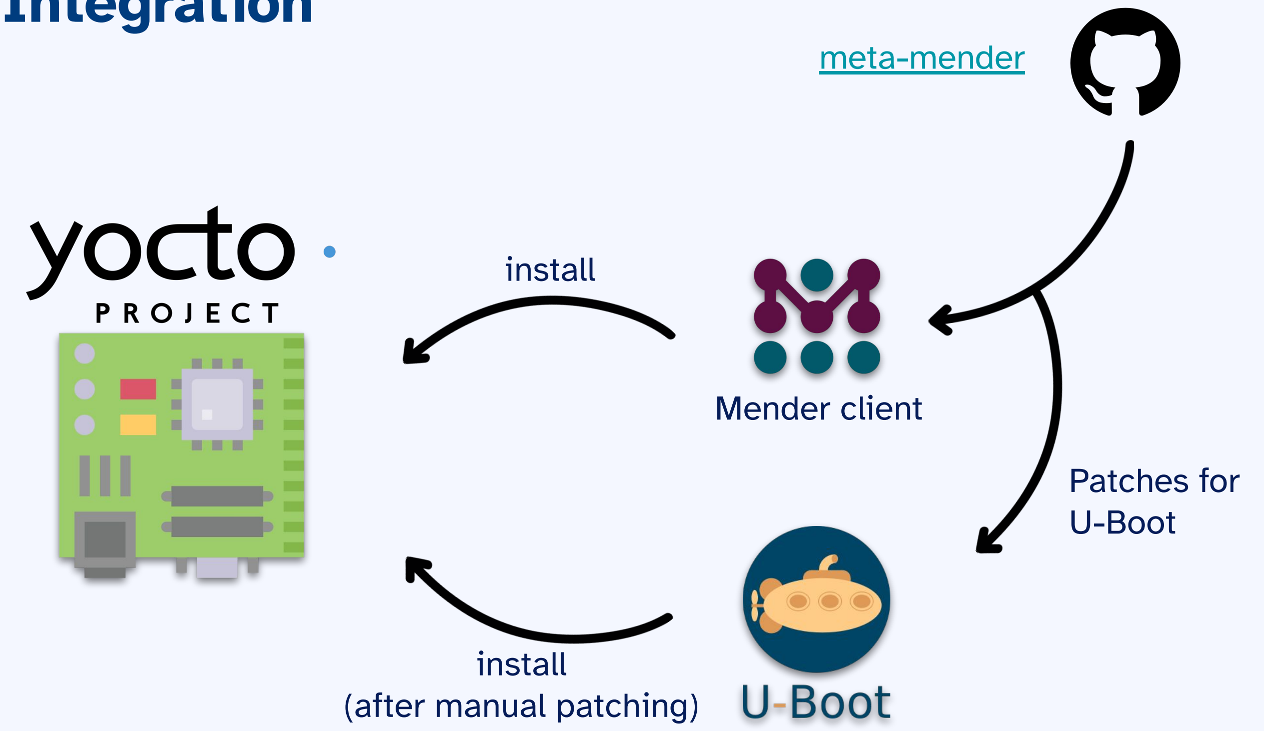
Hands on!

Updating a Yocto-based Linux device using Mender



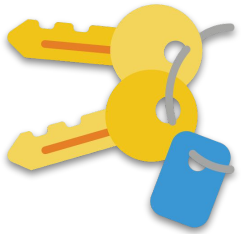
inovex

Mender Integration



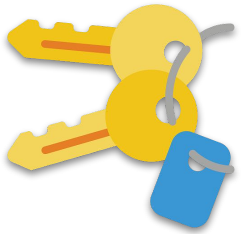
Key Generation

```
# Generate a private key and params
openssl ecparam -genkey -name prime256v1 -out private-and-params.key


# Derive a private key file
openssl ec -in private-and-params.key -out private.key


# Derive a public key file
openssl ec -in private-and-params.key -pubout -out public.key
```



meta-inovex/mender-client_%.bbappend

```
FILESEXTRAPATHS:prepend := "${THISDIR}/files:"

SRC_URI:append = " file://artifact-verify-key.pem"
```



Artifact Signing

```
# Download mender-artifact utility at
# https://docs.mender.io/downloads#mender-artifact
# or install it via the Mender APT repository
# https://docs.mender.io/downloads#install-using-the-apt-repository

# Sign the update file
mender-artifact sign \
 artifact.mender -k private.key -o artifact-signed.mender
```

install-usb-update.sh

```bash
#!/bin/bash

UPDATED="0"
SEARCH_PATH="/media"
SEARCH_ARG="${SEARCH_PATH}/*.mender"

if test -f ${SEARCH_ARG}; then
        echo "Update file found!"
        MENDER_PATH=$(realpath ${SEARCH_ARG})

        /usr/bin/mender install ${SEARCH_PATH}/${MENDER_PATH}
        STATUS=$?

        if [ ${STATUS} -eq 0 ]; then
                echo "Mender install was successful! Rebooting ..."
                UPDATED="1"
        else
                echo "Installing the update failed!"
        fi
fi

MOUNT_PATH=${SEARCH_PATH%.*}
echo ${MOUNT_PATH}
umount ${MOUNT_PATH}

if [ ${UPDATED} -eq 1 ]; then
        reboot
fi
```

Advanced

Auto-magic installation with udev and systemd

usb-update.rules - v1

```
ACTION=="add", \
SUBSYSTEMS=="usb", \
SUBSYSTEM=="block", \
ENV{ID_FS_USAGE}=="filesystem", \
RUN{program}+= "install-update.sh"
```

Starting daemons or other ***long-running processes is not allowed;*** *the forked processes, detached or not,*
will be ***unconditionally killed after the event handling has finished.***
In order to activate long-running processes from udev rules, ***provide a service unit and pull it in from an udev device using the SYSTEMD_WANTS device property.***
See [systemd.device(5)] for details.

usb-update.rules - v2

```
ACTION=="add", \
SUBSYSTEMS=="usb", \
SUBSYSTEM=="block", \
ENV{ID_FS_USAGE}=="filesystem", \
ENV{SYSTEMD_WANTS}+="mender-usb-update.service", \
RUN{program}+= "/usr/bin/systemd-mount \
               --no-block --automount=yes --collect $devnode /media"
```

media.mount

usb-update.service

```
[Unit]
Description=Installs USB updates using the Mender client
Requires=media.mount
After=media.mount

[Service]
ExecStart=/usr/bin/install-usb-update.sh

[Install]
WantedBy=media.mount
```

Debugging

```
# Observe systemd-udevd mechanism
# Is the USB device detected? Errors?
systemctl status systemd-udevd.service -l
journalctl -u systemd-udevd.service -f



# Observe the install script
systemctl status usb-update.service -l
journalctl -u usb-update.service -f
```

Vielen Dank!

Nicht aufgepasst?
Hier nochmal zum Nachlesen!

Anna-Lena Marx
Embedded Systems Developer

Anna-Lena Marx

anna-lena.marx@inovex.de

marx.engineer

inovex

image:

gdoc element: