Der Weg zu einem modernen Yocto Project Stack

Anna-Lena Marx

Anna-Lena Marx

Embedded Systems Dev at inovex

- Linux Kernel
- Yocto
- Android Embedded

@Allegra3141

Allegra42

Allegra
@social.linux.pizza

B.Sc. Computer Science

M.Sc. Embedded Systems

B.Eng. Electrical Engineering - *ongoing Hobby*

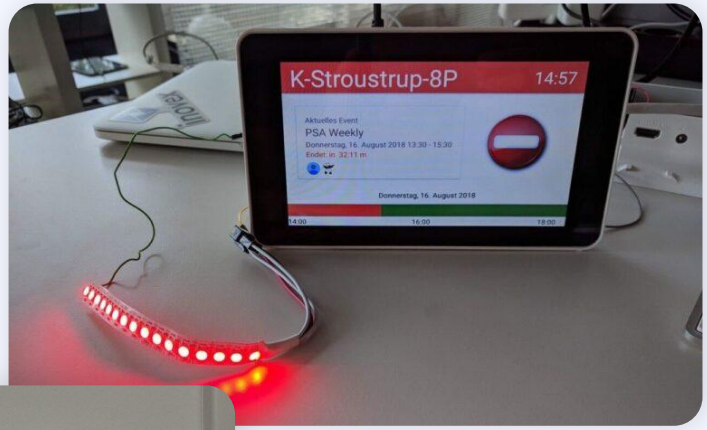
inovex

The Story behind

Meeting Room Information Screens at inovex

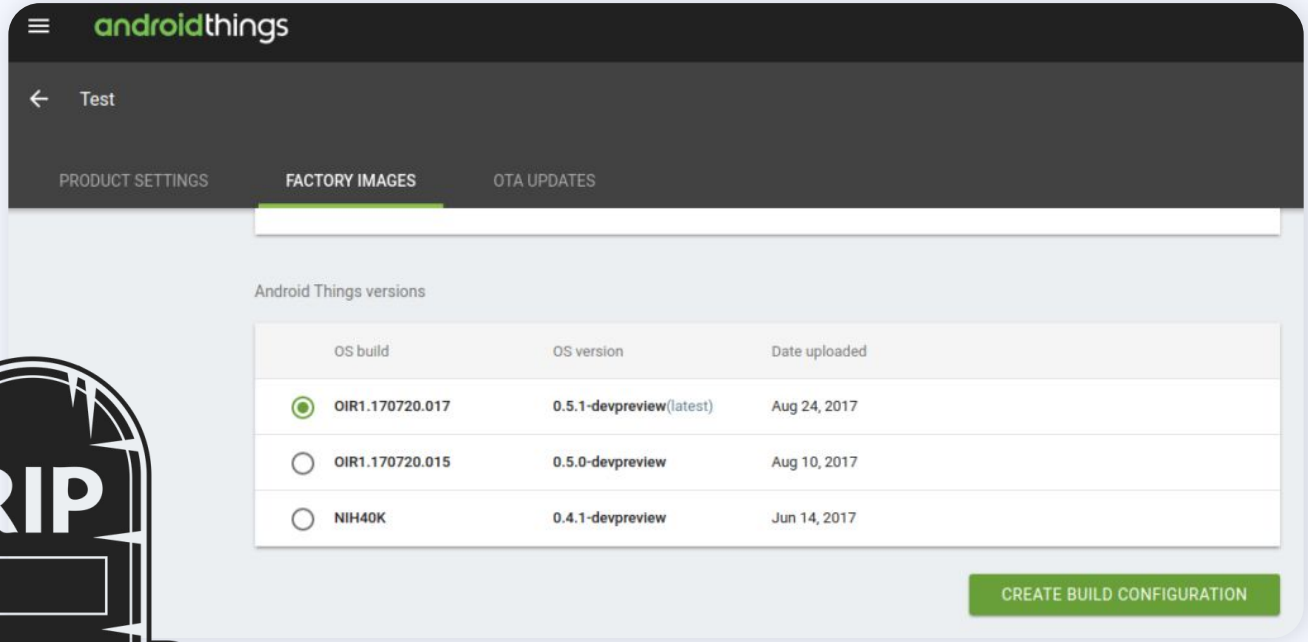
inovex

~ 25 Meeting rooms spread across 5 cities before Corona → lots of Meetings



inovex

Android Things was shutdown in 2022



[Killed by Google: Google Graveyard]

inovex

Reboot

What do we wish for a modern, maintainable System?

Our wishlist

- ❑ Full-stack patchability
- ❑ Version control
- ❑ Reproducible builds
- ❑ Long-Term maintainable
- ❑ Proper license (and version) management
- ❑ Android-like, secure and stable update mechanism
- ❑ Release management Over-the-Air
- ❑ Continuous Integration
- ❑ A vendor independent system on all levels

- ❑ **Full stack patchability**
- ❑ **Version control**
- ❑ **Reproducible builds**
- ❑ **Long-Term maintainable**

✓ **just use** 

- Hardware can be reused, no invest needed
- all points above are fulfilled by design

inovex

Managing Yocto - a side note

Name	Last commit
..	
meta-inovex	rename inovex-media => inovexmedia
meta-mender @ 5b518cc7	update layers
meta-openembedded @ ab9fca48	update layers
meta-raspberrypi @ 934064a0	Update submodule meta-raspberrypi in oder to fix linux-...
meta-security @ b76698c7	update layers
meta-virtualization @ c5f61e54	update layers
poky @ bba32338	update layers

```
<?xml version='1.0' encoding='UTF-8'?>
<manifest>
  <phytec pdn="PD21.1.0" release_uid="BSP-Yocto-FSL-i.MX8M-PD21.1.0" soc="iMX8M" supported_builds="
    phyboard-polaris-imx8m-3/phytec-headless-bundle/yogurt-vendor,
    phyboard-polaris-imx8m-3/phytec-headless-image/yogurt-vendor-secure,
    phyboard-polaris-imx8m-3/phytec-qt5demo-image/yogurt-vendor-xwayland,
    phyboard-polaris-imx8m-3/phytec-vision-image/yogurt-vendor-xwayland,
    phyboard-polaris-imx8m-4/-c populate_sdk phytec-qt5demo-image/yogurt-vendor-xwayland,
    phyboard-polaris-imx8m-4/phytec-headless-bundle/yogurt-vendor,
    phyboard-polaris-imx8m-4/phytec-headless-image/yogurt-vendor-secure,
    phyboard-polaris-imx8m-4/phytec-qt5demo-image/yogurt-vendor-xwayland,
    phyboard-polaris-imx8m-4/phytec-vision-image/yogurt-vendor-xwayland
  " bspextension="FSL" />

  <default revision="zeus" sync-j="2" remote="git.phytec" />

  <remote fetch="https://git.yoctoproject.org/git" name="yocto" />
  <remote fetch="https://github.com/Freescale" name="community" />
  <remote fetch="https://github.com/openembedded" name="oe" />
  <remote fetch="https://github.com/OSSystems" name="OSSystems" />
  <remote fetch="https://github.com/meta-qt5" name="QT5" />
  <remote fetch="https://github.com/meta-rust" name="rust" />
  <remote fetch="git://git.openembedded.org" name="python2" />
  <remote fetch="https://source.codeaurora.org/external/imx" name="CAF" />
  <remote fetch="https://github.com/rauc" name="rauc" />
  <remote fetch="https://github.com/kraj" name="clang" />
  <remote name="git.phytec" fetch="git://git.phytec.de" />
  <remote name="ssh.phytec" fetch="ssh://git@git.phytec.de" />

  <project name="poky" path="sources/poky" remote="yocto" revision="d88d62c20d7d8da85f02edb170dae0280624ad7e">
    <ignorebaselayer />
    <sublayer path="meta" />
    <sublayer path="meta-poky" />
  </project>

  <project name="meta-openembedded" path="sources/meta-openembedded" remote="oe" revision="2b5dd1eb81cd08bc065bc76125f2856e93
    <ignorebaselayer />
    <sublayer path="meta-oe" />
    <sublayer path="meta-networking" />
    <sublayer path="meta-python" />
    <sublayer path="meta-multimedia" />
    <sublayer path="meta-filesystems" />
    <sublayer path="meta-perl" />
    <sublayer path="meta-gnome" />
  </project>
```

Managing Yocto - a side note

There are lots of valid ways to work with Yocto - We moved to KAS because

- easy readable, clear syntax
- easy to use for non Yocto people
- comes already with a container!
  - really nice for CI
- persistent way to edit local.conf

✓ **handle Yocto easier**

```yaml
header:
  version: 11

distro: poky

defaults:
  repos:
    refspec: master

repos:
  poky:
    url: https://git.yoctoproject.org/git/poky
    path: "layers/poky"

    refspec: a361fb3df9c87cf12963a9d785a9f99faa839222
    layers:
      meta:
      meta-poky:
      meta-yocto-bsp:

  meta-openembedded:
    url: https://git.openembedded.org/meta-openembedded
    path: "layers/meta-openembedded"
    refspec: 82c75b466e55d7dca7a2364986ecb704cf63d141
    layers:
      meta-oe:
      meta-python:
      meta-filesystems:
      meta-networking:

  meta-mender:
    url: https://github.com/mendersoftware/meta-mender.git
    path: "layers/meta-mender"
    refspec: df6b158fb00fb5c9d524bff277122adfc1a5e6ff
    layers:
      meta-mender-core:

  meta-virtualization:
    url: https://git.yoctoproject.org/meta-virtualization
    path: "layers/meta-virtualization"
    refspec: a517e15529980f8401b25c99a2d7720ac2d8baae
```


inovex

❑ Proper license (and version) management

Why?

- we want/need to be license compliant
- we want to know exactly **what** we ship in our image
  - Component Name
  - Version
  - License

= Software Bill of Materials (SBOM)

✓ **INHERIT += "create-spdx"**

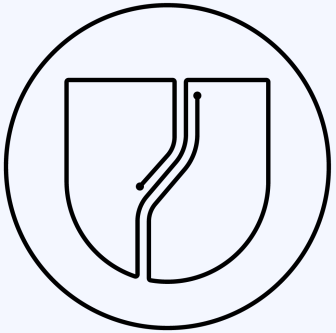
inovex

- ❑ **Android like, secure and stable update mechanism**
- ❑ **Release management Over-the-Air**

More exactly

- Image based updates
- A/B updates as state of the art
  - with a rollback/recovery mechanism
- A matching server implementation
  - allows starting updates for different groups of devices
  - OTA
  - a bit like Android Things console

- ❑ Android like, secure and stable update mechanism
- ❑ Release management Over-the-Air

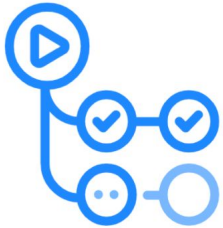


✓ and our winner is ...

... Mender

inovex

❑ Continuous Integration



GitHub Actions

CI/CD

Jenkins

✓ **we go with ... GitLab**

inovex

And what's about Continuous Delivery or Deployment?

- we do **not** want completely automated deployment on embedded devices!

- Yocto with CI gives us deployable artifacts
- Mender provides an API for automated artifact upload
- and make rolling out easy, nevertheless

❑ A vendor independent system on all levels

- hardware can be changed or operated in parallel
- various options on software-side
  - app
  - update system
  - CI/CD
  - we could even switch to Buildroot …

✓ **switching a component is not effortless,**
but possible without dropping the whole stack

inovex

The App side

A maintainable solution on embedded devices

inovex

Flutter

- flutter-pi Embedder
  - UI renders directly on GPU
  - no X11 / Wayland, ... needed
- no JavaScript / Webbrowser mess in Yocto
- keeps the complexity of Yocto away from app devs
- ships in a container for easy changes and updates

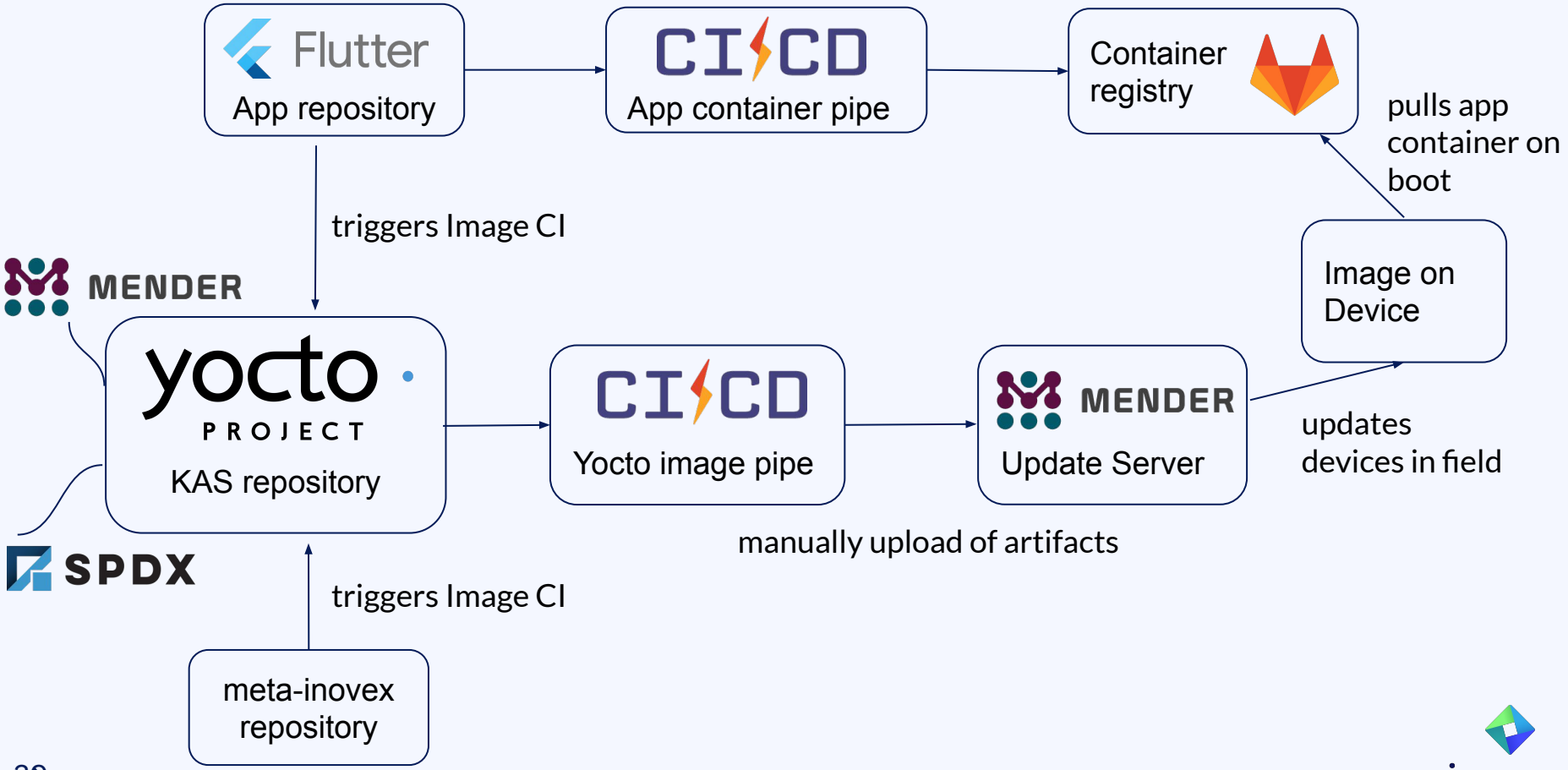
- **nice application without major pain points**

inovex

The App side

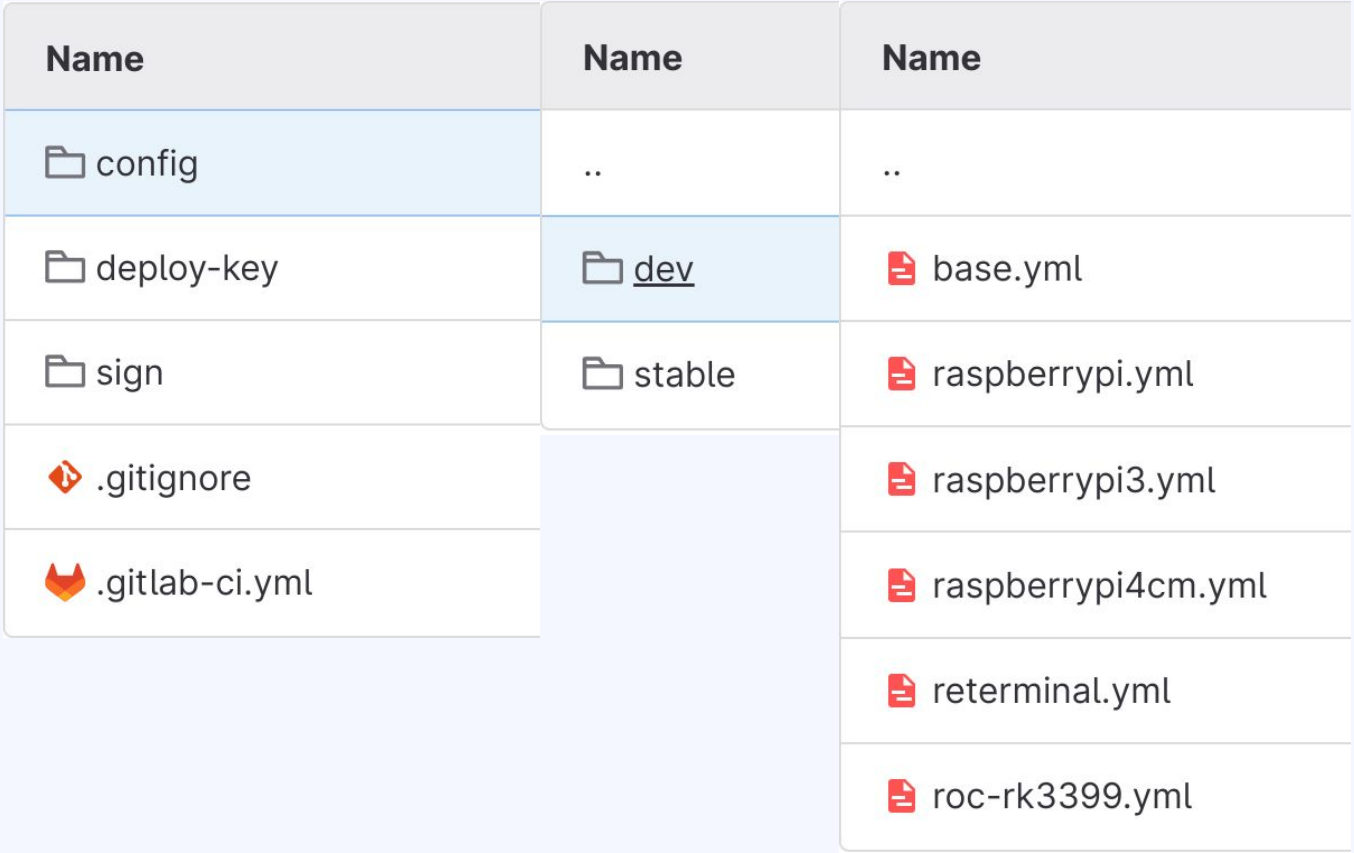
How it looks today

inovex


Flutter
App repository
CI/CD
App container pipe
Container registry
pulls app container on boot
triggers Image CI
MENDER
Image on Device
yocto PROJECT
KAS repository
CI/CD
Yocto image pipe
MENDER
Update Server
updates devices in field
manually upload of artifacts
SPDX
triggers Image CI
meta-inovex repository

inovex

The KAS config repository



inovex

The KAS configuration

```yaml
header:
  version: 11

distro: poky

defaults:
  repos:
    refspec: master

repos:
  poky:
    url: https://git.yoctoproject.org/git/poky
    path: "layers/poky"

    refspec: a361fb3df9c87cf12963a9d785a9f99faa839222
    layers:
      meta:
      meta-poky:
      meta-yocto-bsp:

  meta-openembedded:
    url: https://git.openembedded.org/meta-openembedded
    path: "layers/meta-openembedded"
    refspec: 82c75b466e55d7dca7a2364986ecb704cf63d141
    layers:
      meta-oe:
      meta-python:
      meta-filesystems:
      meta-networking:

  meta-mender:
    url: https://github.com/mendersoftware/meta-mender.git
    path: "layers/meta-mender"
    refspec: df6b158fb00fb5c9d524bff277122adfc1a5e6ff
    layers:
      meta-mender-core:

  meta-virtualization:
    url: https://git.yoctoproject.org/meta-virtualization
    path: "layers/meta-virtualization"
    refspec: a517e15529980f8401b25c99a2d7720ac2d8baae
```


inovex

Yocto CI

```
.build_yocto:
    extends: .setup_build

    script:
        - echo "Starting KAS Yocto build"
        #
        # checkout layers depending on the device configuration
        - kas checkout config/${BUILD_CONFIG}/${DEVICE_YAML}
        #
        # modify sources depending on build type
        - sed -i "s#{{ DOCKER_REGISTRY_AUTH }}#${DOCKER_REGISTRY_AUTH}#" layers/meta-inovex/recipes-config/inovex-config/files/docker-config.json
        #
        # set build string depending on build configuration
        - test "${BUILD_RELEASE}" != "release" &&
          sed -i "s#kirkstone-dev#kirkstone-${BUILD_CONFIG}-${CI_JOB_STARTED_AT}#" config/${BUILD_CONFIG}/base.yml
        - test "${BUILD_CONFIG}" = "stable" && test "${BUILD_RELEASE}" = "release" &&
          sed -i "s#kirkstone-dev#kirkstone-dev-${CI_JOB_STARTED_AT}#" config/${BUILD_CONFIG}/base.yml
        #
        # overwrite app tag for non dev builds
        - test "${BUILD_CONFIG}" = "stable" && test "${BUILD_RELEASE}" != "release" &&
          sed -i "s/:dev/:${APP_TAG}/" layers/meta-inovex/recipes-config/inovex-config/files/flutter-raumplanung.service
        - test "${BUILD_CONFIG}" = "stable" && test "${BUILD_RELEASE}" = "release" &&
          sed -i "s/:dev/:$CI_COMMIT_TAG/" layers/meta-inovex/recipes-config/inovex-config/files/flutter-raumplanung.service
        #
        # remove debug tweaks when building a release
        - test "${BUILD_CONFIG}" = "stable" && test "${BUILD_RELEASE}" = "release" &&
          sed '/EXTRA_IMAGE_FEATURES = "debug-tweaks"/d' config/stable/base.yml
        #
        # Start actual build
        - kas build config/${BUILD_CONFIG}/${DEVICE_YAML}
        - echo "Yocto build finished."
```


inovex

Yocto CI

```
build_rpi3_dev:
  extends: .build_yocto
  stage: build
  cache:
      key: YOCTO_BUILD_CACHE
      paths:
          - build/cache
          - build/sstate-cache
  rules:
      - if: $CI_COMMIT_TAG == null || $CI_COMMIT_TAG == ""
      - when: manual
        allow_failure: true

  variables:
    BUILD_CONFIG:   "dev"
    BUILD_RELEASE:  "no"
    DEVICE_YAML:    "raspberrypi3.yml"
```

```
build_rpi3_stable_release:
  extends: .build_yocto
  stage: build
  cache:
      key: no_cache
      paths:
          - .do-not-cache
  rules:
    - if: $CI_COMMIT_TAG != null
      when: always
    - when: manual
      allow_failure: true

  variables:
    BUILD_CONFIG:   "stable"
    BUILD_RELEASE:  "release"
    DEVICE_YAML:    "raspberrypi3.yml"
```


inovex

Trigger CI when changing meta-inovex

.gitlab-ci.yml 424 bytes

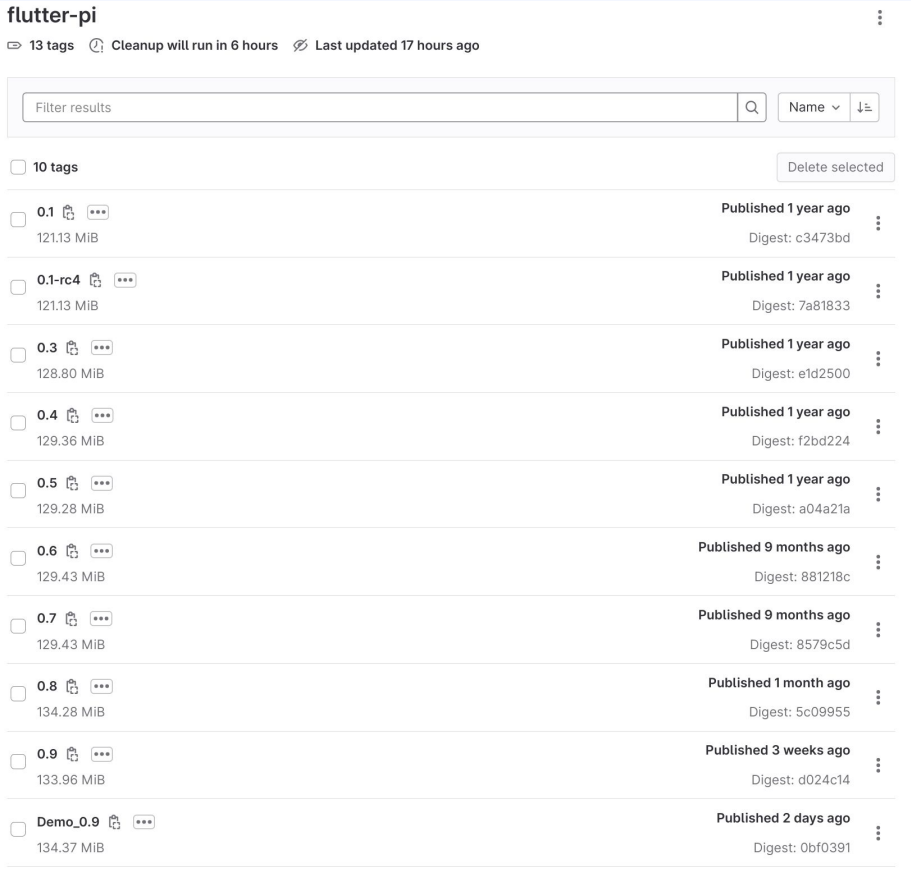
```yaml
stages:          # List of stages for jobs, and their order of execution
  - trigger-build

trigger-build:
  image: ubuntu:impish
  tags:
    - shared
  stage: trigger-build
  before_script:
    - apt-get update && apt-get install -y curl
  script:
    - echo "Trigger KAS Yocto CI"
    - curl -X POST --fail -F token=[redacted] -F ref=main h
```

Doc: Trigger pipelines in GitLab

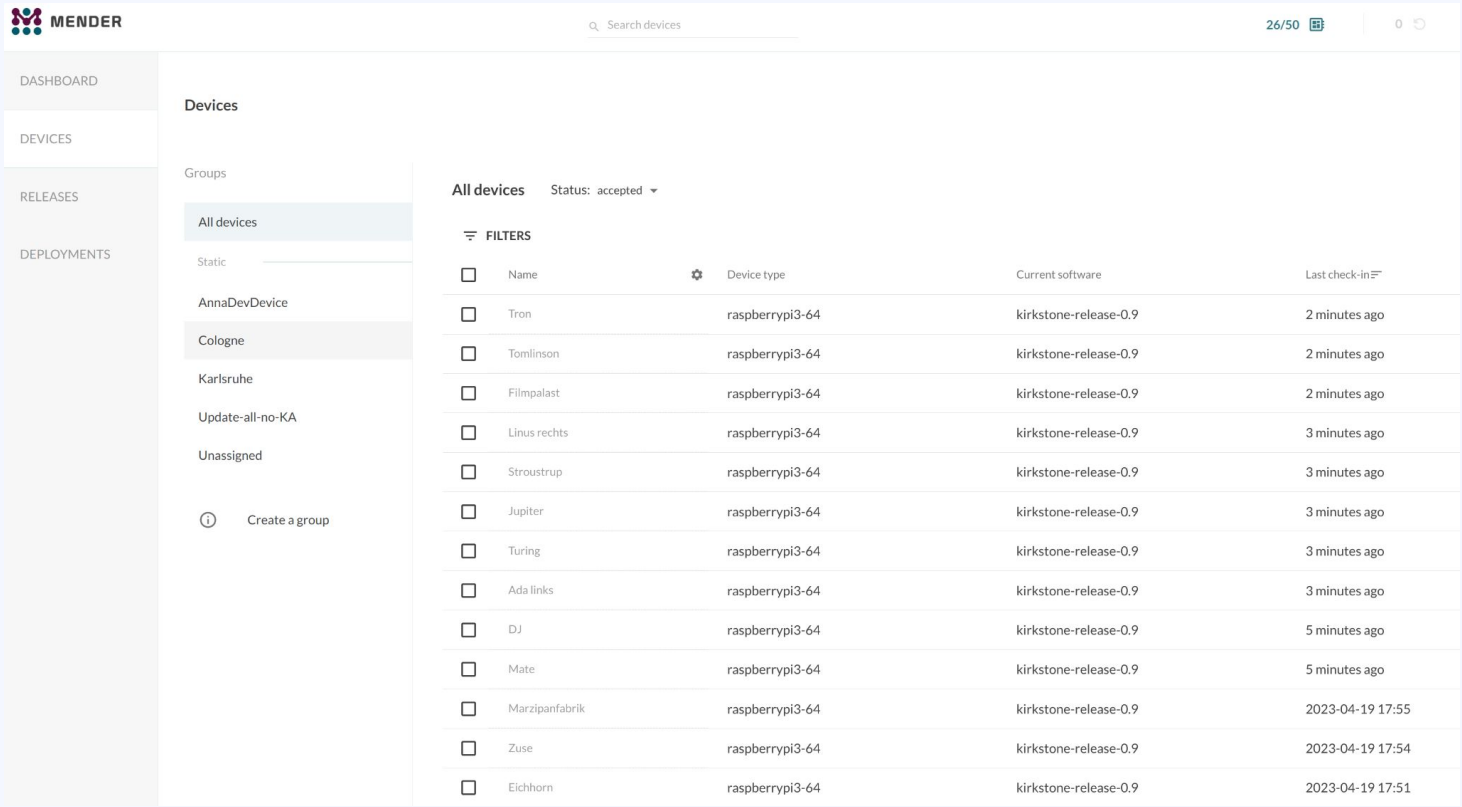
inovex

Container Registry

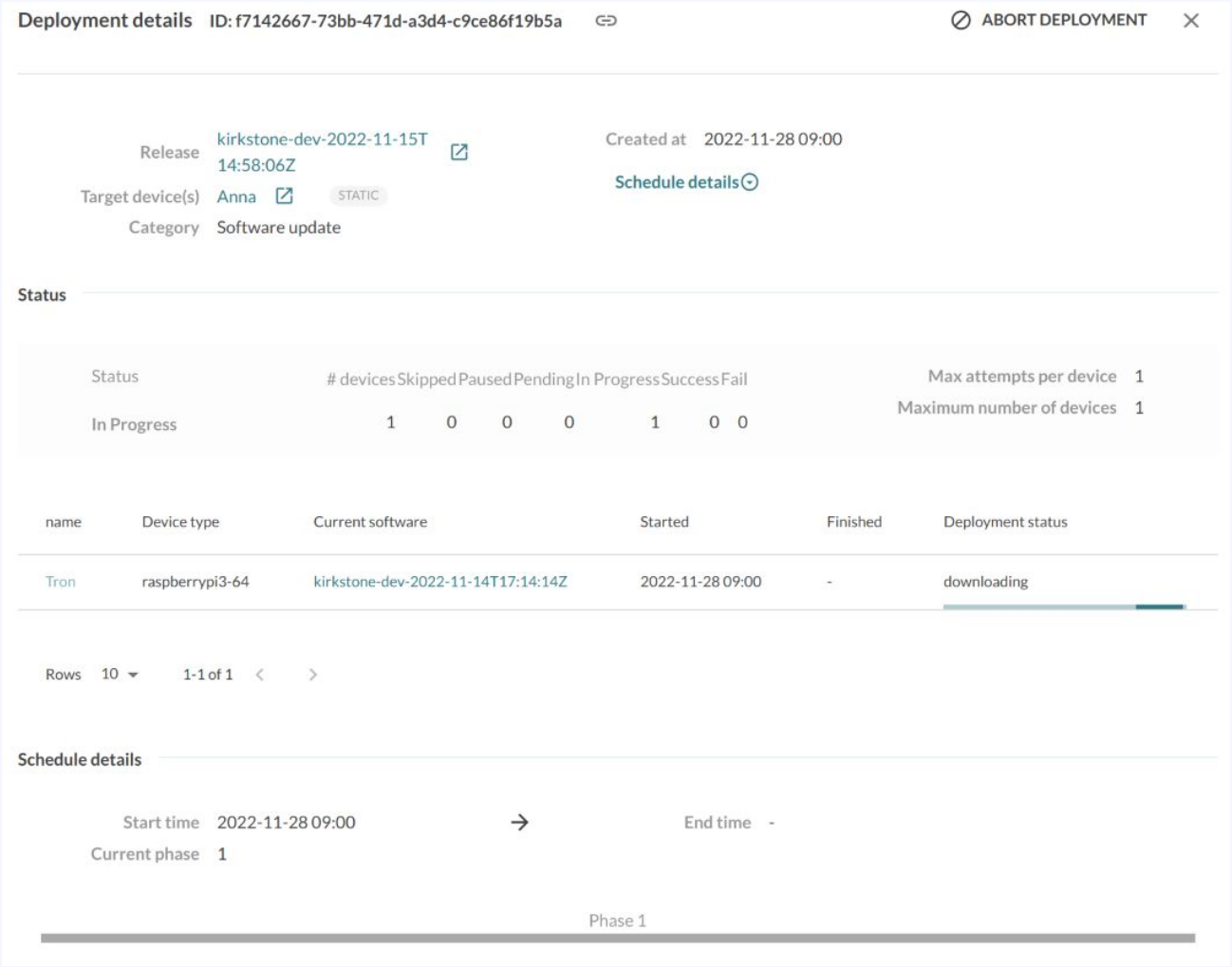


inovex

Mender

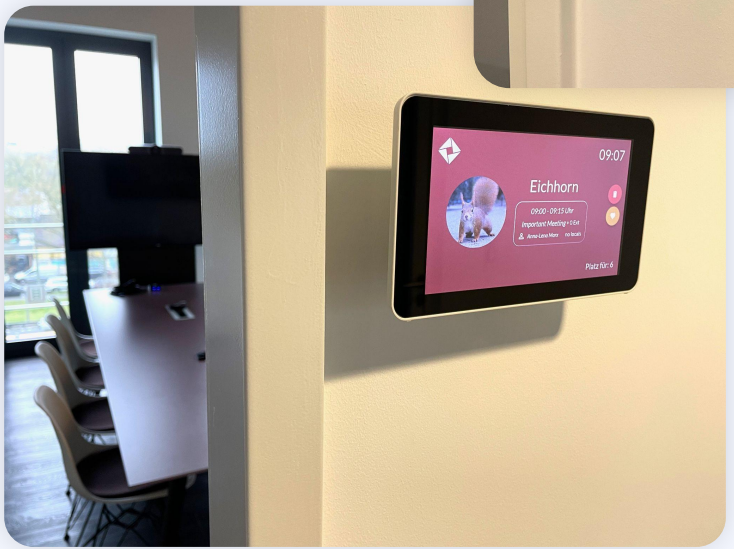


inovex


Deployment details ID: f7142667-73bb-471d-a3d4-c9ce86f19b5a
ABORT DEPLOYMENT
Release kirkstone-dev-2022-11-15T 14:58:06Z
Target device(s) Anna STATIC
Category Software update
Created at 2022-11-28 09:00
Schedule details
Status
Status # devices Skipped Paused Pending In Progress Success Fail
In Progress 1 0 0 0 1 0 0
Max attempts per device 1
Maximum number of devices 1
name Device type Current software Started Finished Deployment status
Tron raspberrypi3-64 kirkstone-dev-2022-11-14T17:14:14Z 2022-11-28 09:00 - downloading
Rows 10 1-1 of 1
Schedule details
Start time 2022-11-28 09:00
End time -
Current phase 1
Phase 1

The result


09:08
Jupiter
Nichts los
Heute ist hier Ruhe
Platz für: 4


09:07
Eichhorn
Platz für: 6


14:48
Jupiter
Platz für: 4

Learnings and recommendations

inovex

What we learned

- Be aware that Google buries a lot of projects
- Have a backup plan if one of your components becomes obsolete or is not longer maintained
- Distribute expertise
  - Beware of a bus factor of 1

Building and maintaining an embedded device is a huge amount of work and involves a wide range of expertise.

What we recommend

- a thoughtful selection of components is key when building embedded or IoT devices with a certain expected lifespan
  - evaluate different options in a useful depth in prior
  - know the pain points!

- check how well a component is maintained and by whom

- setting up a more or less modern Yocto stack is easy
  - keep it modern and well maintained not,
  invest some time regularly!

- keep an eye on security issues and if used versions are affected

Find us, follow us

@inovexgmbh

@inovexlife

inovex

inovex

Thank you!

Anna-Lena Marx

Embedded Systems Dev
inovex GmbH
Ludwig-Erhard-Allee 6
76131 Karlsruhe
anna-lena.marx@inovex.de

inovex