THE LINUX FOUNDATION PROJECTS

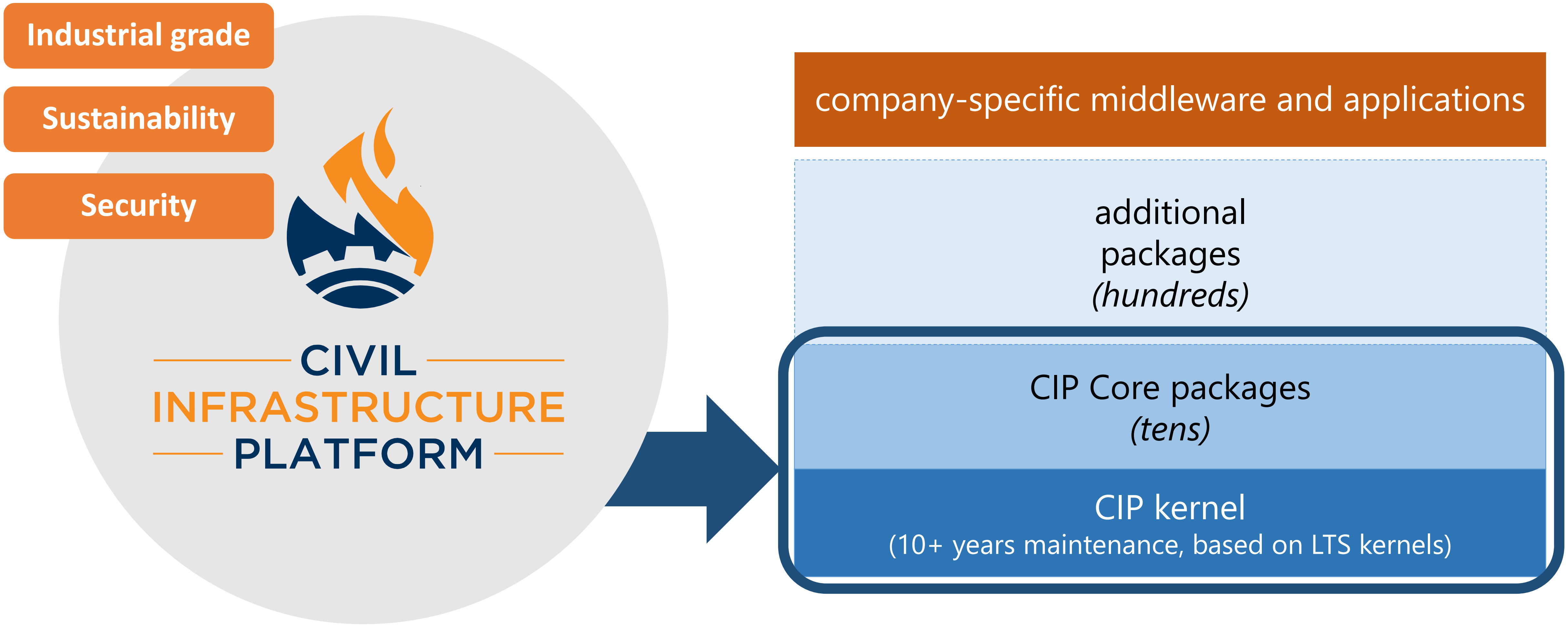
CIVIL INFRASTRUCTURE PLATFORM

CIP Mini Summit: Introduction of CIP Core

Kazuhiro Hayashi, Toshiba Corporation
Jan Kiszka, Siemens AG

Embedded Linux Conference Europe 2019, Oct. 31th 2019

Introduction



Industrial grade
Sustainability
Security
CIVIL INFRASTRUCTURE PLATFORM
company-specific middleware and applications
additional packages *(hundreds)*
CIP Core packages *(tens)*
CIP kernel
(10+ years maintenance, based on LTS kernels)

Establishes an "Open Source Base Layer (OSBL)"

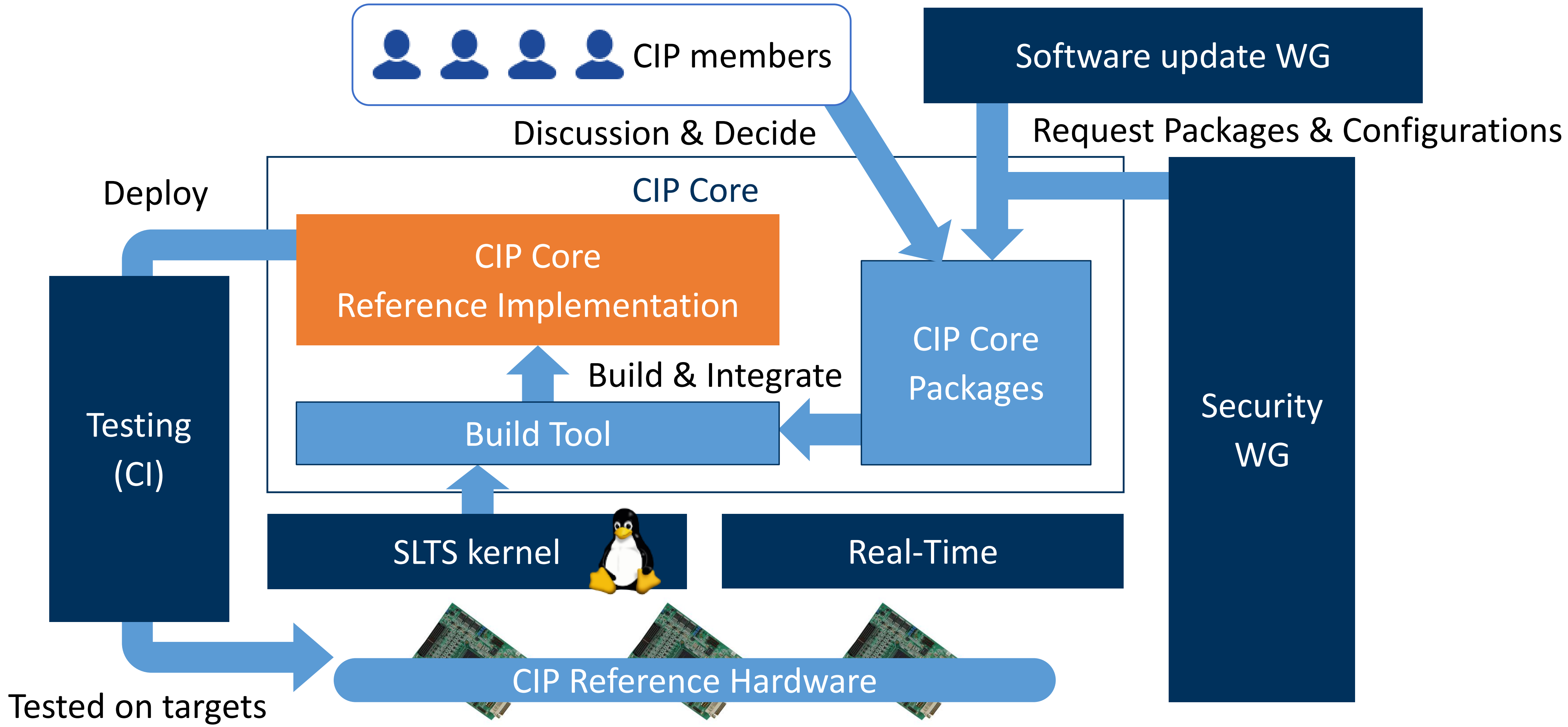
CIP Core

- One of the CIP projects focusing on user land software and tools
- Goals
  - Define a list of “CIP Core packages” maintained for long-term
  - Provide a reference implementation including “CIP Core packages”
  - Test the implementation on the “CIP reference hardware”

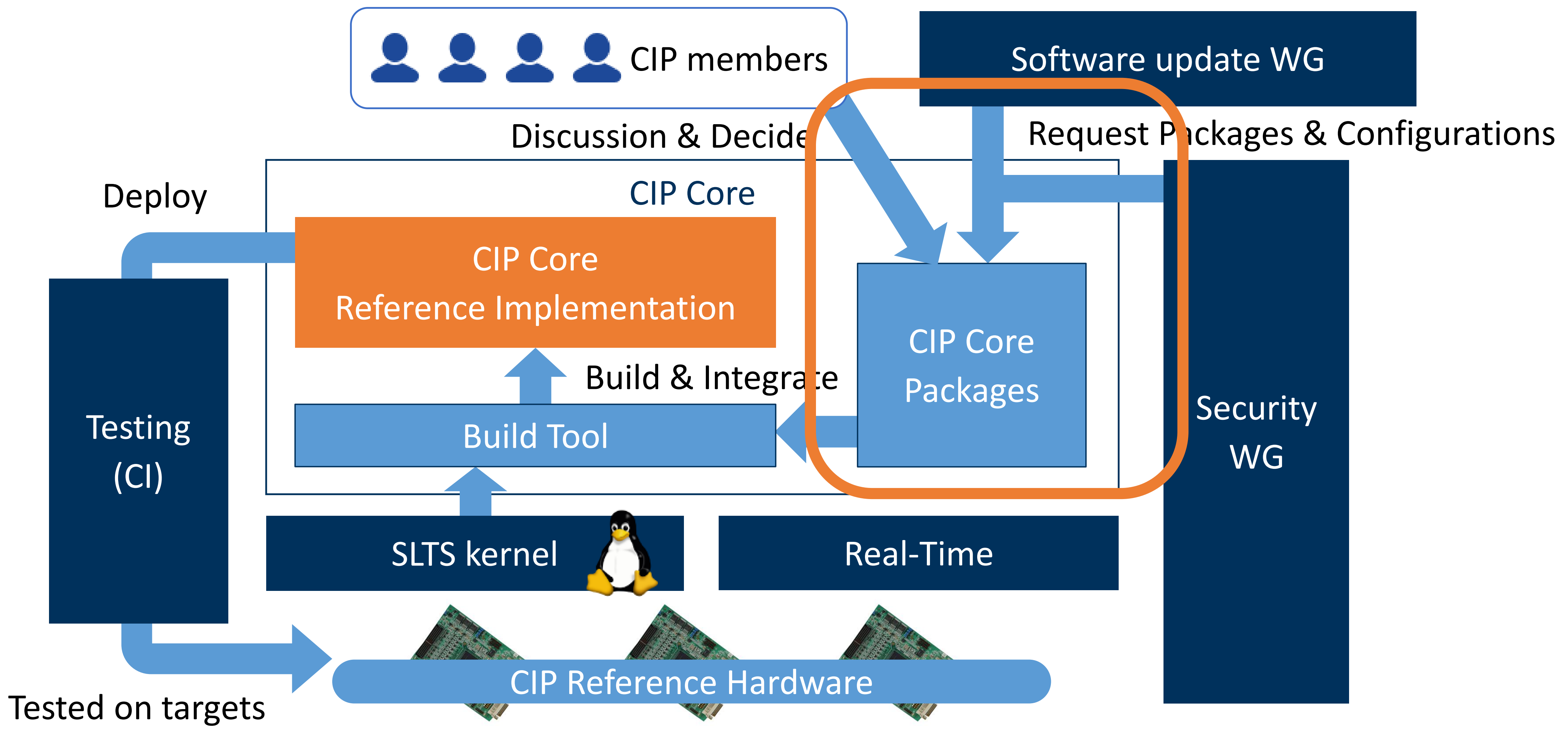
1 SLTS kernel	2 Real-time	3 Testing	4 CIP Core	5 Security WG(*)	6 Software update WG	(*): Workgroup
✔	✔	✔	✔	✔	✔	Industrial grade
✔		✔	✔		✔	Sustainability
✔		✔	✔	✔	✔	Security

CIP Projects and its scopes

CIP Core: Position in CIP Projects

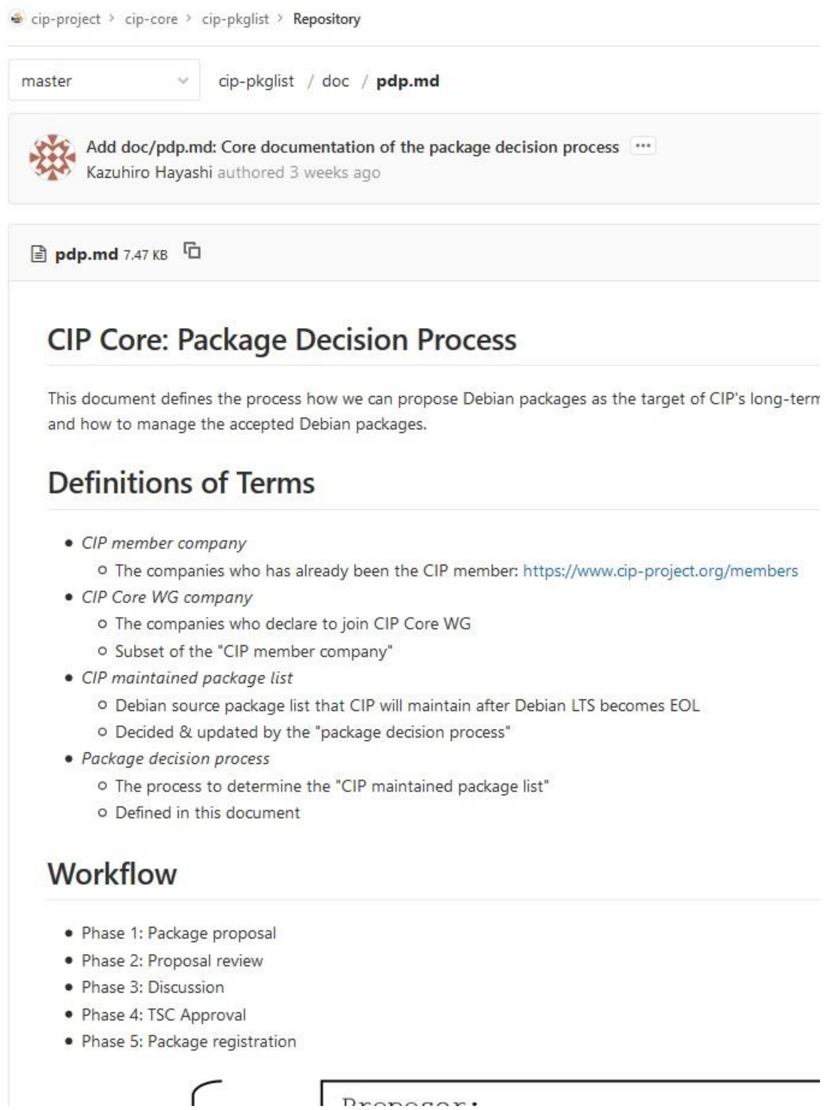


CIP Core: Position in CIP Projects



CIP Core Package List

- Package Decision Process
  - https://gitlab.com/cip-project/cip-core/cip-pkglist
- Components
  - Document for proposal, review, package list update
  - Helper scripts for proposal and updating the package list



cip-project › cip-core › cip-pkglist › Repository

master | cip-pkglist / doc / **pdp.md**

Add doc/pdp.md: Core documentation of the package decision process
Kazuhiro Hayashi authored 3 weeks ago

pdp.md 7.47 KB

CIP Core: Package Decision Process

This document defines the process how we can propose Debian packages as the target of CIP's long-term
and how to manage the accepted Debian packages.

Definitions of Terms

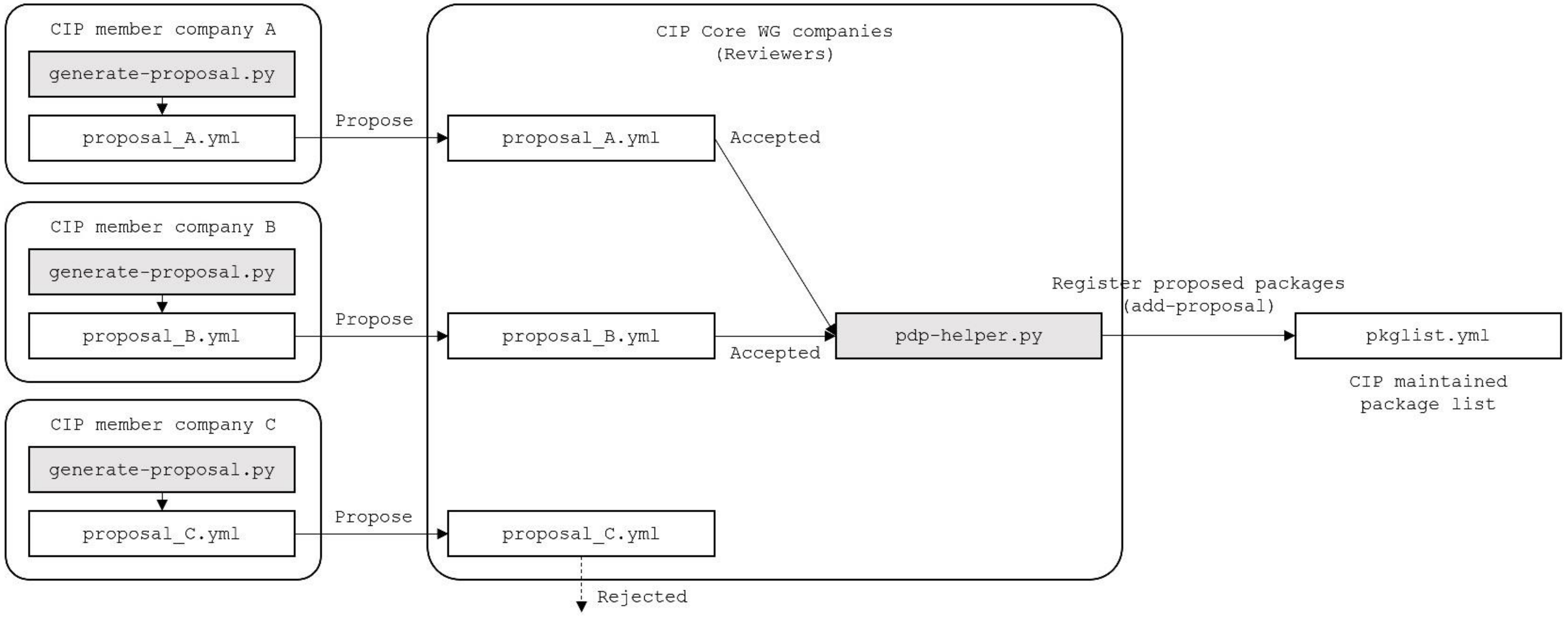
- *CIP member company*
  - The companies who has already been the CIP member: https://www.cip-project.org/members
- *CIP Core WG company*
  - The companies who declare to join CIP Core WG
  - Subset of the "CIP member company"
- *CIP maintained package list*
  - Debian source package list that CIP will maintain after Debian LTS becomes EOL
  - Decided & updated by the "package decision process"
- *Package decision process*
  - The process to determine the "CIP maintained package list"
  - Defined in this document

Workflow

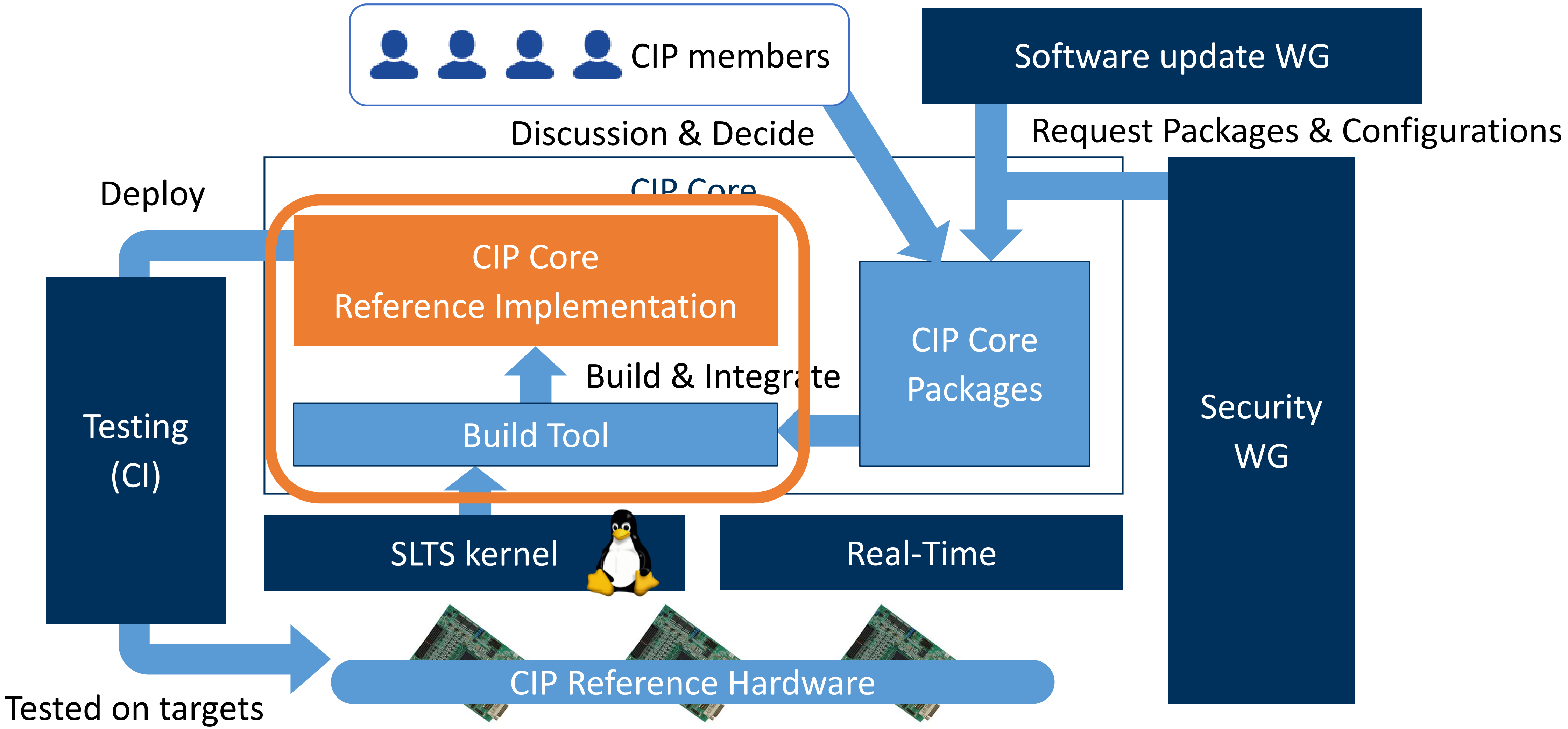
- Phase 1: Package proposal
- Phase 2: Proposal review
- Phase 3: Discussion
- Phase 4: TSC Approval
- Phase 5: Package registration

CIP Core Package List

- Workflow



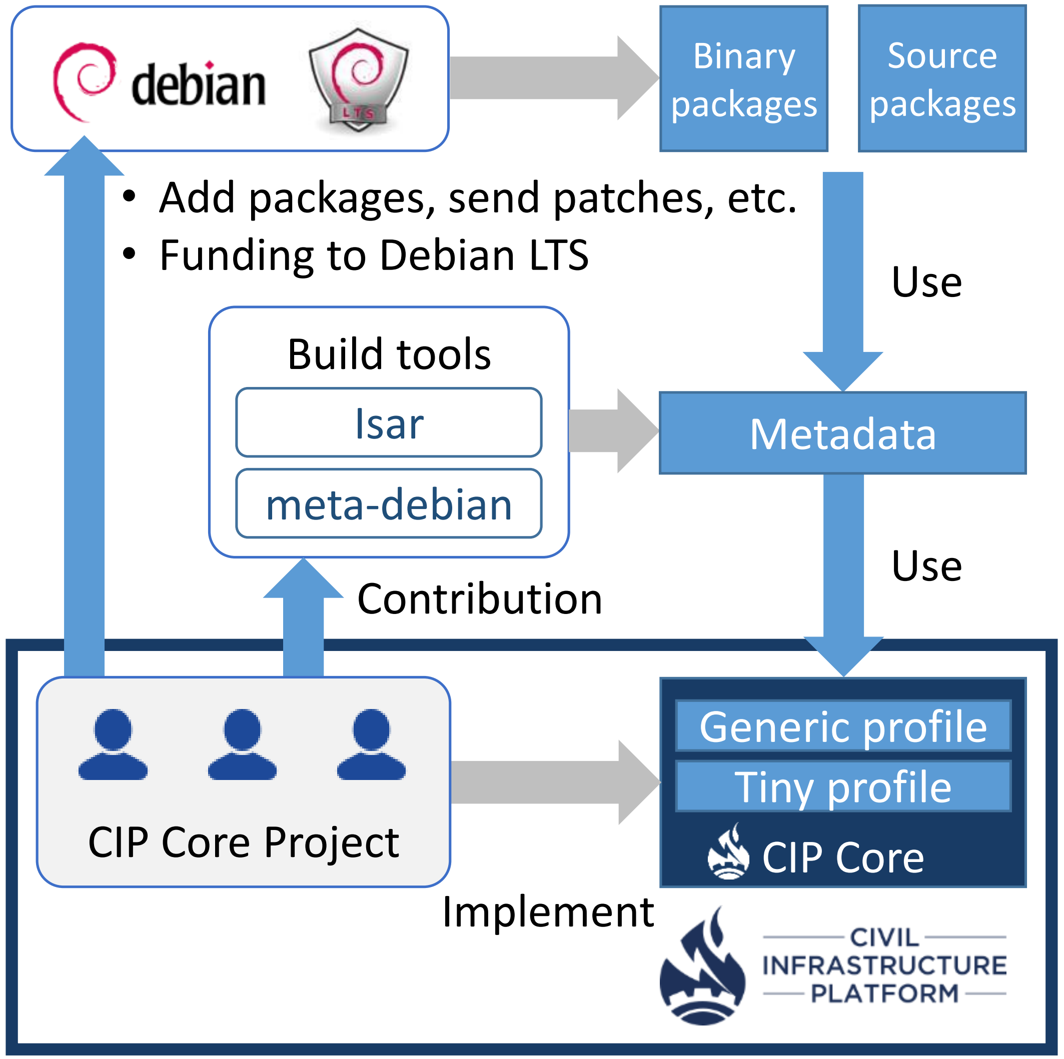
CIP Core: Position in CIP Projects



CIP Core: Implementation

- Debian-based implementation
  - Mature, high-quality, mainstream distro.
  - Many new & old architecture supports
  - Suitable for small and big installations
  - Security updates
- Profiles

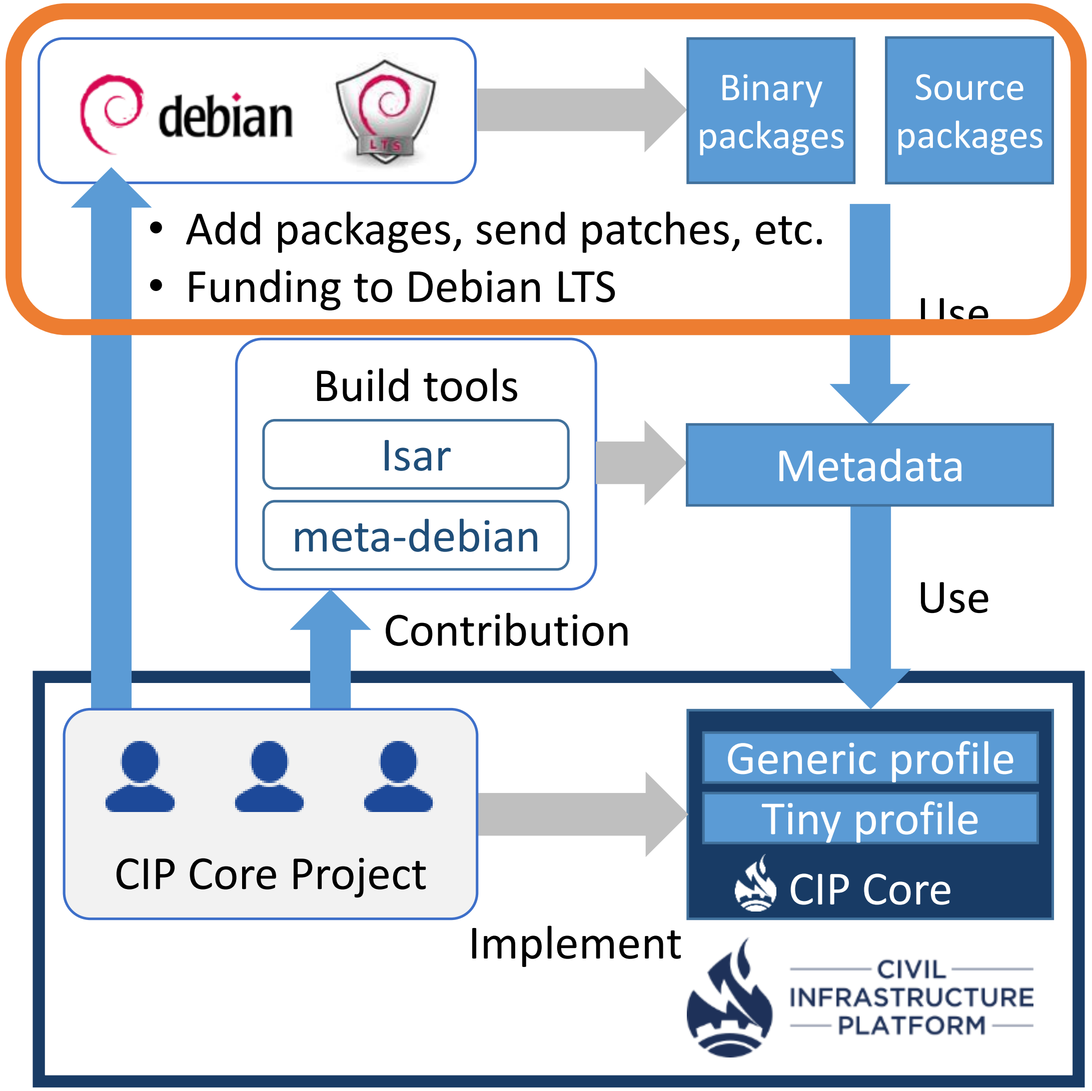
	Generic profile	Tiny profile
Approach	Binary packages	Source packages
Tool	Isar	Deby



CIP Core: Implementation

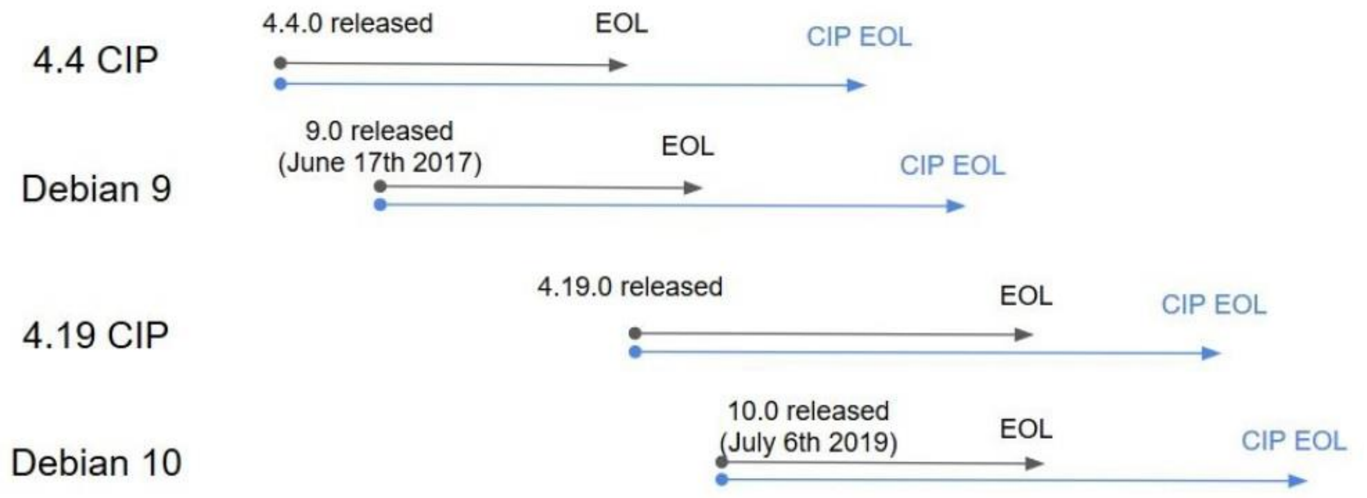
- Debian-based implementation
  - Mature, high-quality, mainstream distro.
  - Many new & old architecture supports
  - Suitable for small and big installations
  - Security updates
- Profiles

	Generic profile	Tiny profile
Approach	Binary packages	Source packages
Tool	Isar	Deby



Target Versions of CIP Core

- CIP kernel & Debian (&Yocto in Tiny profile)



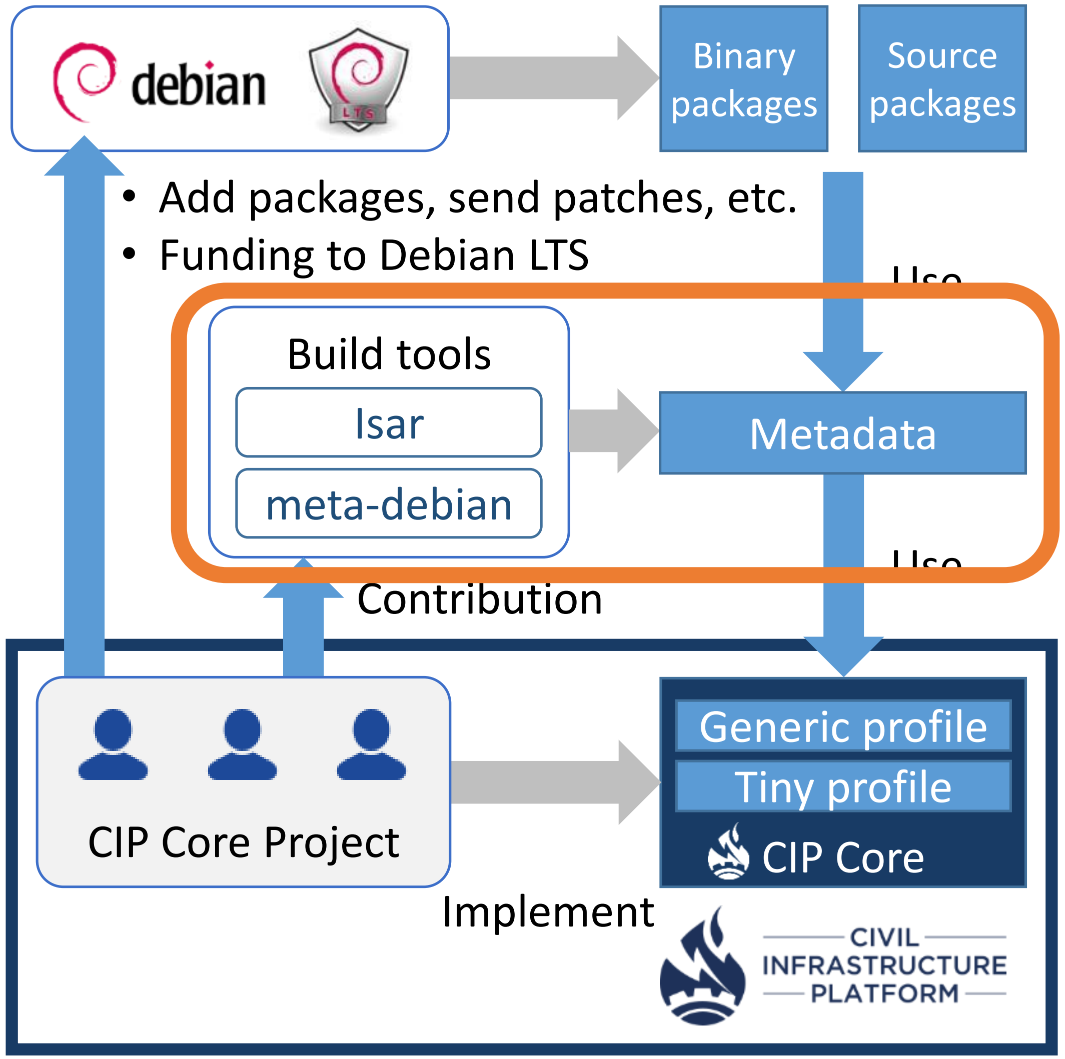
- Current state

	Debian 9 stretch	Debian 10 buster
CIP kernel 4.4	Generic	Unsupported
CIP kernel 4.19	Generic	Generic, Tiny

CIP Core: Implementation

- Debian-based implementation
  - Mature, high-quality, mainstream distro.
  - Many new & old architecture supports
  - Suitable for small and big installations
  - Security updates
- Profiles

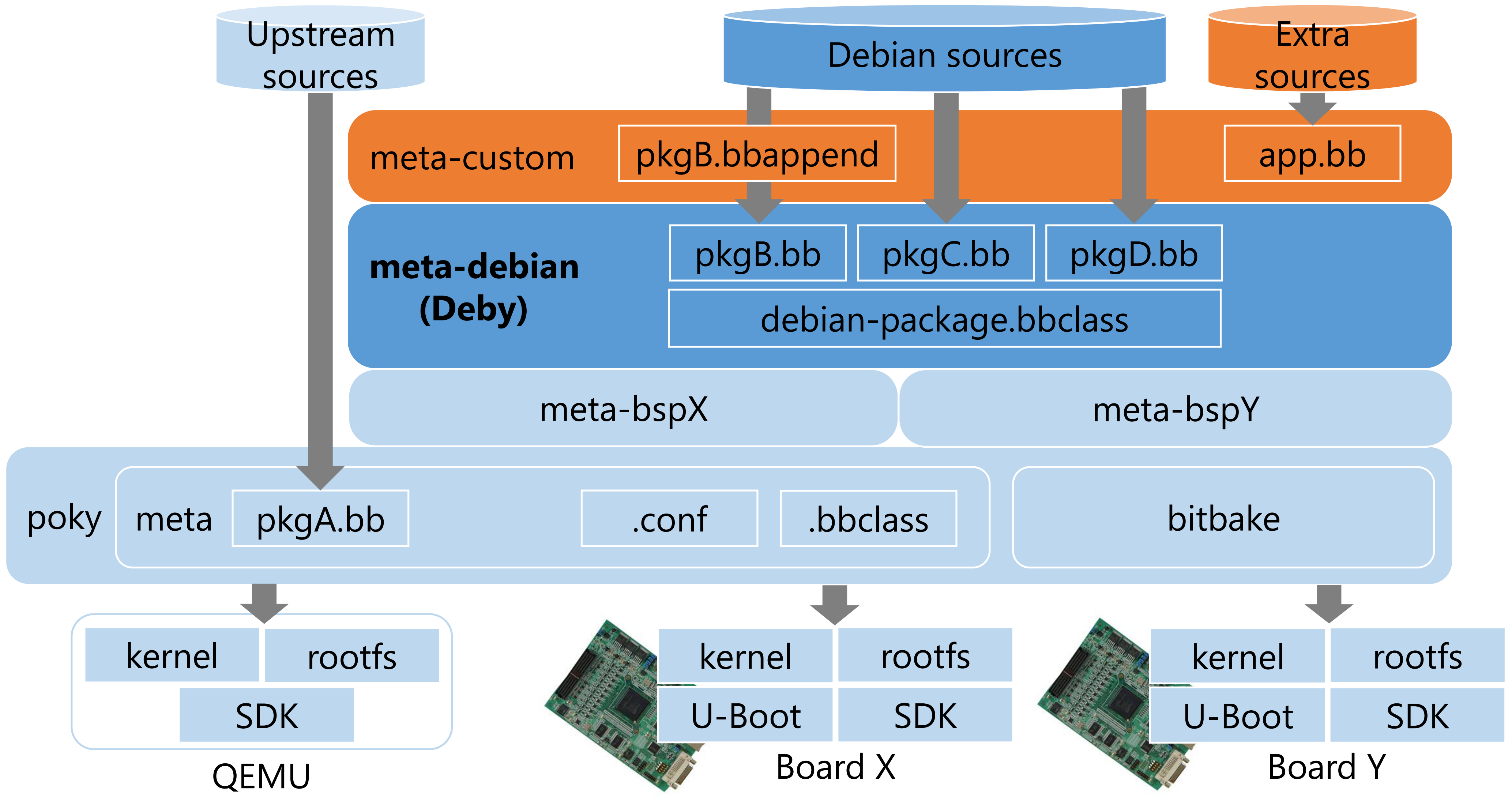
	Generic profile	Tiny profile
Approach	Binary packages	Source packages
Tool	Isar	Deby



Deby (meta-debian)

- Yocto Project extension for using Debian source packages
  - Source code: Debian
  - Build system: Yocto Project
  - https://github.com/meta-debian/meta-debian
- Goals
  - Achieve stability and long-term support with the Yocto advantages
- Features: Yocto based flexibility & extensibility
  - High customizability by own recipes
  - Small footprint (Around 2MB)
  - Various target CPUs and tunings
  - Adaptation to BSP layers provided by board vendors

Deby: How it works

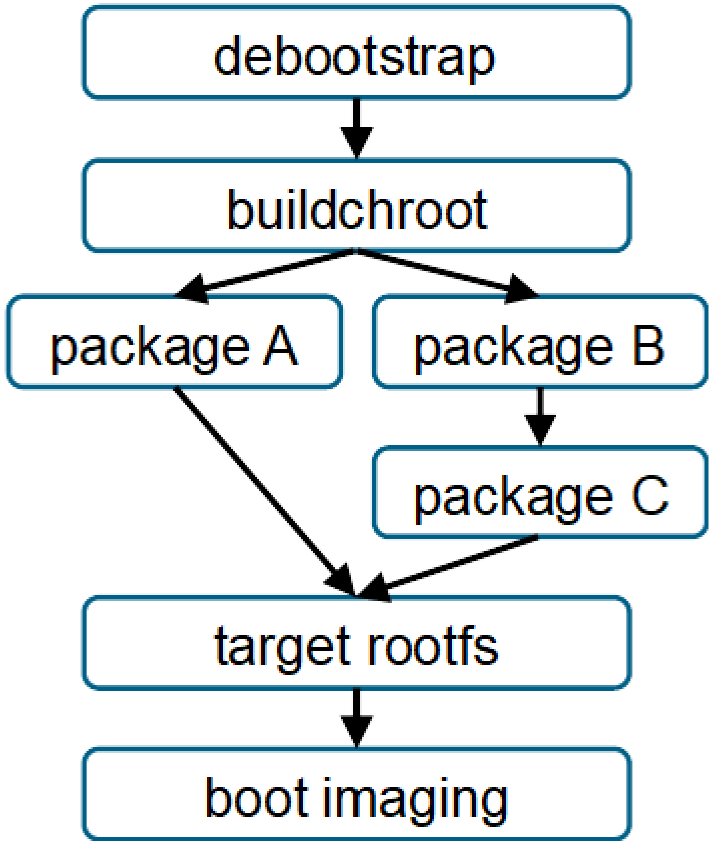


Isar

- **I**ntegration **S**ystem for **A**utomated **R**oot filesystem generation
  https://github.com/ilbers/isar
- Goals
  - Build systems in a Debian way
  - Developer-centric workflow: One-command building
  - Make customizations easy and repeatable
  - Efficient building
- The best of both worlds
  - Debian: Tested binary packages, tools, security updates
  - OpenEmbedded / Yocto: bitbake, recipes, layers
- Reuse Yocto knowledge of developers

Image Generation Sequence of Isar

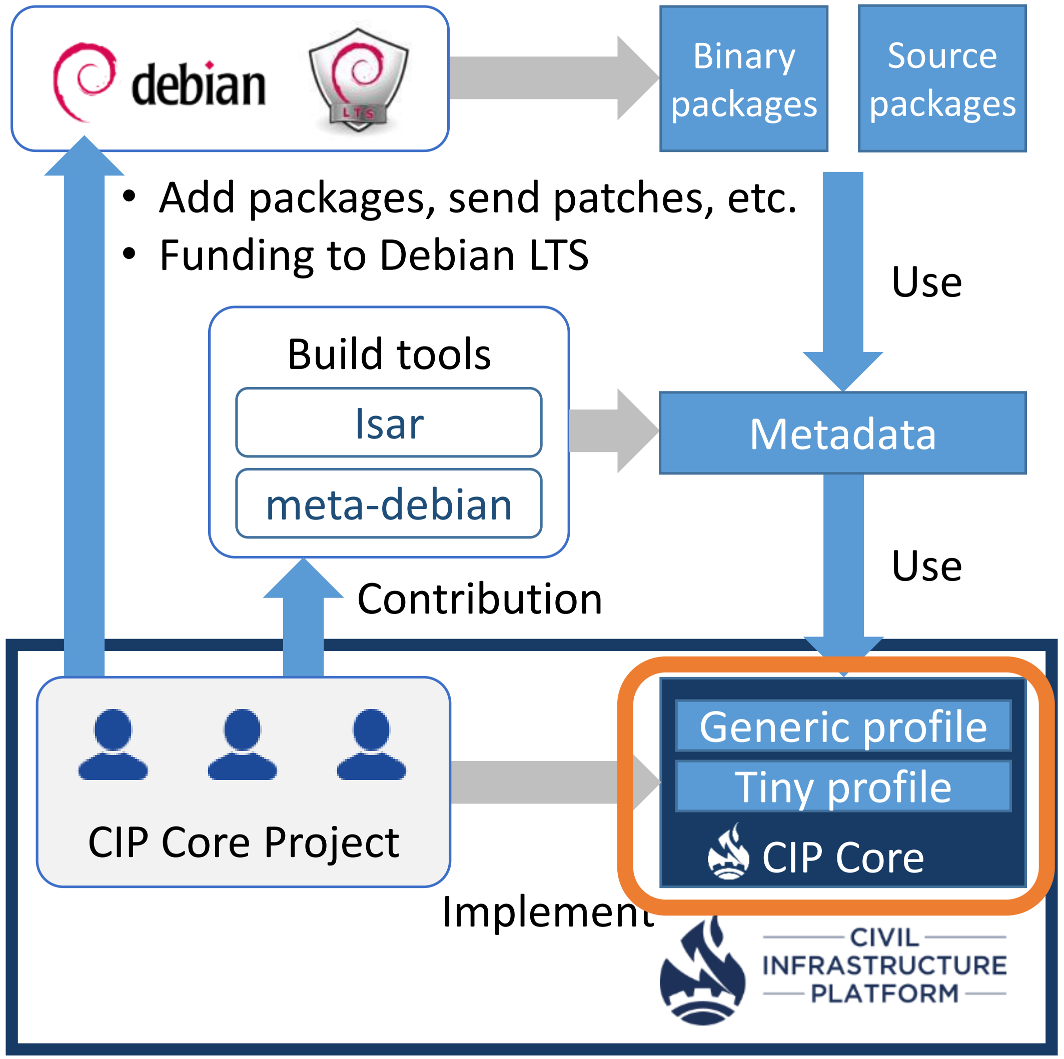
1. debootstrap Debian for target, also for host if cross-building
2. Create buildchroots (target and host)
3. Build custom Debian packages
   - pre-debianized packages
   - ad-hoc debianized packages (customizations, u-boot, kernel, …)
4. Assemble rootfs
   - debootstrap output
   - external packages
   - self-built packages
5. Run images (typically wic)
   - Filesystem image generation
   - Partitioning
   - Bootloader installation and configuration



CIP Core: Implementation

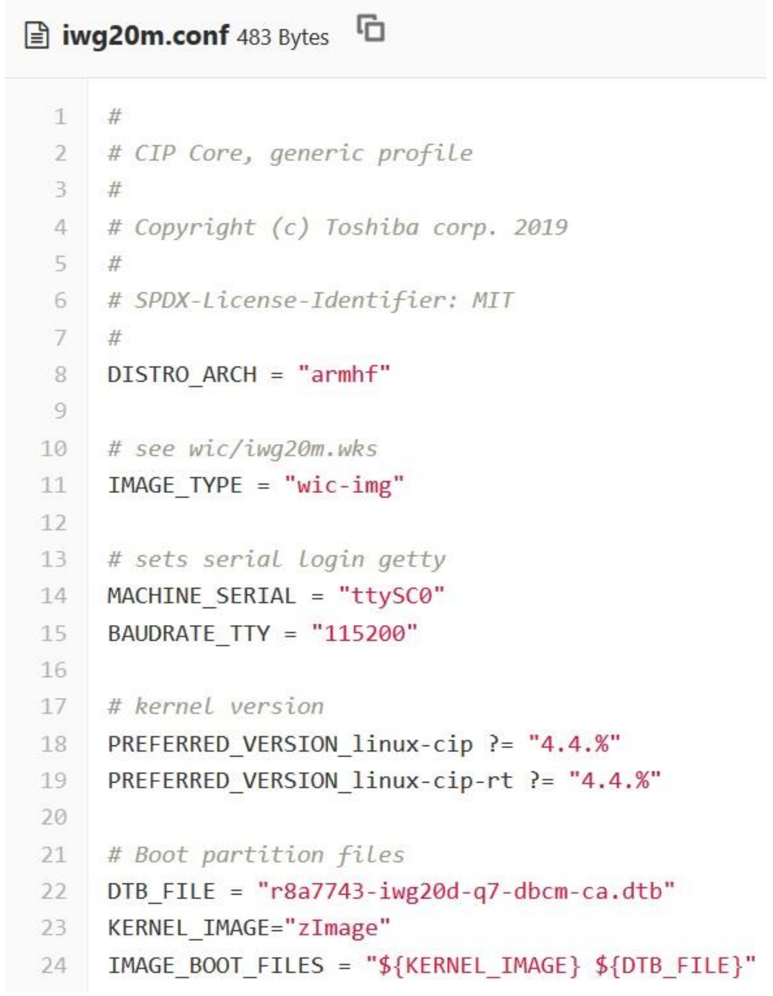
- Debian-based implementation
  - Mature, high-quality, mainstream distro.
  - Many new & old architecture supports
  - Suitable for small and big installations
  - Security updates
- Profiles

	Generic profile	Tiny profile
Approach	Binary packages	Source packages
Tool	Isar	Deby



CIP Core Generic Profile

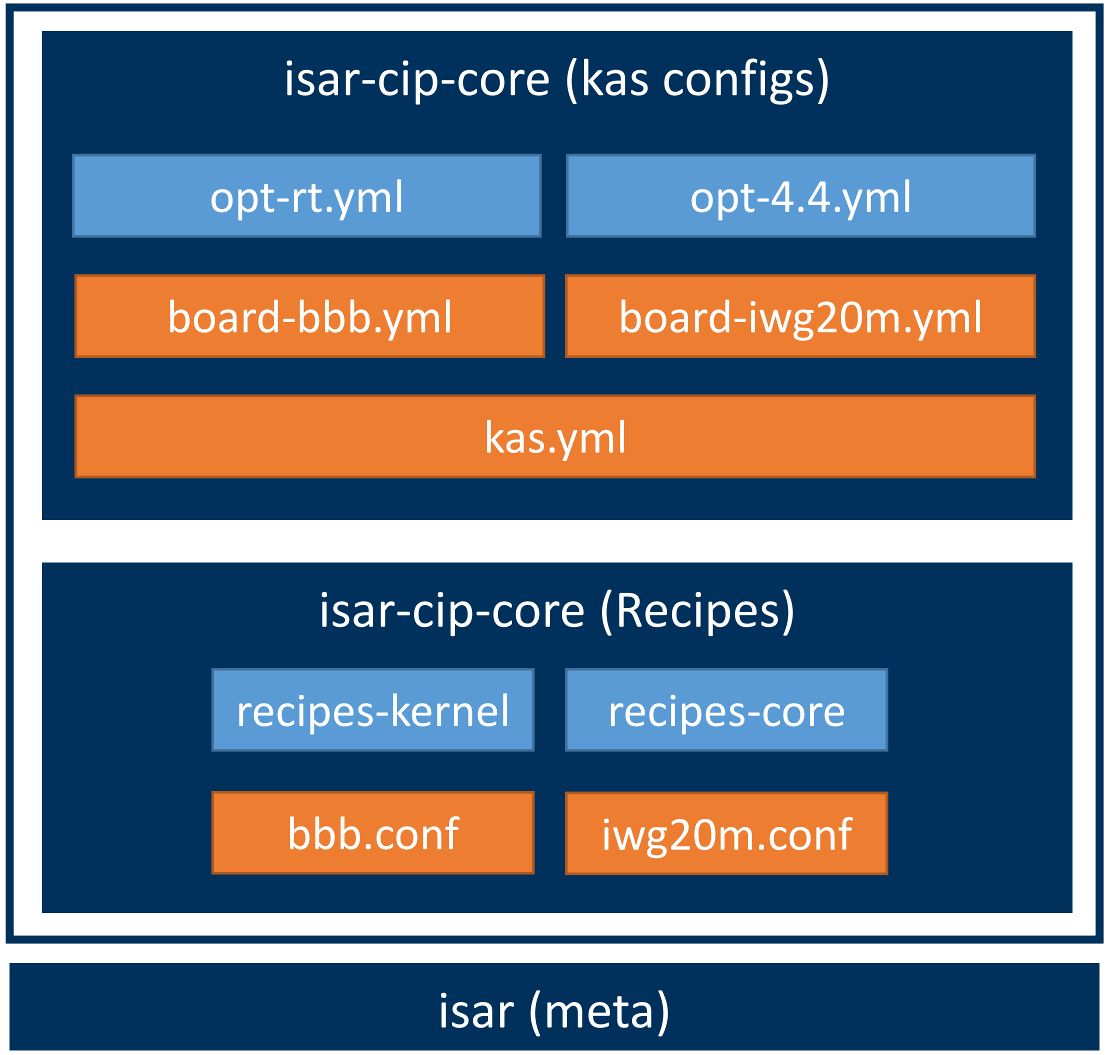
- Repository
  - https://gitlab.com/cip-project/cip-core/isar-cip-core
- Recipes (A layer for Isar)
  - Distro settings to specify kernel & Debian version
  - Machine settings for CIP reference H/W
  - Other common settings for rootfs
- Kas configurations
  - Settings for each board
    - board-bbb.yml, board-iwg20m.yml, etc.
  - Optional feature changes
    - opt-rt.yml, etc.

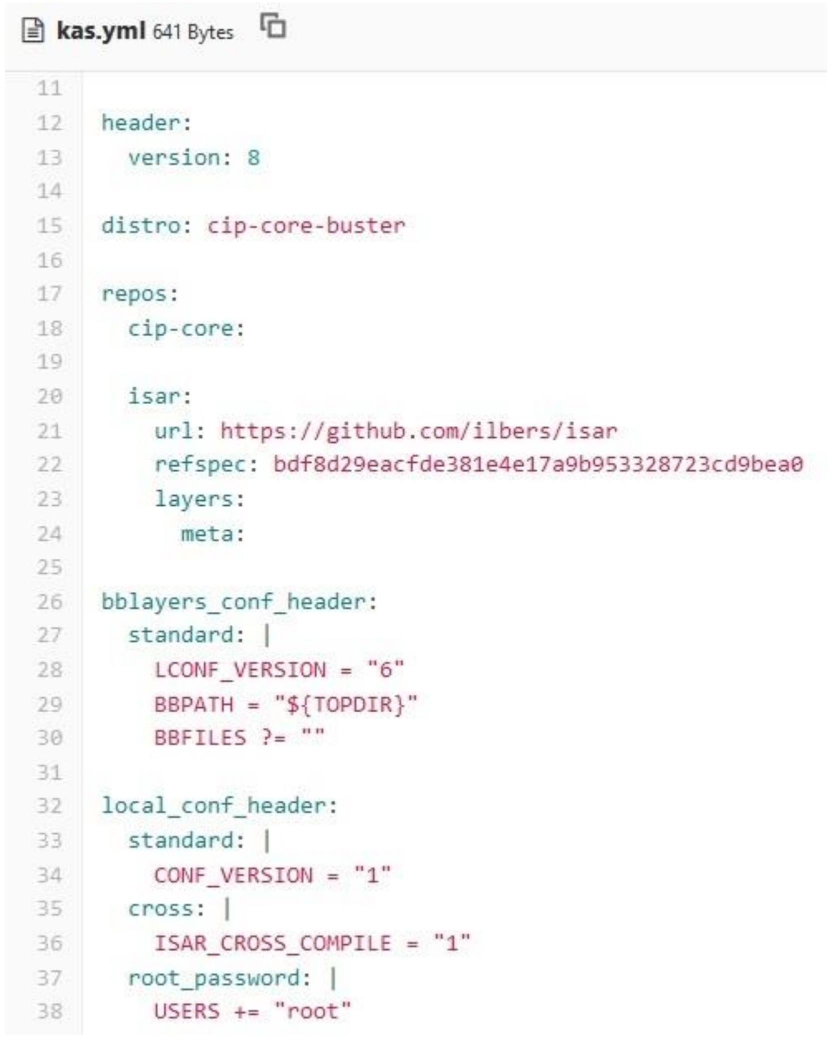

iwg20m.conf 483 Bytes

```
#
# CIP Core, generic profile
#
# Copyright (c) Toshiba corp. 2019
#
# SPDX-License-Identifier: MIT
#
DISTRO_ARCH = "armhf"

# see wic/iwg20m.wks
IMAGE_TYPE = "wic-img"

# sets serial login getty
MACHINE_SERIAL = "ttySC0"
BAUDRATE_TTY = "115200"

# kernel version
PREFERRED_VERSION_linux-cip ?= "4.4.%"
PREFERRED_VERSION_linux-cip-rt ?= "4.4.%"

# Boot partition files
DTB_FILE = "r8a7743-iwg20d-q7-dbcm-ca.dtb"
KERNEL_IMAGE="zImage"
IMAGE_BOOT_FILES = "${KERNEL_IMAGE} ${DTB_FILE}"
```

conf/machine/iwg20m.conf

CIP Core Generic Profile




kas.yml 641 Bytes

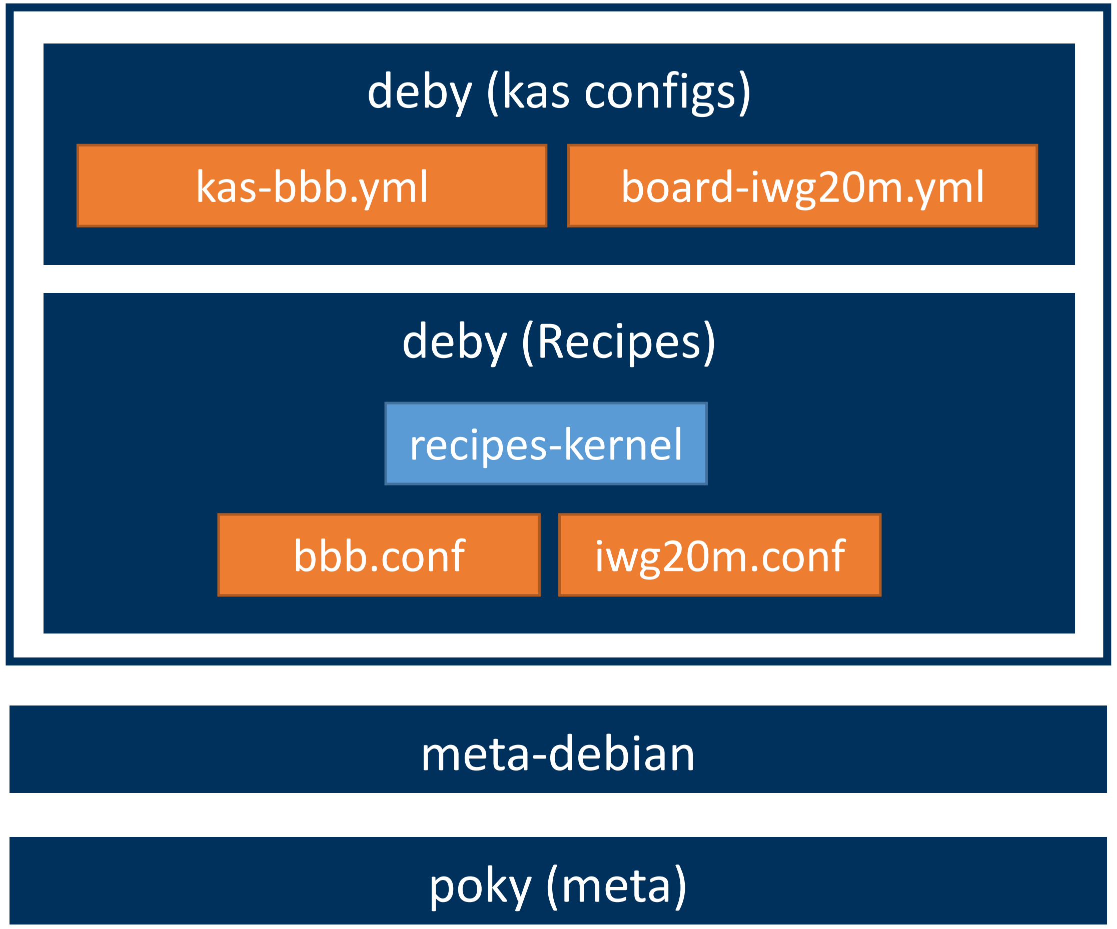
```
header:
  version: 8

distro: cip-core-buster

repos:
  cip-core:

  isar:
    url: https://github.com/ilbers/isar
    refspec: bdf8d29eacfde381e4e17a9b953328723cd9bea0
    layers:
      meta:

bblayers_conf_header:
  standard: |
    LCONF_VERSION = "6"
    BBPATH = "${TOPDIR}"
    BBFILES ?= ""

local_conf_header:
  standard: |
    CONF_VERSION = "1"
  cross: |
    ISAR_CROSS_COMPILE = "1"
  root_password: |
    USERS += "root"
```

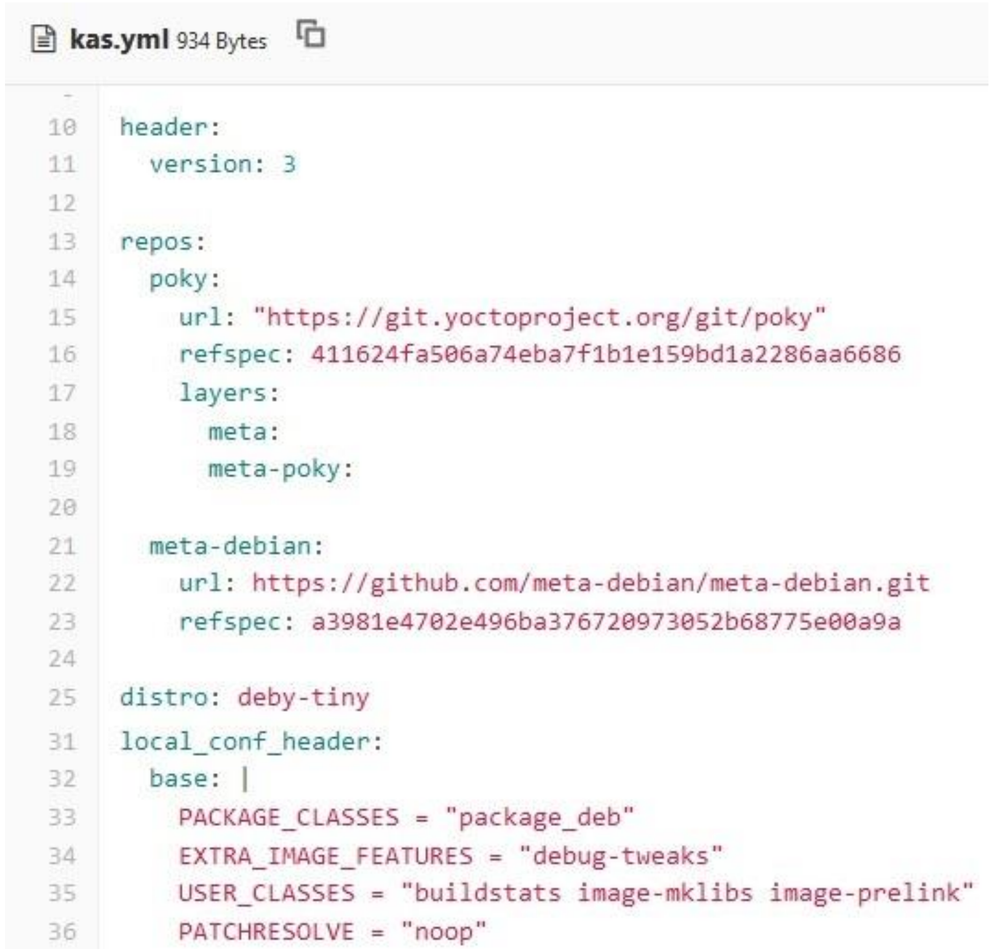
kas.yml

CIP Core Tiny Profile

- Repository
  - https://gitlab.com/cip-project/cip-core/deby
- Recipes (A layer for poky + meta-debian)
  - Target poky branch: **warrior** (Yocto Project 2.7)
  - Machine settings for CIP reference H/W
- Kas configurations
  - Settings for each board
    - kas-bbb.yml, kas-iwg20m.yml, etc.

CIP Core Tiny Profile





```yaml
header:
  version: 3

repos:
  poky:
    url: "https://git.yoctoproject.org/git/poky"
    refspec: 411624fa506a74eba7f1b1e159bd1a2286aa6686
    layers:
      meta:
      meta-poky:

  meta-debian:
    url: https://github.com/meta-debian/meta-debian.git
    refspec: a3981e4702e496ba376720973052b68775e00a9a

distro: deby-tiny
local_conf_header:
  base: |
    PACKAGE_CLASSES = "package_deb"
    EXTRA_IMAGE_FEATURES = "debug-tweaks"
    USER_CLASSES = "buildstats image-mklibs image-prelink"
    PATCHRESOLVE = "noop"
```

kas.yml

Example: Building Images for BeagleBone Black

- Generic profile (Isar)

```
$ git clone https://gitlab.com/cip-project/cip-core/isar-cip-core && cd isar-cip-core
$ wget https://raw.githubusercontent.com/siemens/kas/master/kas-docker
$ chmod a+x kas-docker
$ ./kas-docker --isar build kas.yml:board-bbb.yml
$ dd if=/path/to/cip-core-image-cip-core-buster-bbb.wic.img of=/dev/mmcblk0 ...
```

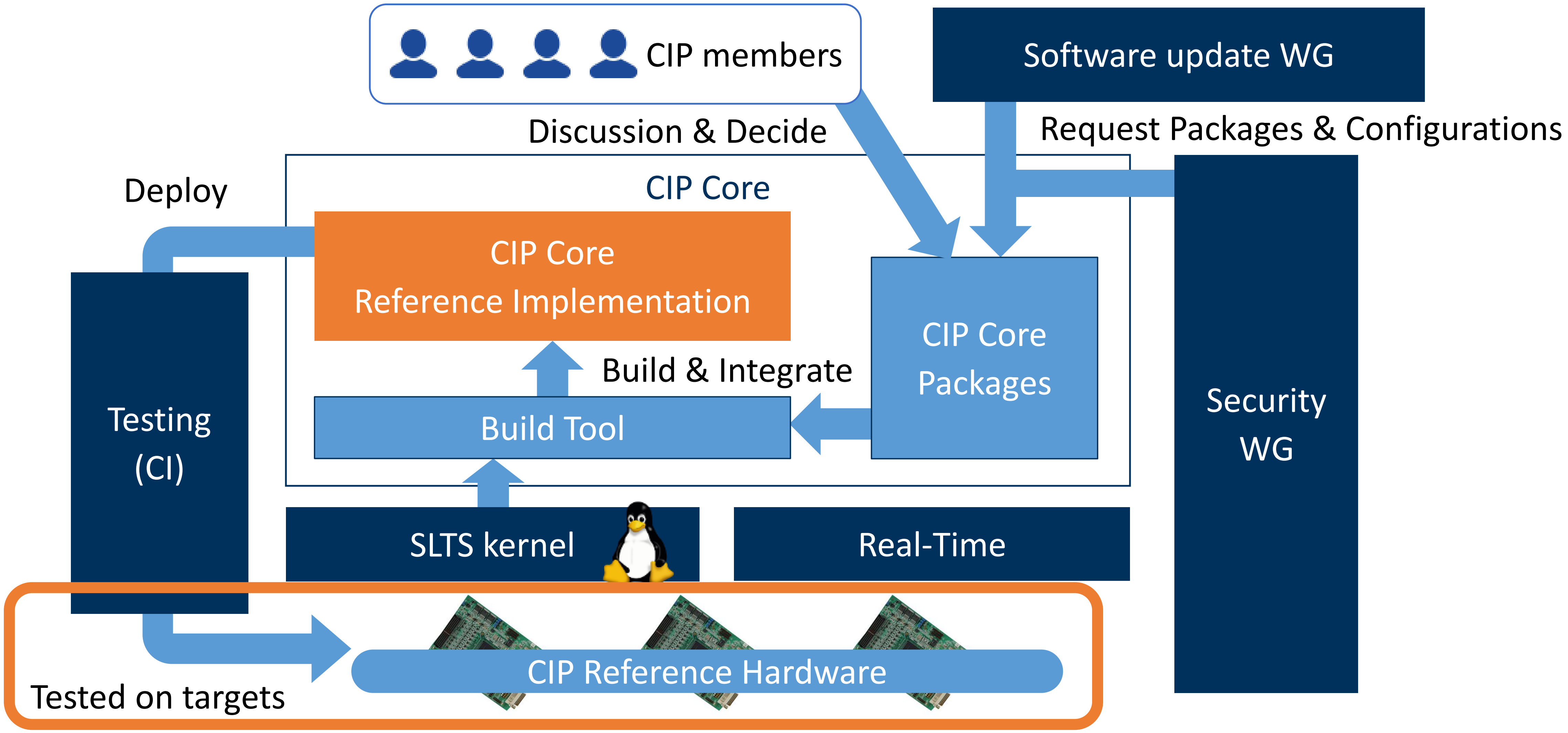
- Tiny profile (Deby)

```
$ git clone https://gitlab.com/cip-project/cip-core/deby && cd deby
$ ./scripts/setup-kas-docker.sh
$ ./kas-docker build kas-bbb.yml
```

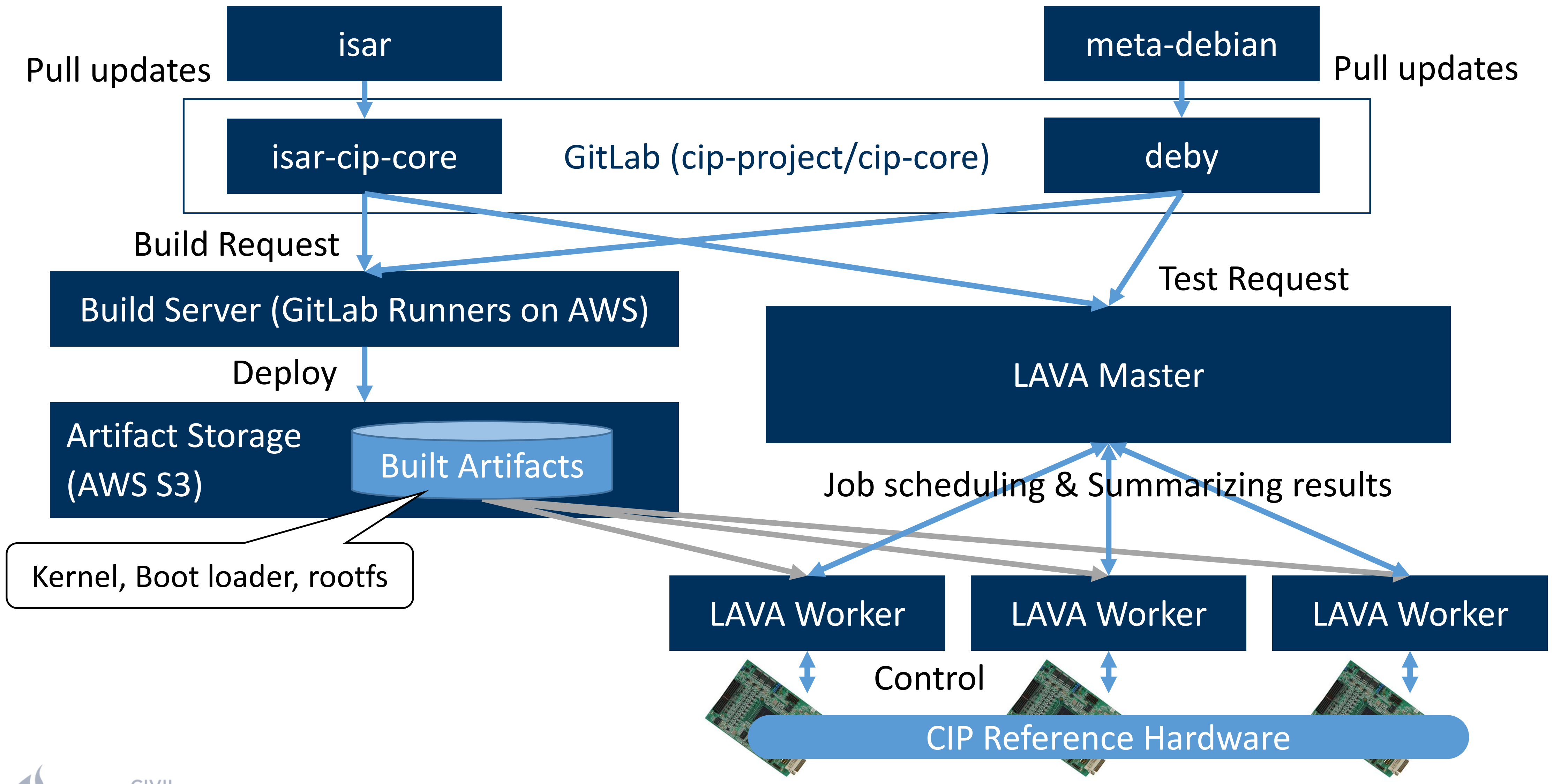
Preferred Use Cases

	Isar (Generic Profile)	Deby (Tiny Profile)
Available Packages	= Debian	App. 50 (+ Yocto Extension)
Footprint	> 100MB	2MB - 100MB
Compatibility	Debian (Binary packages)	Yocto Project (Recipes)
Required skill set	Debian (Packaging) bitbake	Yocto Project
Build time (minimal image)	Around 10min	Around 1h
Customization needs	Selected packages	Up to toolchain settings
Fitted systems (Examples)	IoT gateways, edge devices, industrial controllers …	Small IoT devices …

CIP Core: Position in CIP Projects



CIP Core: Testing Architecture



Future Plans for CIP Core Implementation

- Enable direct use in product development
  - Regular releases of tested layer with dependencies
  - Mirroring of source & binary dependencies
- Provide image corresponding to CIP package list
- Integrate and test results of other CIP workgroups
  - Robust system update (Software update WG)
  - Functions to meet cybersecurity standard requirements (Security WG)

Summary

- CIP provides long-term maintained Open Source base layer, consisting of kernel and essential packages
- CIP Core defines package set and ensures integration
- Two implementation flavors available
  - Deby for smaller, Yocto/OE-compatible projects
  - isar-cip-core for medium to larger, Debian-compatible projects
- More product-ready features to come, from software update to security hardening

Questions

CIVIL
INFRASTRUCTURE
PLATFORM