

Crafting an Edge Compute Platform for Predictive Maintenance with Isar and isar-cip-core

Lukas Rabener

Innomotics GmbH

lukas.rabener@innomotics.com

Alexander Heinisch

Siemens AG

alexander.heinisch@siemens.com

Alexander Heinisch & Lukas Rabener

INNOMOTICS

We have a well-balanced footprint across regions and industries.

17

factories

49

countries with sales & service setup

11

project execution centers

Oil & Gas
Chemicals
CCUS
Power
Infrastructure
Metals
Mining
Water
Wastewater
HVAC
Food & Beverage
Pulp & Paper



HVM FACTORIES

- Norwood, USA
- Berlin, Germany
- Erlangen, Germany
- Nuremberg, Germany
- Sibiu, Romania
- Drásov, Czech Republic
- Tianjin, China
- Mumbai, India

MVD FACTORIES

- New Kensington, USA
- Jundiaí, Brazil
- Nuremberg, Germany
- Shanghai, China
- Mumbai, India

LVM FACTORIES

- Tuebingen, Germany
- Maroldsweisach, Germany
- Mohelnice, Czech Republic
- Frenštát, Czech Republic
- Yizheng, China
- Tianjin, China

MV Drives and Systems

Product portfolio

GH180 a/c



GH150 / SH150 / GH180 w/c



GL150 / SL150



GM150 / SM150



Our Industry Sectors



Why INSPIRE IQ?

Problem, Impact, Solution

Problem

99% uptime still means

80+ hours of unplanned downtime/year 

70% of companies
don't know their asset maintenance or upgrade schedule¹ 

82% of companies
had >1 unplanned downtime over the past 3 years
Average number of outages = 2¹ 

Impact

Unplanned
downtime costs

10X
as much as planned
downtime² 

\$10k – \$250k/hr
estimated **typical cost**
of downtime³ 

Solution

INSPIRE IQ

a holistic solution from Siemens to securely digitize your drive train.

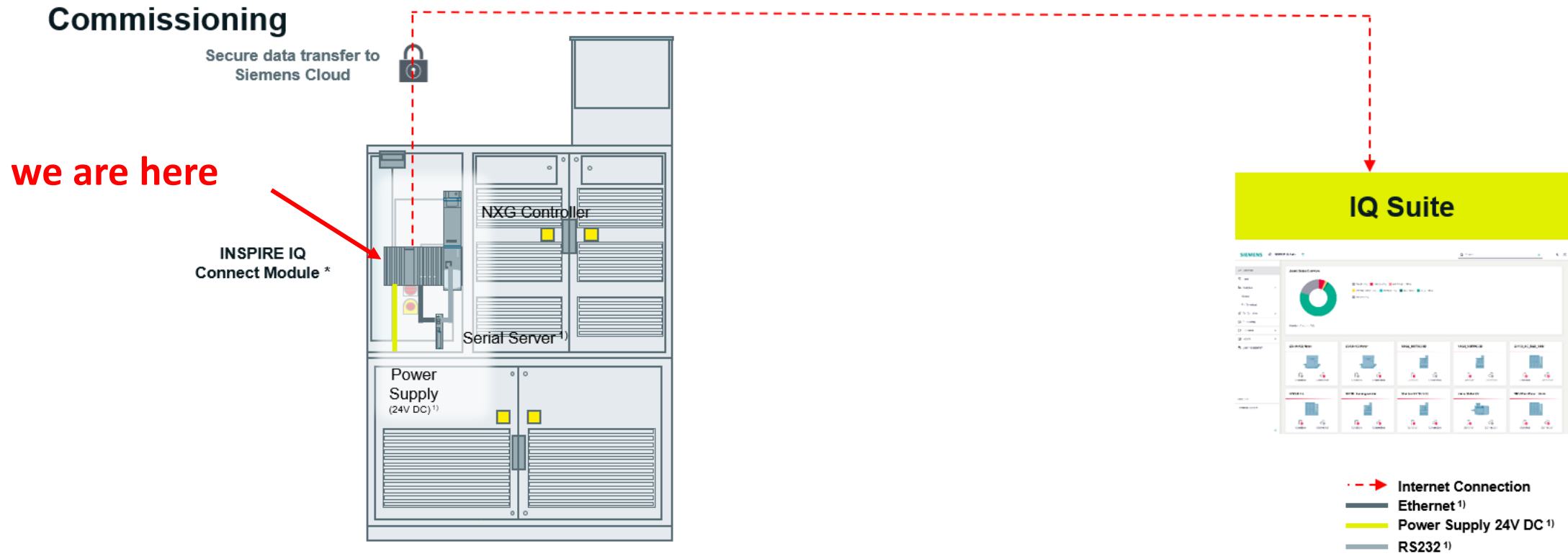
Partnership
to unlock your

**productivity,
availability
& serviceability**

References: 1 Vanson Bourne Research Study | 2 ARC Advisory Group | 3 Stratus

INSPIRE IQ connectivity

Connect 500 – Perfect Harmony Drives



Perfect Harmony Drive with Option C68

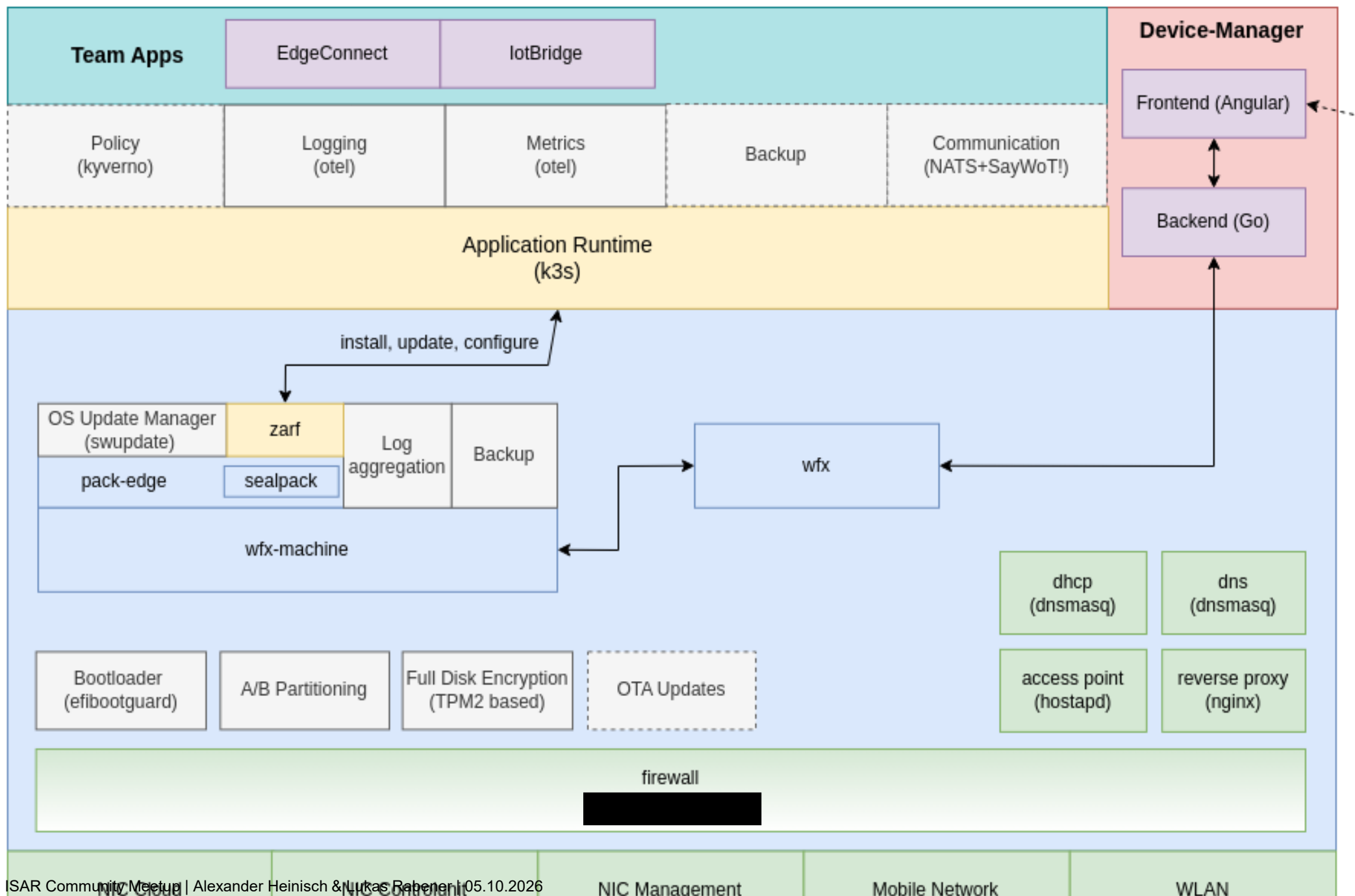
¹⁾ Preinstalled ex-factory

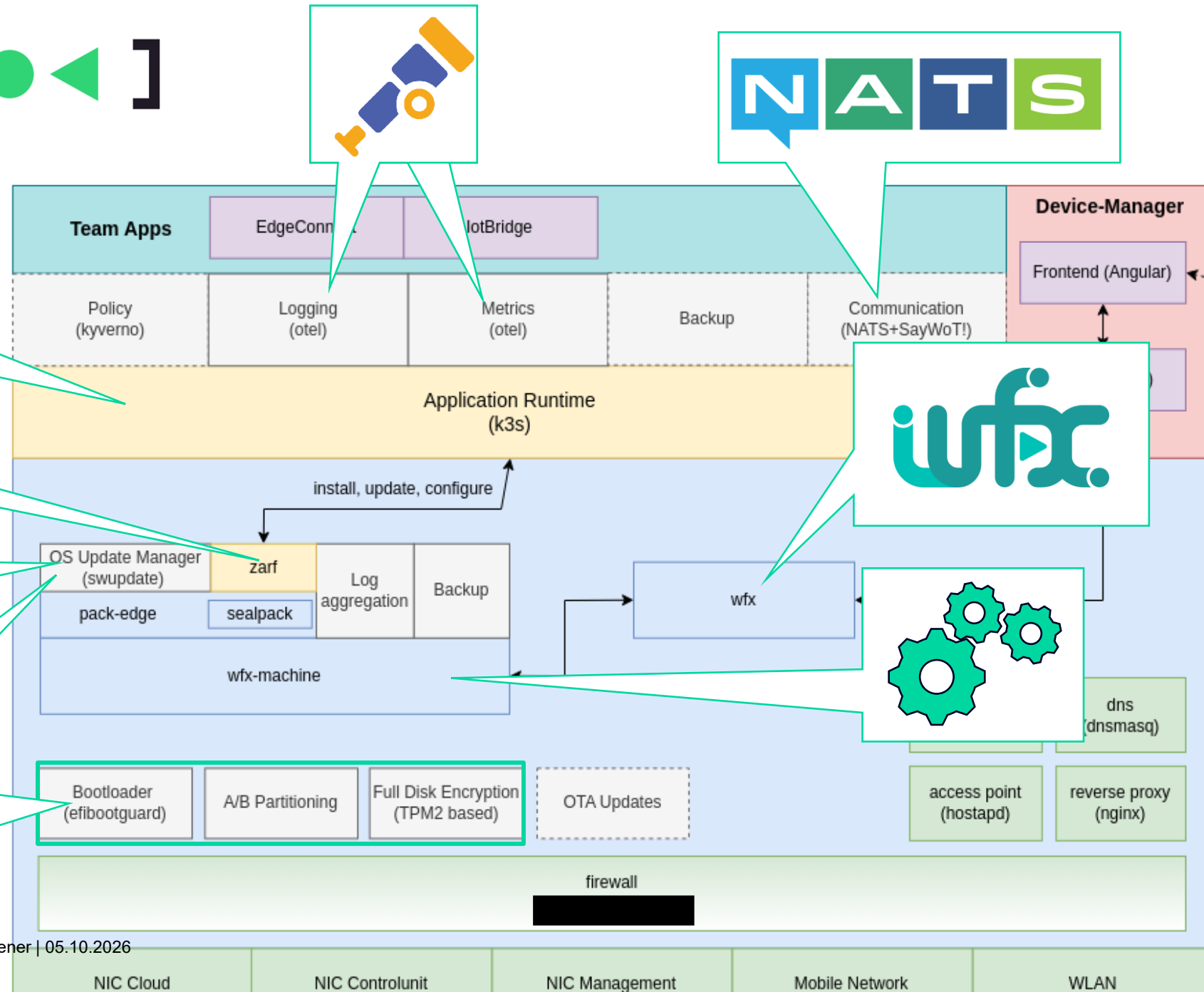
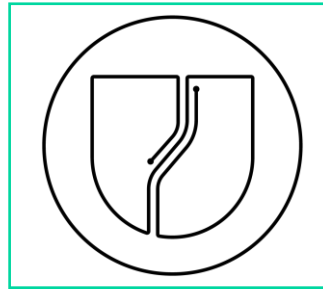
Connect 500

- Innomotics Edge Device
Shipped with Innomotics MV drive to enable Inspire IQ connectivity
- Primarily Airgapped
- Optional: Gateway for mobile Networking
- Collection | Preprocessing | Forwarding
- Service Provider for control assets
- Application runtime K3s
- Reuse / Shift of existing applications (Cloud to Edge)



Architecture





Image

High Level Overview

Image Build

- ISAR: Integration System for Automated Root filesystem generation
<https://github.com/ilbers/isar>
 - Build Debian based embedded systems
 - Debian: Tested binary packages, tools, security updates
 - Yocto: bitbake, recipes, layers
- Civil Infrastructure Platform (CIP):
<https://www.cip-project.org/>
 - open source “base layer” of industrial grade software
 - super-long-term maintenance of the Linux kernel
- gitlab-ci based builds (w. support for apt caching)
- LAVA for automated OS tests (qemu and ssh based for HIL)
- CIS Benchmarks in addition to IEC62443 compliant hardening



Image Build Highlights

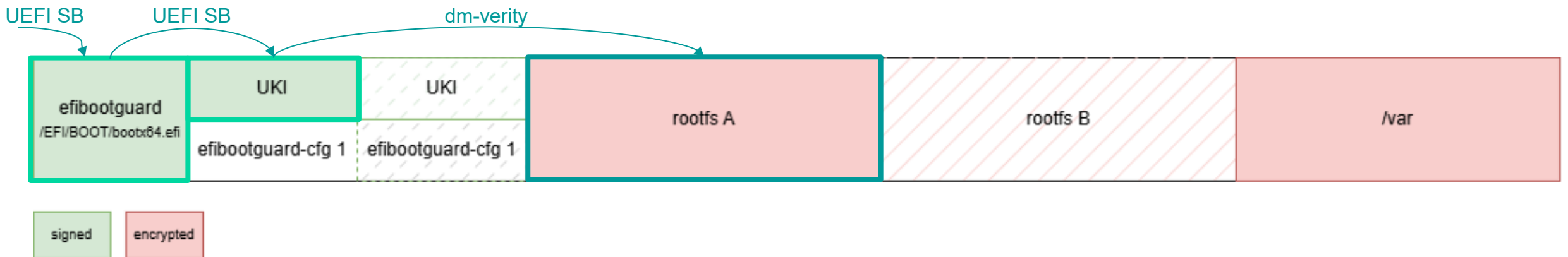
- kas-container
- Heavy use of multiconfig [dev,qa,prod]{qemu,vmware,lex-dsc-nano}[installer,[pxe-boot-server]]
- Upstream debian kernel + Custom Initramfs
- Driver integration:
 - Power Cap driver (LTC3350)
 - LED driver (F75111)
 - Watchdog driver (F75111) incl. Bootloader Integration
- Customization for preferred signing provider
 - UKI / kernel modules / swu signing
 - AWS KMS for {prod, qa}
 - snakeoil for dev

https://github.com/innomotics/f75111_linux_driver

Trusted Boot

UEFI Secure Boot (End-to-End)

- Setup of PK, KEK, db
- no MOK util – no MS Keys
- AWS KMS based signing infrastructure
- signed bootloader, signed UKI
- dm-verity of ro-rootfs



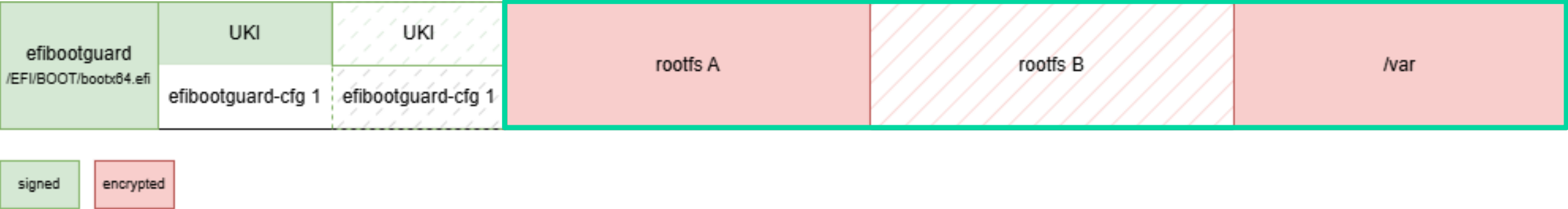
Full Disk Encryption

- LUKS
- TPM2 sealed key
- PCR measurement on Secure Boot state
- rootfs and data partitions

Table 12-1. Example PCR Allocation

PCR Number	Allocation
0	BIOS
1	BIOS configuration
2	Option ROMs
3	Option ROM configuration
4	MBR (master boot record)
5	MBR configuration
6	State transitions and wake events
7	Platform manufacturer specific measurements
8-15	Static operating system
16	Debug
23	Application support

Source: A Practical Guide to TPM 2.0

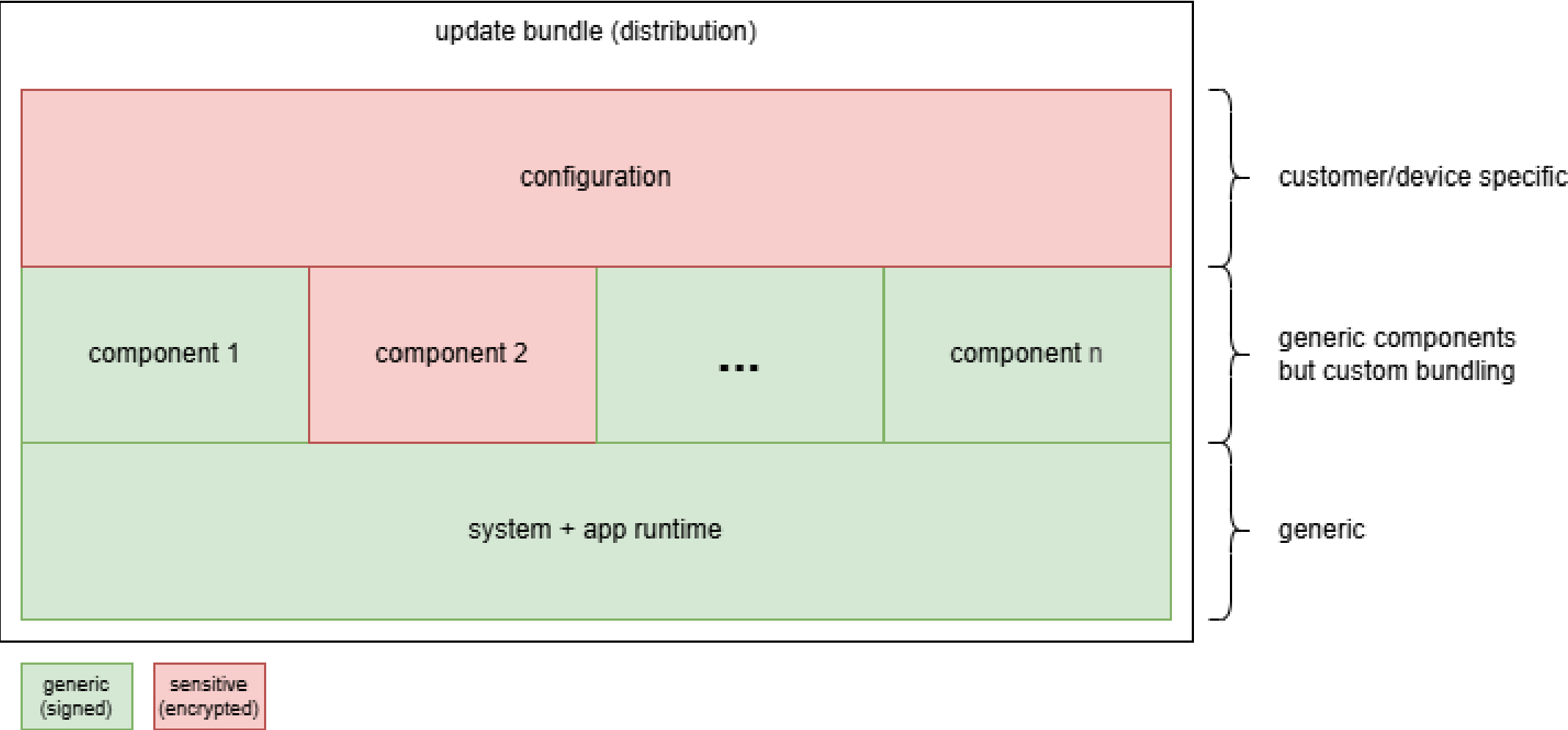


Updates

Distribution Based

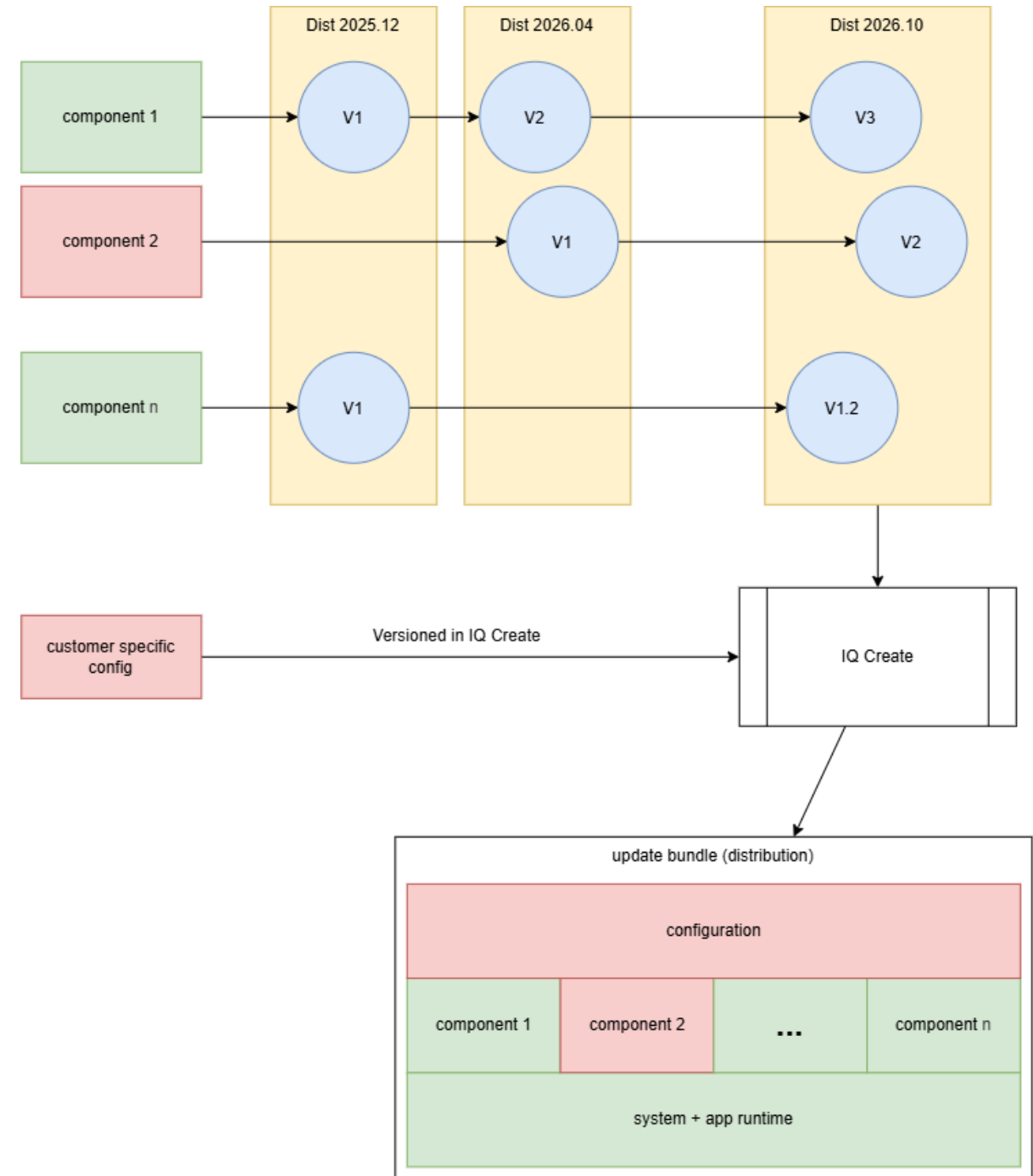


Distribution Based Updates



Distribution Based Updates

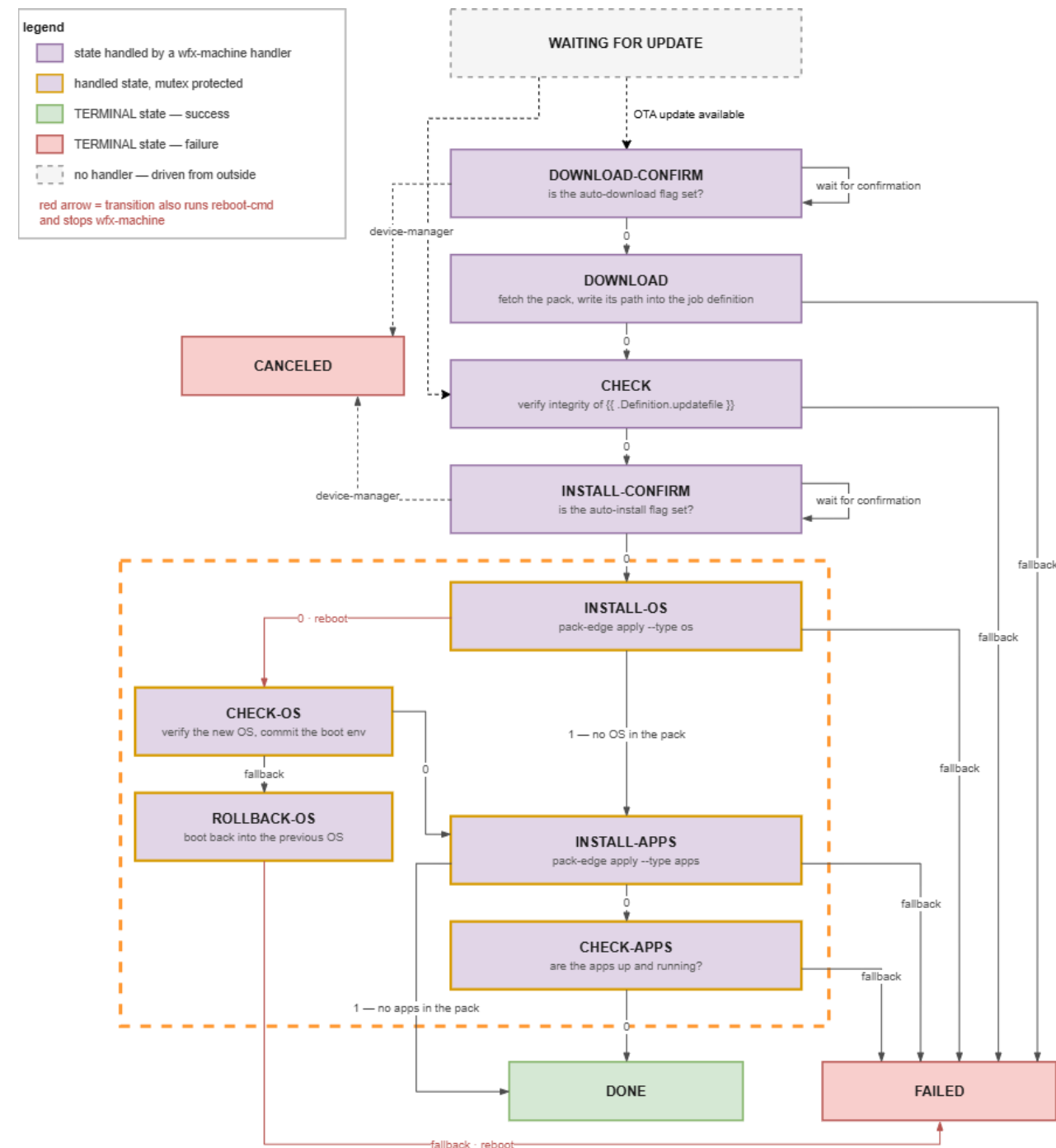
- Specific versions bundled to a distribution
- Individual release-cycle of components
- Reduce test complexity
Well tested set of combinations
- Easier airgapped distribution
Single file vs. multiple interdependent sources



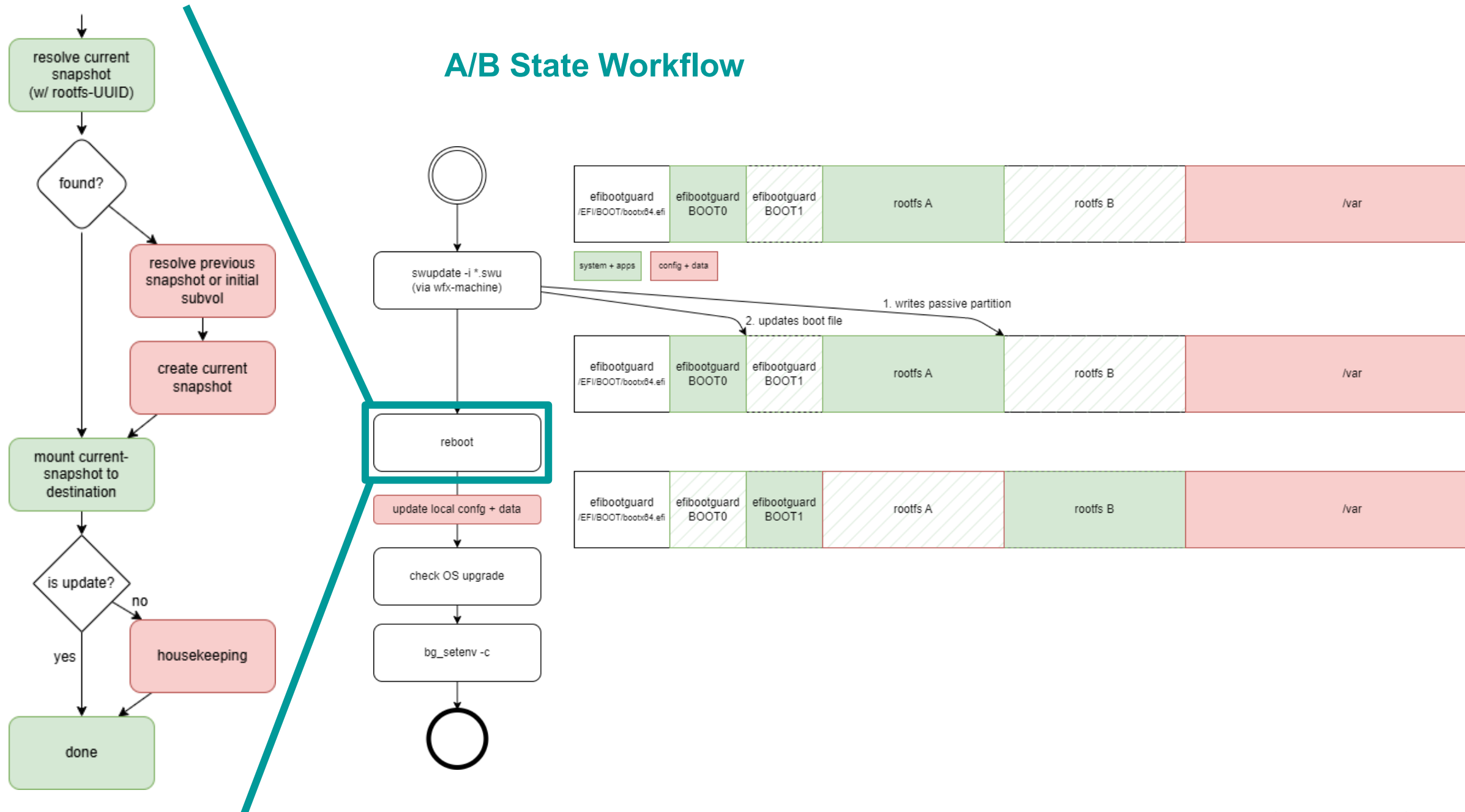
Update Workflow: wfx-machine

- Yaml based DSL for wfx
- Handles workflow lifecycle mgmt
- Mutex states
only one job resides in critical section
- Declarative state transitions
Based on command result
- Used for complex or long running processes
e.g. updates, log bundling, ...

PR on the way: <https://github.com/siemens/wfx>



A/B State Workflow

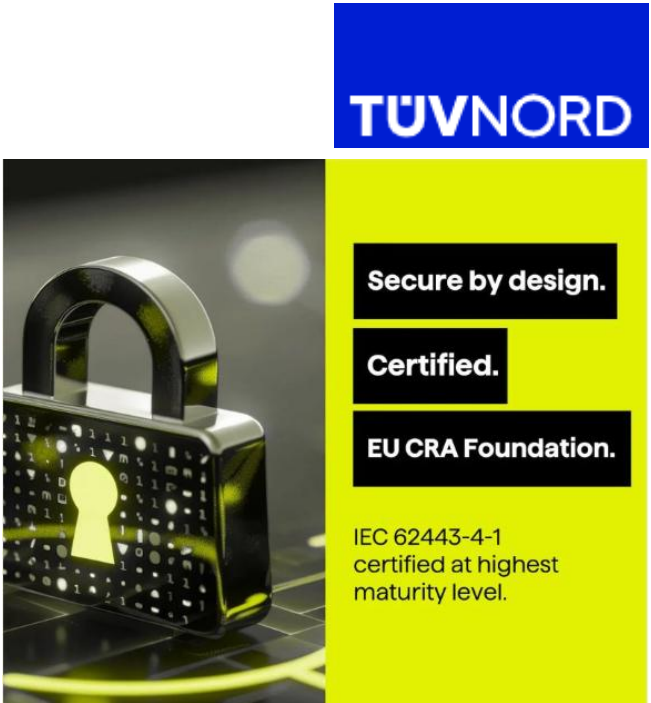


Hardening

Target IEC62443-4-2 Security Level 3

- Innomotics development process achieved certification according to IEC 62443-4-1 at highest Maturity Level (ML-4)
- IEC62443-4-2 Security Level 3 as target for Edge Platform
- Certification planned for FY26/27; Auditor TÜV Nord

FR1 – Identification & Authentication Control	CR 1.1 RE (2) Multifactor authentication for all interfaces	3
FR1 – Identification & Authentication Control	CR 1.2 RE (1) Unique identification and authentication	3
FR1 – Identification & Authentication Control	CR 1.5 E (1) Hardware security for authenticators	3
FR1 – Identification & Authentication Control	CR 1.7 RE (1) Password generation and lifetime restrictions for human users	3
FR1 – Identification & Authentication Control	CR 1.9 RE (1) Hardware security for public key-based authentication	3
FR2 – Use Control	CR 2.7 – Concurrent session control	3
FR2 – Use Control	CR 2.9 RE (1) Warn when audit record storage capacity threshold reached	3
FR3 – System Integrity	CR 3.4 RE (2) Automated notification of integrity violations	3
FR3 – System Integrity	HDR 3.11 RE (1) Notification of a tampering attempt	3
FR4 – Data Confidentiality	CR 4.2 RE (1) Erase of shared memory resources	3
FR4 – Data Confidentiality	CR 4.2 RE (2) Erase verification	3
FR6 – Timely Response to Events	CR 6.1 RE 1 – Programmatic access to audit logs	3
FR7 – Resource Availability	CR 7.6 RE (1) Machine-readable reporting of current security settings	3



Read-only rootfs

- no overlay for /etc used
- Symlinks created at build time point to configs on /var
- Configs are rendered out of immutable go templates on runtime via apply service

```
overlay-mounts: |  
    INITRAMFS_OVERLAY_MOUNT_OPTIONS ?= "defaults,nodev,nosuid"  
    INITRAMFS_OVERLAY_PATHS = ""
```

```
[Match]  
Name=controlunit  
  
[Network]  
DHCP=no  
{{- if .Network.InterfacesConfig.controlunit.CIDR }}  
Address={{ .Network.InterfacesConfig.controlunit.CIDR }}  
{{- end }}
```

Requirements of IEC62443:

CR 3.4 – Software and information

CR 3.4 RE (1) Authenticity of software

HDR/SAR 3.2 – Protection from malicious code

Firewall

- before:
 - K3s in host network namespace
 - K3s continually adds, updates, and removes firewall rules
- now:
 - K3s in dedicated network namespace
 - separation of concerns: host firewall independent of K3s
 - default-deny firewall on the host



Requirements of IEC62443:

- CR 7.1 – Denial of service protection
- CR 7.1 RE(1) Manage communication load from component
- CR 7.6 – Network and security configuration settings
- CR 7.7 – Least functionality

```
[Unit]
After=cluster-private-network.service cluster-cleanup.service

[Service]
NetworkNamespacePath=/run/netns/${K3S_NETNS_NAME}

NoNewPrivileges=yes
PrivateTmp=yes
ProtectSystem=strict
ProtectHome=yes
RestrictAddressFamilies=AF_INET AF_NETLINK AF_UNIX
```

CIS Benchmark

- Integrated in CI/CD via lava job:
 - Debian hardening scripts from: github.com/ovh/debian-cis
 - K8s : github.com/aquasecurity/trivy



Requirements of IEC62443:
CR 7.6 RE (1) Machine-readable reporting of current security settings

Assets 3 Delete selected

☐	Name	Size	Created	
☐	▼ k3s-cis-report.json	23.42 KiB	1 week ago	⋮
☐	▼ iptables-list.txt	14.10 KiB	1 week ago	⋮
☐	▼ cisreport.txt	178.59 KiB	1 week ago	⋮

```

device_type: qemu
job_name: CIS Benchmark Test on IQ Edge OS QEMU
timeouts:
  job:
    minutes: 25
  action:
    minutes: 5
  connection:
    minutes: 5
priority: medium
visibility: public
context:
  arch: amd64
  lava_test_results_dir: "/home/lava-%s"
actions:
- deploy:
  timeout:
    minutes: 5
  to: tmpfs
  images:
    rootfs:
      url: file://IMAGE_PATH
      image_arg: -drive file={rootfs},discard=unmap,if=none,id=disk,format=raw -drive if=pflash,format=
      compression: xz
# [TRUNCATED]

- test:
  timeout:
    minutes: 15
  definitions:
# [TRUNCATED]
- repository: https://lava_test_token:$GIT_TOKEN@code.siemens.com/sidrive-iq/teams/team-ecs/edge-mgmt
  from: git
  path: tests/cisscan/cis.yaml
  branch: main
  name: cisscan

```

```

metadata:
  name: debian-cis
  description: Run debian-cis benchmark
  os:
    - debian
  devices:
    - qemu
run:
  steps:
    - cp ./tests/cisscan/debian/default /etc/default/cis-hardening
    - sed -i "s#CIS_LIB_DIR=.##CIS_LIB_DIR=$(pwd) /tests/cisscan/lib# " /etc/default/cis-hardening
    - sed -i "s#CIS_CHECKS_DIR=.##CIS_CHECKS_DIR=$(pwd) /tests/cisscan/bin/hardening# " /etc/default/cis-hardening
    - sed -i "s#CIS_CONF_DIR=.##CIS_CONF_DIR=$(pwd) /tests/cisscan/etc# " /etc/default/cis-hardening
    - sed -i "s#CIS_TMP_DIR=.##CIS_TMP_DIR=$(pwd) /tests/cisscan/tmp# " /etc/default/cis-hardening
    - chmod tx /tests/cisscan/bin/hardening.sh
    - ./tests/cisscan/bin/hardening.sh --audit-all

```

```

[1;32m99.99_check_distribution [ OK ] Your distribution is debian and the version is supported[0m
[1;32m99.99_check_distribution [ OK ] Check Passed[0m
##### SUMMARY #####
Total Available Checks : 241
Total Runned Checks : 241
Total Passed Checks : [ 140/241 ]
Total Failed Checks : [ 101/241 ]
Enabled Checks Percentage : 100.00 %
Conformity Percentage : 58.09 %
+ set +x
<LAVA_SIGNAL_ENDRUN 1_cisscan 411_1.1.3.5>
Received signal: <ENDRUN> 1_cisscan 411_1.1.3.5
Ending use of test pattern.
Ending test lava.1_cisscan (411_1.1.3.5), duration 370.64
case: 1_cisscan
commit_id: 44b54f670ed12188ffabe1779cd6fc51456ebdb
definition: lava
duration: 370.64
namespace: common
path: tests/cisscan/cis.yaml
repository: https://lava_test_token:$GIT_TOKEN@code.siemens.com/sidrive-iq/teams/team-ecs/edge-mgmt/testing/lava-tests.git
result: pass
revision: unspecified
uuid: 411_1.1.3.5

```

Results

Factory Reset

- integrated via isar-cip-core initramfs hook:
 - FDE support? (let's discuss: [#164](#))
- SL-3 requires additional "erase verification"
 - implemented via hardware led

Requirements of IEC62443:

CR 4.2 – Information persistence

CR 4.2 RE (1) Erase of shared memory resources

CR 4.2 RE (2) Erase verification

```
# Enable secure decommission (factory reset) by default for every image.
OVERRIDES:append = ":factory-reset"

IMAGE_INSTALL += "factory-reset-helper"
PREFERRED_PROVIDER_factory-reset-helper = "factory-reset-helper-efi"

INITRAMFS_FACTORY_RESET_DEVICES = "/dev/disk/by-partlabel/var"
INITRAMFS_FACTORY_RESET_CLEAR_TPM = "1"
INITRAMFS_FACTORY_RESET_LUKS_FORMAT_TYPE = "ext4"
```

PenTest Fraunhofer IOSB

- Protocol fuzzing on all interfaces:
 - Ethernet, ARP, IPv4, TCP, UDP, HTTP, HTTPS
 - Monitoring: ARP, TCP, HTTP, HTTPS
 - up to 10.000 mutations per field
- Vulnerability Scan
 - OpenVAS (github.com/greenbone/openvas-scanner)
- Modbus Client Tests
 - In-house application of Innomotics



Automating Tests With LAVA and BATS

Unit and Integration Testing to avoid
Regressions

Diogo Mattioli & Lukas Rabener

INNOMOTICS

Automated Testing (ISAR Meet Up 2025)

- before:
 - tests cloned by Lava job & executed
- now:
 - including tests into rootfs
 - test recipe included in dev-image
- Enables Testing during development
 - ➔ run tests with a single CMD
 - ➔ qemu, VM or on hardware

```
329 tests, 0 failures, 1 skipped in 214 seconds
```

```
inherit dpkg-raw

DESCRIPTION = "Testing files"
DEBIAN_DEPENDS += "bats, dnsutils, curl, sntp"

FILESEXTRAPATHS:prepend := "${TOPDIR}/../:"

SRC_URI = " \
    file://tests/ \
    "

do_install[cleandirs] += "\
    ${D}/tests \
    "

do_install() {
    cp -rf ${WORKDIR}/tests/* ${D}/tests/
}
```

Device-Bootstrapping

Target-bootstrapper

Factory integration (wedding-phase)

1. Perform as many setup tasks as possible here

Preparation

1. System gets deployed on a **batch of devices**
2. Setup Secure Boot, Deploy Image, ...
3. Report installation log and device specific information to ERP
4. Reboot + **enroll encryption + expand last partition**
5. Devices are put in storage

2.+ 3. Deploy steps automatically

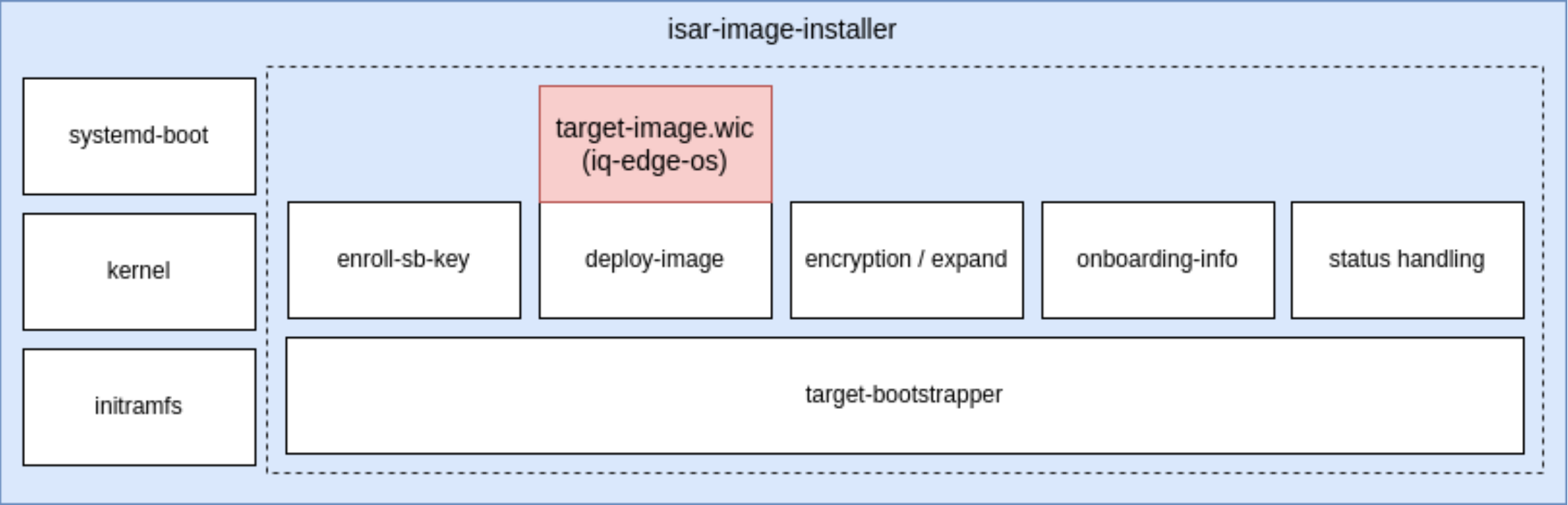
Cheap

Assembly

6. During manufacturing of asset the worker „grabs“ one of the devices
7. Performs mechanical and electrical installation of the device
8. Boots the presetup device
9. Associates device with the actual asset (based on a unique device key shown on HMI / Browser)
10. Installs asset specific onboarding bundle
11. Performing Factory Acceptance Test (FAT)

Expensive

1. isar-image-installer: target-bootstrapper framework



<https://github.com/ilbers/isar/tree/master/meta-isar/recipes-installer/target-bootstrapper>

```

6 header:
7   version: 14
8
9 local_conf_header:
10  target-bootstrapper-innomotics: |
11  OVERRIDES .- ".target-bootstrapper"
12
13  TARGET_BOOTSTRAPPER_ADDITIONAL_PACKAGES = ""
14  TARGET_BOOTSTRAPPER_ADDITIONAL_PACKAGES:append:lex-dsc-nano = " status-led"
15  TARGET_BOOTSTRAPPER_ADDITIONAL_PACKAGES:append = " install-secureboot-certificates deploy-image device-info-collector"
16
17  TARGET_BOOTSTRAPPER_TASK_status-led[script] = "blink-led.sh"
18  TARGET_BOOTSTRAPPER_TASK_status-led[workdir] = "/usr/bin"
19  TARGET_BOOTSTRAPPER_TASK_status-led[effort] = "1"
20
21  TARGET_BOOTSTRAPPER_TASK_install-secureboot-certificates[script] = "install-secureboot-certs.sh"
22  TARGET_BOOTSTRAPPER_TASK_install-secureboot-certificates[workdir] = "/usr/bin"
23  TARGET_BOOTSTRAPPER_TASK_install-secureboot-certificates[effort] = "1"
24
25  TARGET_BOOTSTRAPPER_TASK_deploy-image[script] = "deploy-image-wic.sh"
26  TARGET_BOOTSTRAPPER_TASK_deploy-image[workdir] = "/usr/bin"
27  TARGET_BOOTSTRAPPER_TASK_deploy-image[effort] = "2"
28
29  TARGET_BOOTSTRAPPER_TASK_device-info-collector[script] = "device-info-collector.sh"
30  TARGET_BOOTSTRAPPER_TASK_device-info-collector[workdir] = "/usr/bin"
31  TARGET_BOOTSTRAPPER_TASK_device-info-collector[effort] = "1"
32

```

2. isar-image-installer: unattended mode

- no interaction needed
- configuration via kernel cmdline

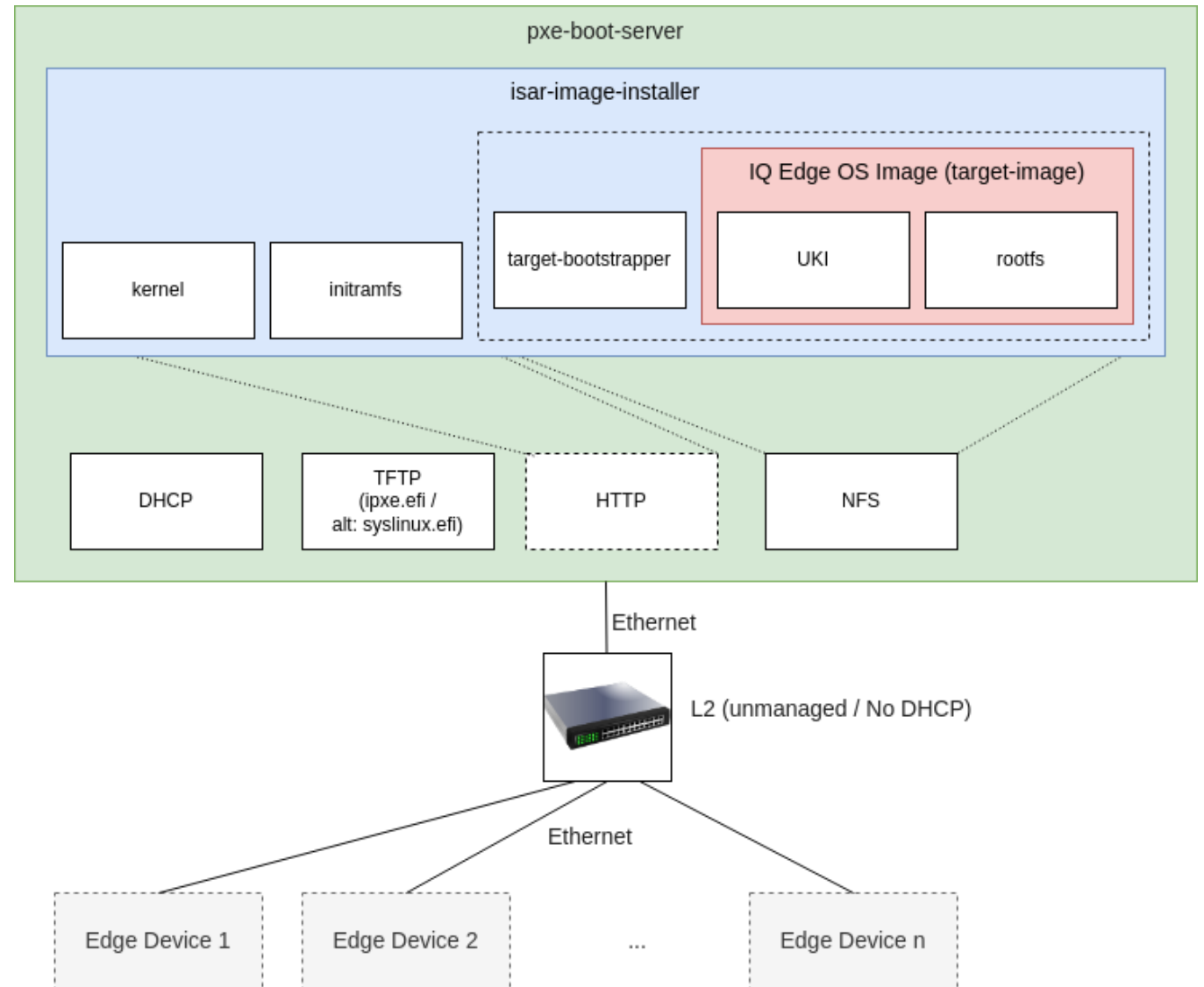
```
32  
33  ADDITIONAL_KERNEL_CMDLINE:append = " \\  
34  .... installer.image.uri=/install/${PXE_TARGET_IMAGE_DATA_FILE} \\  
35  .... installer.target.dev=${INSTALLER_TARGET_DEVICE} \\  
36  .... installer.target.overwrite=OVERWRITE \\  
37  .... "  
38  ADDITIONAL_KERNEL_CMDLINE:append:pxe-nfsdata = " \\  
39  .... device-info-collector.target=nfs://${PXESERVER_IP}:/var/installer/data \\  
40  .... "
```

<https://github.com/ilbers/isar/tree/master/meta-isar/recipes-installer/deploy-image>

3. meta-isar-pxeboot

- Scale bootstrapping of multiple devices
- Multiconfig
 - to provide single click deployment
 - Examples for QEMU, Virtual Box, ...
- Support for pxe / ipxe
 - ipxe bootstrap
 - faster transfer of kernel / initramfs
- Installer rootfs via NFS

<https://github.com/innomotics/meta-isar-pxeboot>



What about AI?

How we work with AI (claude code)

- repo level skills managed in git (.claude/skills/debug-edge)
 - /debug-edge skill for ssh based debugging (dev image qemu / VM)
- chrome-devtools MCP for UI / e2e debugging



```
Claude Code v2.1.284
Sonnet 5 with high effort · Amazon Bedrock
~/projects/os/meta-innomotics-edge

> /debug-edge check the system status

• I'll start by checking connectivity to the edge device.

Ran 1 shell command

• Connectivity is good. Now checking for failed systemd units.

Ran 1 shell command

• System status check – step 1 & 2 complete:

- Connectivity: OK – SSH reachable, device responds.
- Failed units: None – systemctl --failed shows 0 loaded units, so no service failures at boot.

No issues found so far. Want me to continue checking boot errors, disk/mounts, resources, network, and k3s cluster health, or is this enough for now?

* Worked for 14s · done 9:15 AM
```

Demo

IQ EDGE OS

1.7.1

System Information:

Variant.....: IQ Edge OS (edge-umware) - Development Version
Distro.....: Debian GNU/Linux 13 (trixie)
Kernel.....: Linux 6.12.101+deb13-amd64 x86_64
Build-Id.....: 1.7.1

Interfaces:

Cloud.....: 192.168.160.132
Control Unit: 172.17.20.100
Management..: 192.168.0.185

Quick Start:

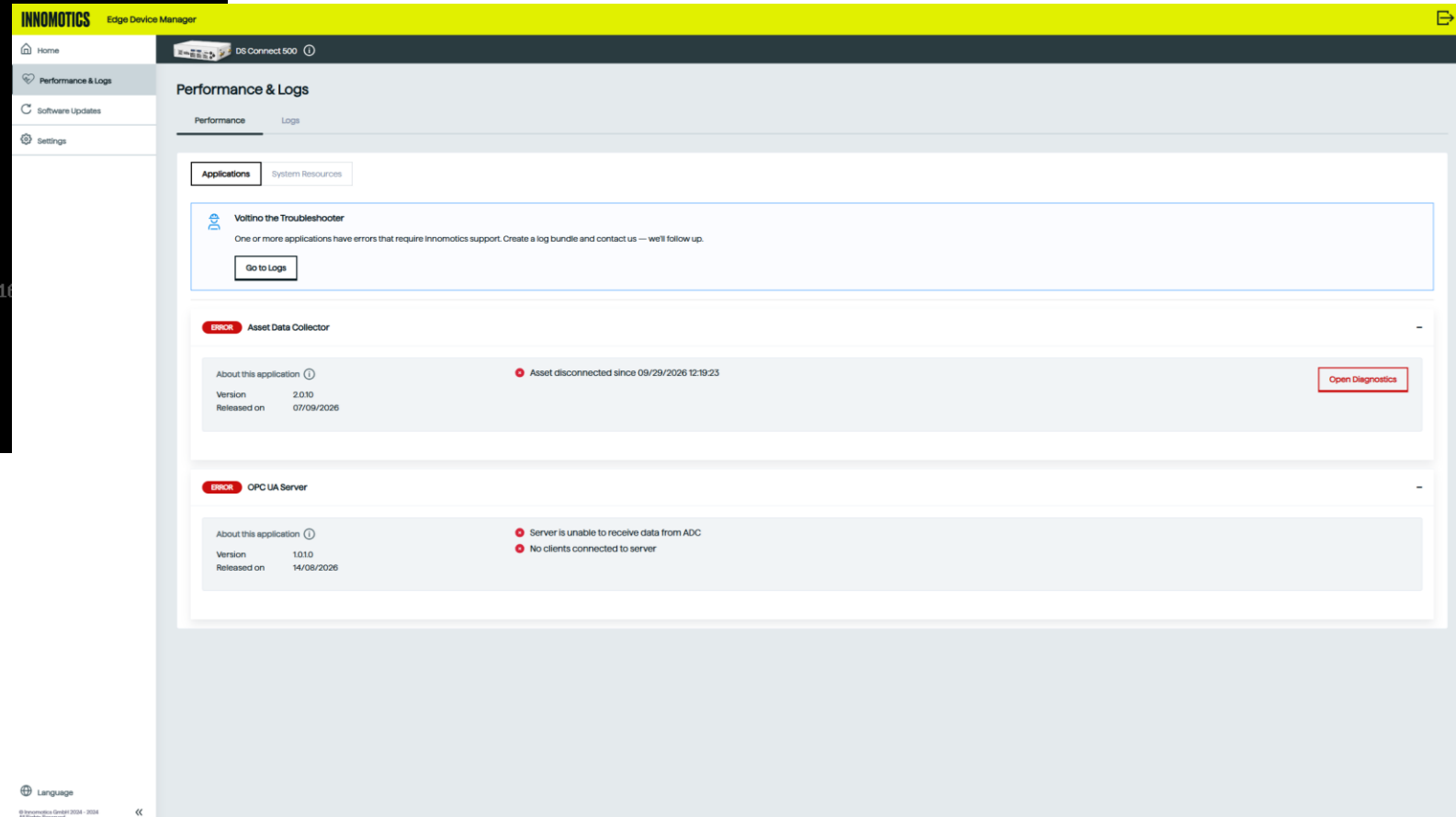
1. Connect to the management port and configure your host to use **DHCP**
2. Open: **<https://edge.innomotics.internal>**

Alternative (if DHCP is not available):

1. Configure your host's network interface with an address on the 192.168.0.0/24
2. Open: **<https://192.168.0.185>**

You are currently connected via: **tty1** (38400)

device-mgmt login:



Key Achievements

- **Reproducible ISAR/Debian-based image and build system in production**
kas-container build, multiconfig across dev/qa/prod and all targets
- **Secure boot chain end to end**
UEFI Secure Boot, UKI and module signing, AWS KMS integration
Full disk encryption bound to platform state
- **Robust A/B updates and distribution-based delivery**
pack-edge and wfx-machine driven update workflow, btrfs-snapshots
- **Automated device bootstrapping**
Unattended installer, target-bootstrapper framework, PXE boot (meta-isar-pxeboot)
- **Hardening towards IEC 62443-4-2 SL3**
Read-only rootfs, firewall, CIS benchmark, Fraunhofer IOSB pentest passed
- **Automated testing and AI-assisted engineering workflows established**

Q&A