

Easily Generating Debian-Based Embedded Systems

Clara Kowalsky, Felix Moessbauer – Siemens AG

About us



Clara Kowalsky

[<clara.kowalsky@siemens.com>](mailto:clara.kowalsky@siemens.com)

- Siemens Foundational Technology
- (In-house) Embedded Linux consultant & developer
- Contributor to major OSS projects for Siemens (isar, cip-core, Xenomai, OE-Core)
- Trainer for isar and kas

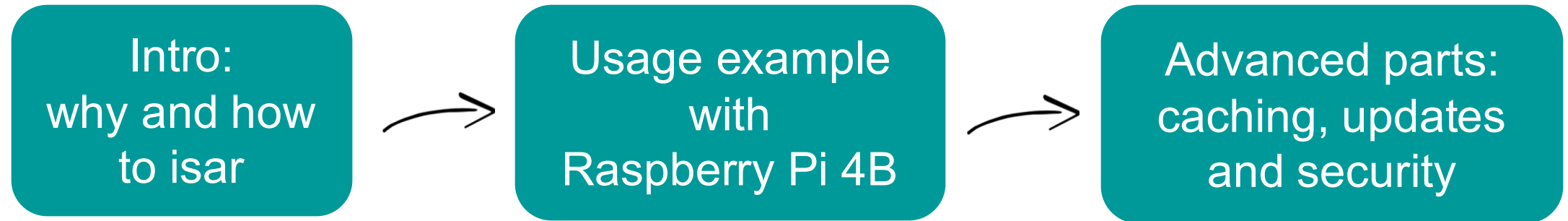


Felix Moessbauer

[<felix.moessbauer@siemens.com>](mailto:felix.moessbauer@siemens.com)

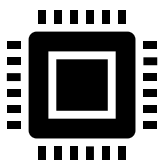
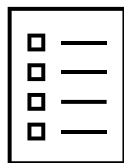
- Siemens Foundational Technology
- (In-house) Embedded Linux consultant & developer
- Contributor to major OSS projects for Siemens (kernel, isar, kas, linux-entra-ss0)

Agenda



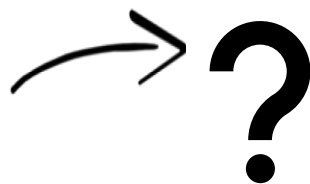
Which Linux for Your Product?

Task: deploy 100
Raspberry Pis...



Requirements:

- Reproducible images
- Custom applications
- Security updates
- Maintainable long-term
- Quickly create test images



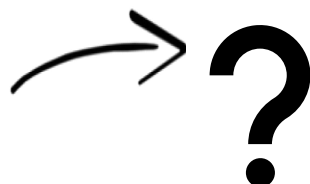
“Roll your own”

yocto PROJECT

openembedded

BuildRoot
Making Embedded Linux Easy

- Highly customizable
- But production times explode with long package lists



Distribution-based



- Reuse standard desktop/ server distribution
- Install pre-built binary packages
- But hardly customizable, larger images & slower boots

Let's take the best of both worlds!

Which Linux for Your Product?

Let's choose Debian!

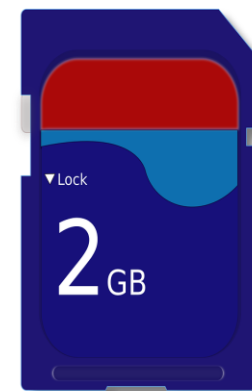
- High-quality mainstream Linux distribution
- Broad hardware support
- Long-term support (10 years with Debian Extended LTS + CIP kernel)
- Free software only *



Missing pieces



- Generator for bootable images
- Package customizations
- Efficient variant management
- A/B updates, Secure Boot



* except for components "non-free" and "non-free-firmware"

isar [1] – Package and Image Build System for Debian



- integration system for automated root filesystem generation
 - Started as OSS 2016 by ilbers and Siemens
- Best of both worlds
 - **Binary** Debian feed
 - Yocto-like recipes, layers, configs (**BitBake**) for deviations and imaging
- Reuse Yocto knowledge of developers

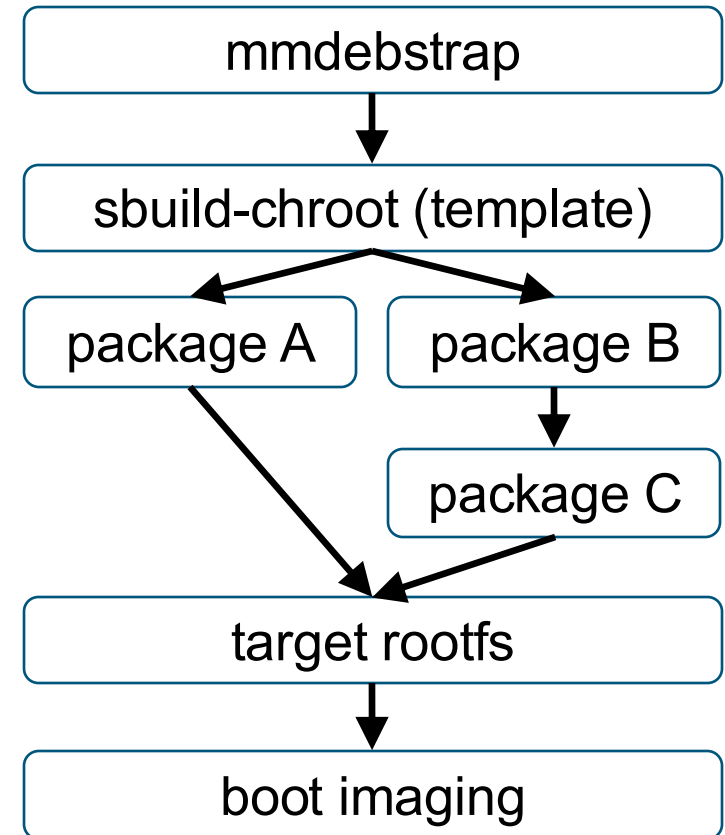


Copyright Schütz, CC BY-SA 3.0

[1] <https://github.com/ilbers/isar>

isar – What is Supported

- Supported architectures and targets (and more...):
 - **x86**: QEMU, standard PC, industrial PC
 - **ARM, ARM64**: QEMU, RPi3, RPi4, BBB, i.MX6 boards
 - **MIPS**: QEMU
 - **RISC-V**: QEMU, SiFive boards
- Supported image types [1] (and more...):
 - Direct SD card writable (wic, ext4, ubi)
 - Container (oci-*, docker-*)
 - Virtual machine (ova, vmdk)
 - Archive/ filesystem (tar, cpio, ubifs, squashfs)

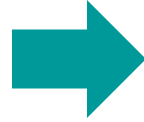


[1] https://github.com/ilbers/isar/blob/master/doc/user_manual.md#image-type-selection

How to Build an Image with isar and kas

Complicated manual way:

1. Fetch repos your project relies on
2. Create bblayer.conf and local.conf
3. Initialize the build environment
4. Build: bitbake my-target



Easier: Use kas [1] instead!

One command to rule them all:

`kas-container build kas.yaml`

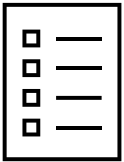
Key benefits of kas:

- Zero setup hassle
- Reproducible environment
- Version control friendly
- CI/CD ready

[1] <https://github.com/siemens/kas>

How to Build an Image with isar and kas – The Plan

Task: deploy 100
Raspberry Pis...



Let's choose
isar + kas

isar
build easily



Option 1:
Configure image
directly in isar via kas
menu

Option 2:
Create your own layer
and configure image
there

Make image
customizations:
e.g., add packages

- Pre-debianized packages
- Ad-hoc debianized packages from isar recipes
- Ad-hoc debianized packages from own recipes, e.g., custom C application

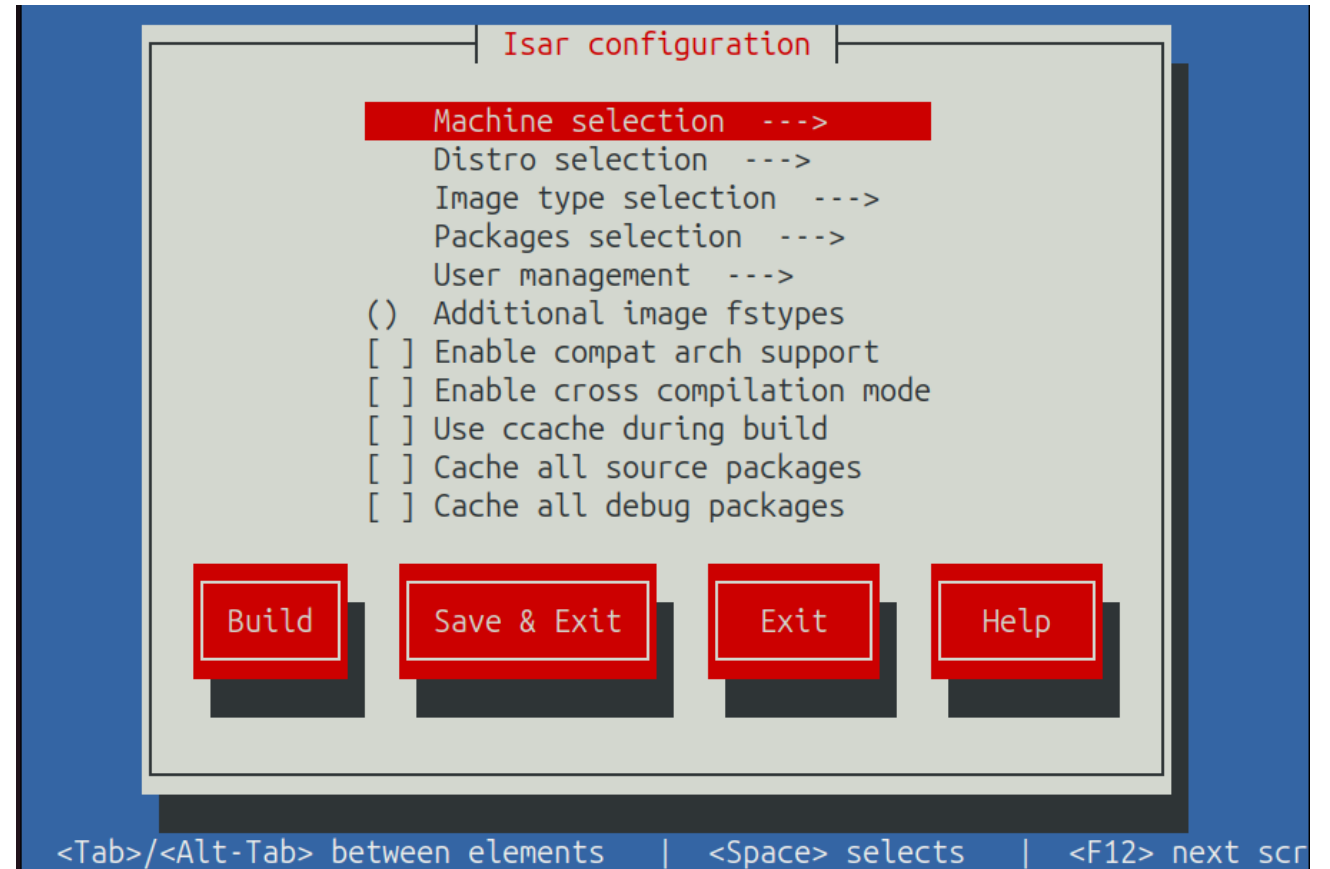


Build image &
bring it to device



Option 1: Build Image for Raspberry Pi 4B in isar - with kas menu

- Clone isar [1]
- Enable privileged docker or podman
- Run [2]: `./kas-container menu`



[1] <https://github.com/ilbers/isar>

[2] <https://github.com/siemens/kas/blob/master/kas-container>

Option 1: Build Image for Raspberry Pi 4B in isar - with kas menu

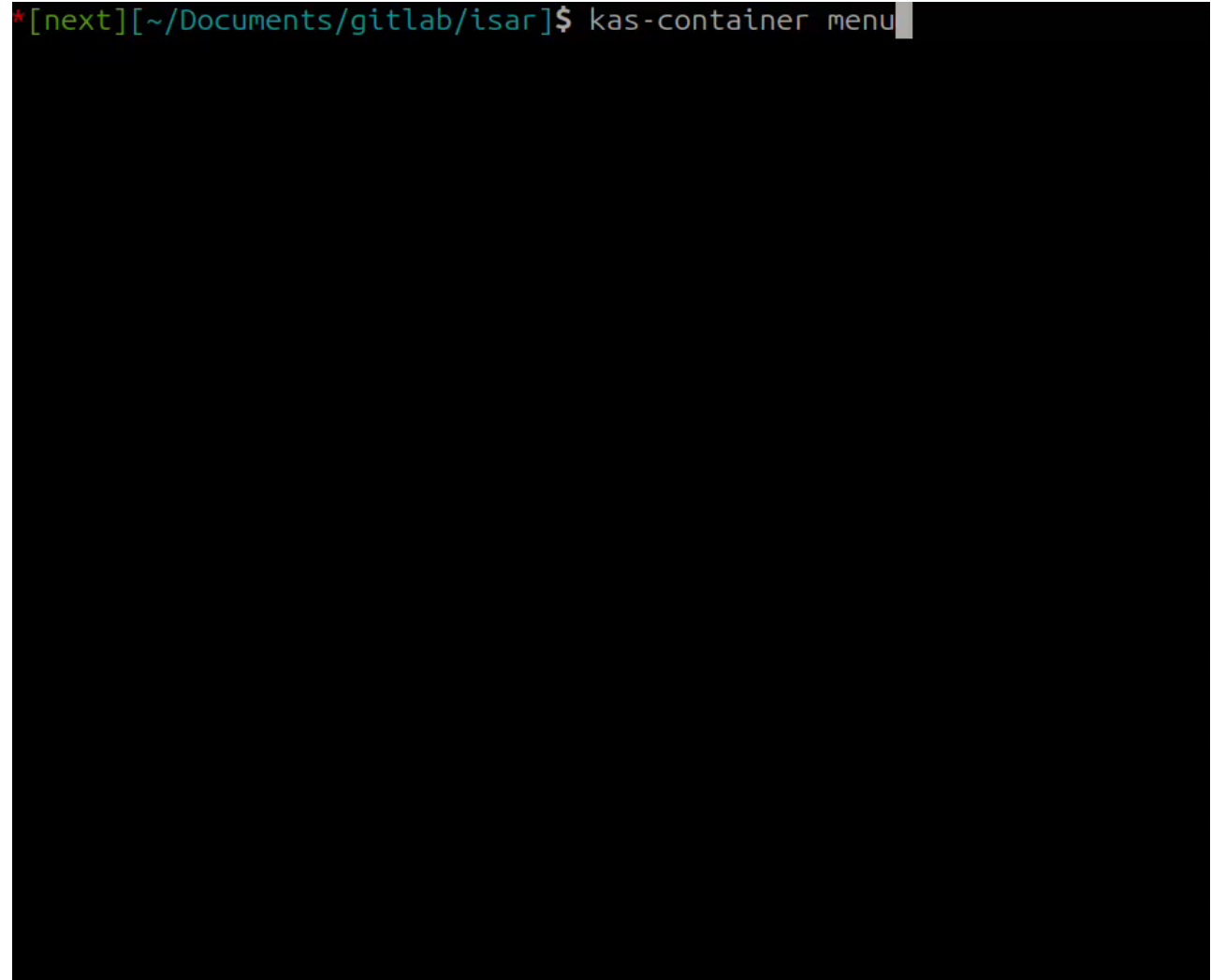
Select machine:

- rpi-arm64-v8:
 - Only use the RPi proprietary bootloader
 - Distro Raspberry Pi OS 11/12
- rpi-arm64-v8-efi:
 - RPi proprietary bootloader + U-Boot as EFI provider
 - Distro Debian bookworm

Select package:

- hello-isar [1]

[1] <https://github.com/ilbers/isar/blob/master/meta-isar/recipes-app/hello-isar/hello-isar.bb>



```
*[next][~/Documents/gitlab/isar]$ kas-container menu
```

Boot the Raspberry Pi 4B

- Output image: build/tmp/deploy/images/rpi-arm64-v8*/*.wic
- Flash it to an SD card (e.g., with balenaEtcher [1] or with bmaptool)

```
Debian GNU/Linux 12 localhost ttyS1

localhost login: root
Password:
Linux localhost 6.1.0-37-arm64 #1 SMP Debian 6.1.140-1 (2025-05-22) aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu May 29 17:04:13 UTC 2025 on ttyS1
root@localhost:~# hello-isar
Hello, world!
```



[1] <https://etcher.balena.io/>

Boot the Raspberry Pi 4B - Issues

Issues with booting the rpi-arm64-v8-efi image?

 Maybe your RPi bootloader is too old

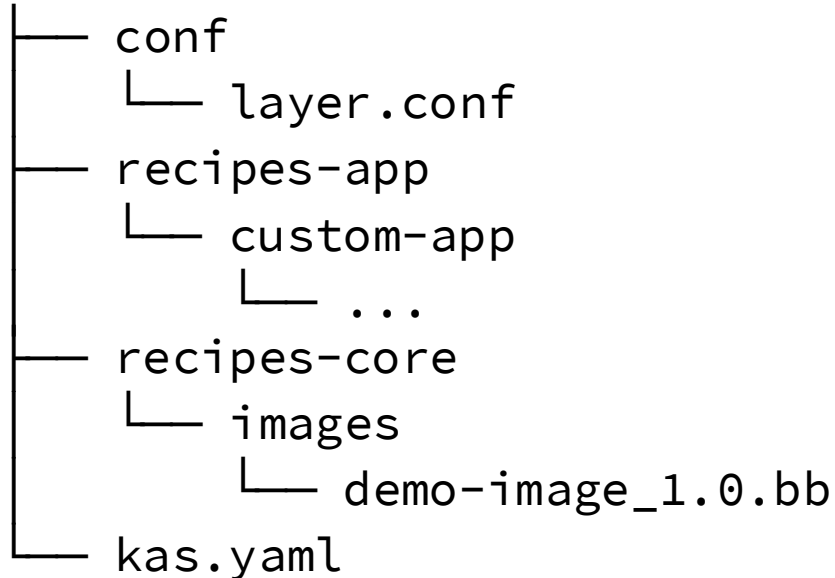
- EFI boot support for RPi 4B was introduced with bootloader update 2020-09-03 [1]
- Update the bootloader:
 - Boot an image with Raspberry Pi OS first (machine rpi-arm64-v8)
 - Update the bootloader: `rpi-eeeprom-update -a; reboot`
 - Boot the rpi-arm64-v8-efi image

[1] <https://github.com/raspberrypi/rpi-eeeprom/releases/tag/v2020.09.03-138a1>

Option 2: Build Image for Raspberry Pi 4B - In Your Own Layer

Minimal open source demo project [1]:

meta-demo



[1] <https://github.com/siemens/isar-demo>

[2] <https://github.com/ilbers/isar/blob/master/meta-isar/conf/machine/rpi-arm64-v8-efi.conf>

[3] <https://github.com/ilbers/isar/blob/master/meta/conf/distro/debian-bookworm.conf>

```
# cat kas.yaml
```

```
header:
  version: 17
  build_system: isar

machine: rpi-arm64-v8-efi
distro: debian-bookworm
target: demo-image

repos:
  meta-demo:
    isar:
      url: https://github.com/ilbers/isar
      commit: 0f1df8f...
      layers:
        meta:
        meta-isar:

local_conf_header:
  image-format: |
    IMAGE_FSTYPES = "wic wic.xz"
```

Same machine as
before [2]

Same distro as
before [3]

Option 2: Build Image for Raspberry Pi 4B - In Your Own Layer

```
# cat recipes-core/images/demo-image_1.0.bb
require recipes-core/images/isar-image-base.bb

# isar built package
IMAGE_INSTALL += "custom-app"

# package from Debian upstream
IMAGE_PREINSTALL += "vim"

USERS += "root"
USER_root[flags] = "clear-text-password"
USER_root[password] = "root"
```

Same base image
as before [1]

Our custom
package

[1] <https://github.com/ilbers/isar/blob/master/meta-isar/recipes-core/images/isar-image-base.bb>

Option 2: Build Image for Raspberry Pi 4B - In Your Own Layer

Minimal open source demo project:

```
meta-demo
├── conf
│   └── layer.conf
├── recipes-app
│   └── custom-app
│       ├── custom-app-1.0
│       │   ├── main.c
│       │   └── Makefile
│       └── custom-app_1.0.bb
├── recipes-core
│   └── images
│       └── demo-image_1.0.bb
└── kas.yaml
```

Let's debianize some sources
written in C with isar

```
# cat recipes-app/custom-app/custom-app-1.0/main.c

#include <stdio.h>

int main() {
    printf("This is my custom application!\n");
    return 0;
}
```

Option 2: Build Image for Raspberry Pi 4B - In Your Own Layer

```
# cat recipes-app/custom-app/custom-app_1.0.bb
```

Recipe for our
custom package



```
SRC_URI="file://${PN}-${PV}"

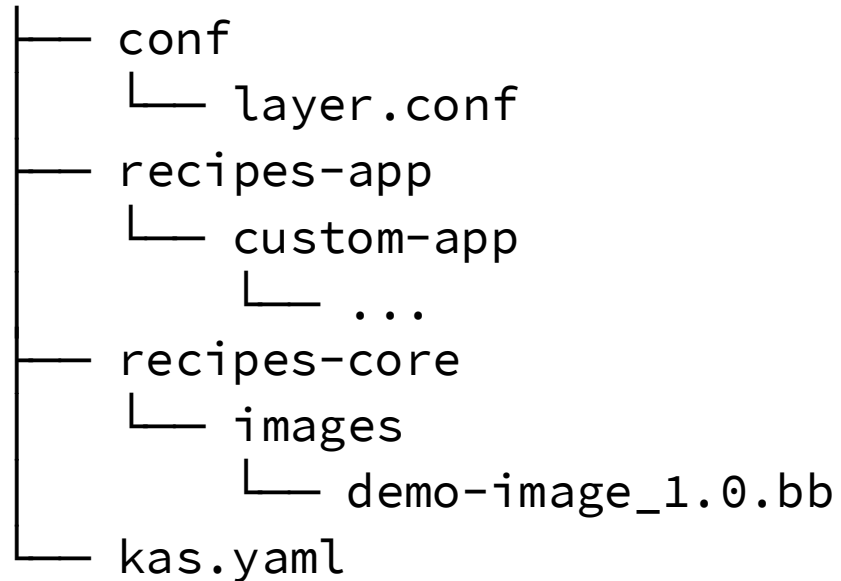
inherit dpkg

do_prepare_build[cleandirs] += "${S}/debian"
do_prepare_build() {
    deb_debianize
}
```

Option 2: Build Image for Raspberry Pi 4B - In Your Own Layer

Minimal open source demo project:

meta-demo



cat layer.conf

BitBake
boilerplate

```
BBPATH .= ":${LAYERDIR}"
BBFILES += "${LAYERDIR}/recipes-*/*/*.bb \
           ${LAYERDIR}/recipes-*/*/*.bbappend"
BBFILE_COLLECTIONS += "meta-demo"
BBFILE_PATTERN_meta-demo = \
    "^${LAYERDIR}/recipes-"
BBFILE_PRIORITY_meta-demo = "7"
LAYERVERSION_meta-demo = "3"

LAYERSERIES_COMPAT_meta-demo = "next"
LAYERDIR_meta-demo = "${LAYERDIR}"
```

Option 2: Build Image for Raspberry Pi 4B - In Your Own Layer

Trigger the build: **`kas-container build kas.yaml`**

GitLab CI integration:

```
# cat .gitlab-ci.yml
```

```
image: ghcr.io/siemens/kas/kas-isar:4.8.2

# define proxy config if needed

demo-image:
  script:
    - kas build kas.yaml
  artifacts:
    paths:
      - build/tmp/deploy/images/*/*.wic.xz
      - build/tmp/deploy/images/*/*.wic.bmap
      - build/tmp/deploy/images/*/*.manifest
```

Option 2: Build Image for Raspberry Pi 4B - In Your Own Layer

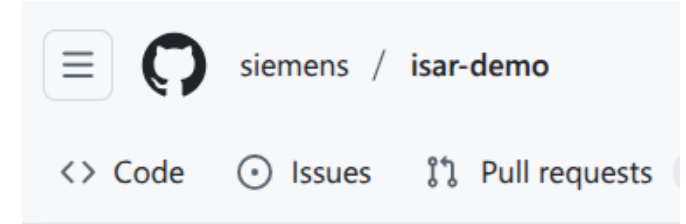
GitHub CI integration:

```
# cat .github/workflows/build.yaml
```

```
name: build
on: [push]

jobs:
  demo-image:
    name: demo-image
    runs-on: ubuntu-24.04
    steps:
      - name: Checkout
        uses: actions/checkout@v4
      - name: Build image
        run: ./kas-container build kas.yaml
      - name: Upload image
        uses: actions/upload-artifact@v4
    ...
```

```
...
        with:
          name: demo-image
          path: |
            build/tmp/deploy/images/*/*.wic.xz
            build/tmp/deploy/images/*/*.wic.bmap
            build/tmp/deploy/images/*/*.manifest
```



← Back to pull request #1

✓ chore: add GitHub Actions

Summary

Jobs

✓ demo-image

Boot the Raspberry Pi 4B

```
root@localhost:~# which vim
/usr/bin/vim
root@localhost:~# custom-app
This is my custom application!
```

vim and the custom package are correctly installed



The isar Ecosystem

Civil Infrastructure Platform

- Secure boot (OP-TEE and EFI)
- OTA firmware updates
- Reproducible builds (bit identical images)



Yocto & OpenEmbedded

- Fast (partial) rebuilds by using Yocto's powerful SState infrastructure
- Contributing isar features back to Yocto and BitBake



Debian

- Supports Debian snapshot mirrors (now behind a CDN)
`ISAR_USE_APT_SNAPSHOT=1`
`ISAR_APT_SNAPSHOT_DATE=<...>`
- SBOM generation for rootfs
- Huge set of well-maintained packages



Future Plans

isar: Usability & Maintenance

- Support rootless builds (currently privileges are needed, e.g., for mknod)
- SBOM: support CycloneDX and SPDX 
- initrd: add Dracut support
- Build time reduction via cross installation

kas

- Add Debian trixie container 
- Renovate / Dependabot support
- (Yocto) buildtools support
- Repo tool to kas yaml converter 
- Improve "Getting Started" guide

Overview of Related Talks



<https://sched.co/1LcOs>

Building Products with Debian and Isar

Jan Kiszka
Baurzhan Ismagulov

Open Source Summit Japan 2019

<https://sched.co/OVsN>



<https://sched.co/1aBFE>



<https://sie.ag/2bsA3L>



<https://sched.co/c3Wx>

Community

isar

- Homepage & code: <https://github.com/ilbers/isar>
- Discussions and patches via isar-users@googlegroups.com
- Demo project: <https://github.com/siemens/isar-demo>
- Public CI: <https://gitlab.isar-build.org/ilbers/isar/-/pipelines>
- Public users
 - <https://gitlab.com/Xenomai/xenomai-images>
 - <https://gitlab.com/cip-project/cip-core/isar-cip-core>
 - <https://github.com/siemens/meta-iot2050>

kas

- Homepage & code: <https://github.com/siemens/kas>
- Discussions and patches via kas-devel@googlegroups.com

16:20
CEST



Yocto or Debian for Your Embedded System? Yes. - Alan Martinovic, Northern.tech
G104 (Level 1)



Civil Infrastructure Platform: State of Industrial Grade Linux - Yoshitake Kobayashi & Dinesh Kumar, Toshiba Corporation
G107 (Level 1)

Community Meetup

Public isar community meetup

- Save the date: December 8, 2025
- Location: Design Offices Nuremberg Central Station
- Details will be published on the isar mailing list and on <https://opensource.siemens.com/events/>

Thank you!

Time for questions.



Clara Kowalsky

clara.kowalsky@siemens.com

[linkedin.com/in/clara-kowalsky](https://www.linkedin.com/in/clara-kowalsky)



Felix Moessbauer

felix.moessbauer@siemens.com

[linkedin.com/in/felixmoessbauer](https://www.linkedin.com/in/felixmoessbauer)

