Testing Overview

Chris Paterson, Renesas Michael Adler, Siemens

CIP Testing Working Group
Automated Testing Summit Lyon, 31/10/19

Introduction

- Michael Adler
  - Linux Consultant, Siemens AG
  - Working on Linux container solutions and firmware updates for embedded devices
  - Interested in functional programming and modern programming languages
  - Passionate NixOS user

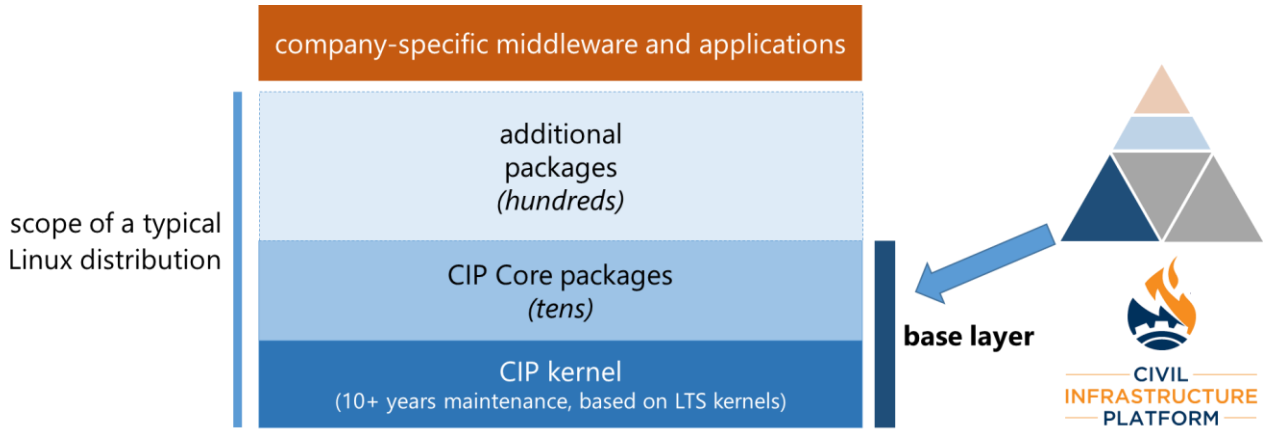
- Chris Paterson
  - Project lead in the Linux team at Renesas Electronics Europe
  - Heading the “testing” working group for the CIP project
  - Newly found love for CI

Introduction

Civil Infrastructure Platform

Establishing an **open source base layer** of industrial grade software to enable the use and implementation of software building blocks for civil infrastructure

- Super Long-Term Support(ed) Linux Kernels – 10+ years
- Real Time Linux
- CIP Core – Reference filesystem
- Security – IEC 62443 certification



The backbone of CIP are the member companies

CIVIL INFRASTRUCTURE PLATFORM

RENESAS SIEMENS TOSHIBA

CodeThink cybertrust HITACHI Inspire the Next MOXA Plat'Home There, we are. Internet of Things

Budget ¥ € £ $ → Developers, maintainers

Optional: funding of selected projects

Contribution & usage / integration

Open Source Projects

Real-Time LINUX KernelCI debian Reproducible Builds EDGE X FOUNDRY

Testing Goals

- Provide an environment to test CIP software projects on the CIP reference hardware
  - SLTS Kernel
  - SLTS RT Kernel
  - CIP Core
    - Tiny profile (Deby)
    - Generic profile (ISAR)
  - SW update
  - […]
- Open source
- Collaboration

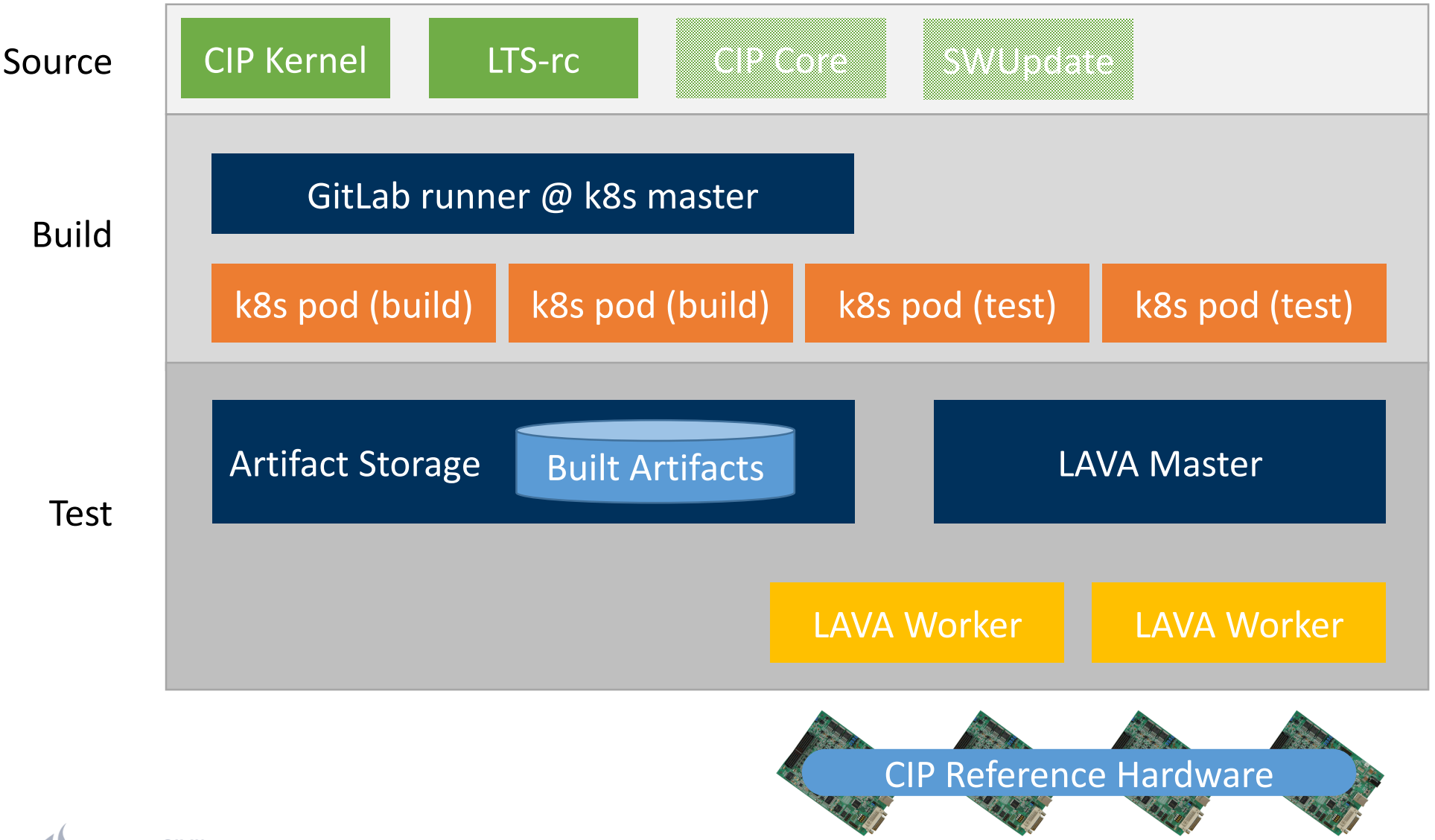
CIP Reference Hardware

- The CIP project is currently supporting the following reference hardware:

Device	SLTS v4.4	SLTS v4.19
AM335x Beaglebone Black (Armv7)	✔	✔
QEMU x86_64	✔	✔
RZ/G1M iWave Qseven Dev Kit (Armv7)	✔	✔
RZ/G2M HopeRun HiHope (Armv8)		✔
SIMATIC IPC227E (x86-64)	✔	✔
OpenBlocks IoT VX2 (x86-64)		✔

- Currently testing ~30 Kernel configurations (including some RT configs) across the two SLTS versions.

Architecture Overview



Tests (early days!)

- Currently CIP are running the following tests:
  - Boot test
    - uname -a
  - Spectre/Meltdown checker
    - https://github.com/Linaro/test-definitions/tree/master/automated/linux/spectre-meltdown-checker-test

- In progress:
  - LTP
    - ltp-cve-tests, ltp-dio-tests, ltp-fs-tests, ltp-ipc-tests, ltp-math-tests, ltp-open-posix-tests, ltp-sched-tests, ltp-syscalls-tests and ltp-timers-tests
    - https://github.com/Linaro/test-definitions/tree/master/automated/linux/ltp
    - https://github.com/Linaro/test-definitions/tree/master/automated/linux/ltp-open-posix

Next Steps

- Improve job reporting
  - Results in GitLab/LAVA/email are easily lost
  - We plan to submit all test results to the KernelCI project

- Increase test coverage
  - CIP Core (reference filesystem)
  - kselftests, Jitterdebugger, Linaro test definitions, Benchmarks, Hardware testing (CAN/PCIe/USB etc.)

- Add more boards/LAVA workers
  - Speed up testing
  - Improve board availability

- **Collaboration with the Automated Testing Community**


THE LINUX FOUNDATION
PROJECTS
CIVIL
INFRASTRUCTURE
PLATFORM
Tooling

gitlab-cloud-ci

Container-based CI-infrastructure from Scratch

gitlab-cloud-ci
Motivation

SIEMENS
Ingenuity for life

- ISAR: Integration System for Automated Root filesystem generation
  - Similar to Yocto, but use upstream deb-packages as much as possible
    - => No need to maintain your own Linux distribution anymore
  - Perform cross-architecture builds
  - Very versatile and flexible, but also easy to make mistakes => CI solution desired
  - Used by Civil Infrastructure Project, Siemens and others

- **Problem**: Standard (Gitlab) CI runners do not work with ISAR-based builds *in general*
- Reasons:
  - `binfmt_misc` is not namespace-aware (yet!), but is required for cross-architecture builds
  - `debootstrap`: some (minor) issues with CAP_MKNOD

gitlab-cloud-ci
Possible Solutions

SIEMENS
Ingenuity for life

- **“Hard” Way**
  - Make `binfmt_misc` user namespace aware and merge it upstream
  - Fix all issues with `debootstrap` (mostly working around MKNOD issue)
  - Not *that* hard, but would take some years until it is merged upstream and available in distros

- **Easy Way**
  - Just use privileged containers (CAP_SYS_ADM)
  - … or use a good old-fashioned VM after all

Which way to go?

Both ways! But start with the easy one and wait for the hard one ☺

SIEMENS
Ingenuity for life

- **Following the “Easy“ Way**

- Let’s make a wishlist! It should be:
  - **Fast**: CI feedback time must be short (fast CPU, fast SSD, fast network)
  - **Scalable**: Perform many CI jobs in parallel
  - **Secure**: Can we always trust our payload? Isolation would be nice!
  - **Cheap**: opex, capex ☺
  - **Possibly Reliable**: SA or HA, reproducible setup (automated)
  - Must be compatible/usable with Gitlab
- Avoid vendor lock-in
  - Should work on-premise and in the cloud

gitlab-cloud-ci
Enter gitlab-cloud-ci

SIEMENS
Ingenuity for life

- gitlab.com/cip-project/cip-testing/gitlab-cloud-ci
  - Developed internally at Siemens and published under Apache-2.0 license
  - Written in Python 3
  - Yet another Kubernetes bootstrapping tool? Like Kops or Kubespray?
  - Or is it more a “general tool” like Terraform?
  - Neither! gitlab-cloud-ci is a thin wrapper (“**glue code**”) around existing battle-proven tools
- **Features**
  - Create and bootstrap Kubernetes cluster (SA/HA) from scratch
    - Choose between AWS and on-premise setup
    - Heavy-lifting is done by kops/kubeadm
  - Deploy dashboard, cluster-autoscaler, Gitlab runners, binfmt_misc “hack” and more
  - Everything that can be created can be destroyed as well
  - Basic lifecycle management functionality

gitlab-cloud-ci
Real-World Usage

SIEMENS
Ingenuity for life

- **CIP**: Has been permanently in-use for almost 6 months:
  - 100% uptime for master node (thanks AWS)
  - Occasionally minor hick-ups with cluster-autoscaler (or AWS)
    - AWS + Kubernetes autoscaling is rather slow (a few minutes) and sometimes results in timeouts
    - Might improve once we upgrade to latest Kubernetes + cluster-autoscaler add-on
  - Started small (m5d.xlarge 4 vCPUs, 16GB mem, 1xNVMe SSD),
    but recently went up to m5d.4xlarge (16vCPUs, 64GB mem, 2xNVMe SSD)
  - Dynamically scaling between 0 and 40 slave nodes

- **Siemens**: Used for internal Gitlab instance (on-premise setup)

gitlab-cloud-ci
Outlook & Contributing

SIEMENS
Ingenuity for life

- **Contributions are welcome!**
- … bug reports too ☺
- Planned Features:
  - Add cluster monitoring (Issue #4)
  - Mixed EC2 instance sizes (configurable on a per-job basis) (Issue #6)
  - More configuration options (Issue #3)
  - Profiles
  - Kata Containers
- Ideas:
  - GCE integration (cheaper than AWS because master node is free)

linux-cip-ci

- Simple ‘tool’ that manages the build and test process for the CIP Linux SLTS Kernels.
- Consists of two Docker containers:
  - ‘Build’
    - Contains: Linux Kernel build scripts & dependencies, CIP Kernel configs repo
    - Appropriate cross-toolchain downloaded on the fly
    - Builds the Kernel for the specified configuration
    - Stores build artifacts in GitLab’s artifact storage
  - ‘Test’
    - Contains: LAVA test job generation scripts, LAVA and AWS CLI tools
    - Uploads build artifacts to AWS S3 storage (so LAVA workers can access them)
    - Creates LAVA job definitions for the specified platforms and tests
    - Submits LAVA jobs and waits for the results

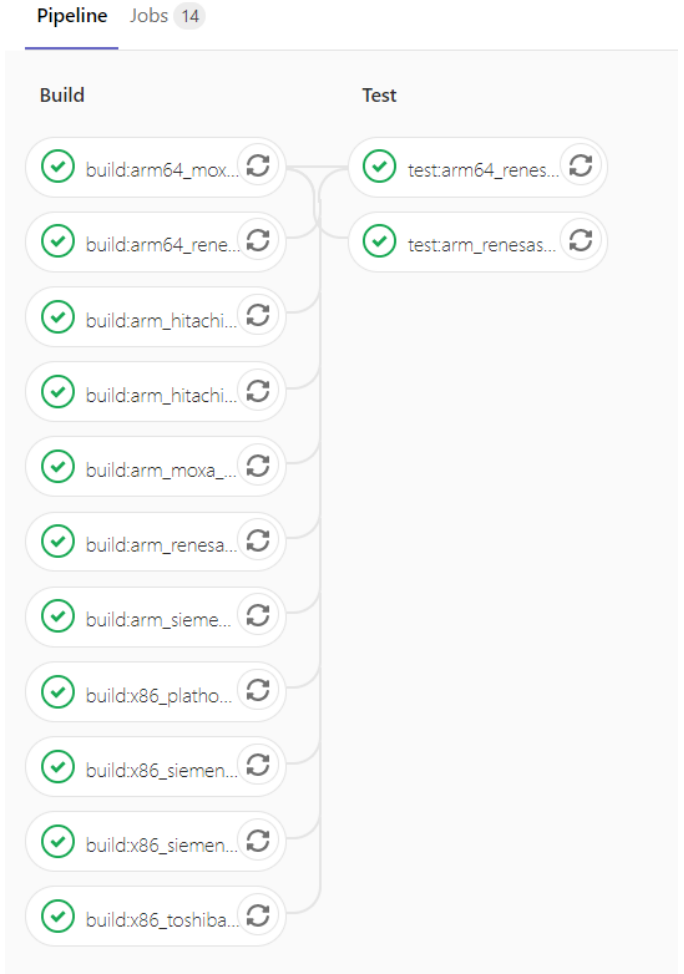
- https://gitlab.com/cip-project/cip-testing/linux-cip-ci

linux-cip-ci: Example .gitlab-ci.yml

```yaml
variables:
  GIT_STRATEGY: clone
  GIT_DEPTH: 10
  DOCKER_DRIVER: overlay2
  DOCKER_IMAGE_TAG: v2

build:arm64_renesas_defconfig:
  stage: build
  image: registry.gitlab.com/cip-project/cip-testing/linux-cip-ci:build-$DOCKER_IMAGE_TAG
  variables:
    BUILD_ARCH: arm64
    CONFIG: renesas_defconfig
    CONFIG_LOC: cip-kernel-config
    DEVICES: r8a774c0-ek874 r8a774a1-hihope-rzg2m-ex
    DTBS: r8a774c0-ek874.dtb r8a774a1-hihope-rzg2m-ex.dtb
  script:
    - /opt/build_kernel.sh
  artifacts:
    name: "$CI_JOB_NAME"
    when: always
    paths:
      - output

test:arm64_renesas_defconfig:
  stage: test
  needs: ["build:arm64_renesas_defconfig"]
  image: registry.gitlab.com/cip-project/cip-testing/linux-cip-ci:test-$DOCKER_IMAGE_TAG
  variables:
    GIT_STRATEGY: none
  script:
    - /opt/submit_tests.sh
  artifacts:
    name: "$CI_JOB_NAME"
    paths:
      - output
```



Questions?

Thank you!

Contact

#cip (freenode)	cip-dev@lists.cip-project.org
patersonc	chris.paterson2@renesas.com
therisen	michael.adler@siemens.com

Links

- CIP testing wiki page
  - https://wiki.linuxfoundation.org/civilinfrastructureplatform/ciptesting/centalisedtesting
- CIP reference hardware
  - https://wiki.linuxfoundation.org/civilinfrastructureplatform/ciptesting/cipreferencehardware
- CIP Kernel Configurations
  - https://gitlab.com/cip-project/cip-kernel/cip-kernel-config
- CIP LAVA master
  - https://lava.ciplatform.org/
- CIP lava-docker
  - https://gitlab.com/cip-project/cip-testing/lava-docker
- CIP Kubernetes k8s pod manager
  - https://gitlab.com/cip-project/cip-testing/gitlab-cloud-ci
- CIP Kernel CI build tool
  - https://gitlab.com/cip-project/cip-testing/linux-cip-ci