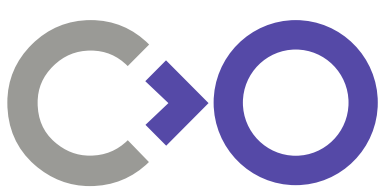
Testing your distribution with LAVA

Andrej Shadura, Guillaume Tucker

Collabora

25 August 2018



COLLABORA

Open First

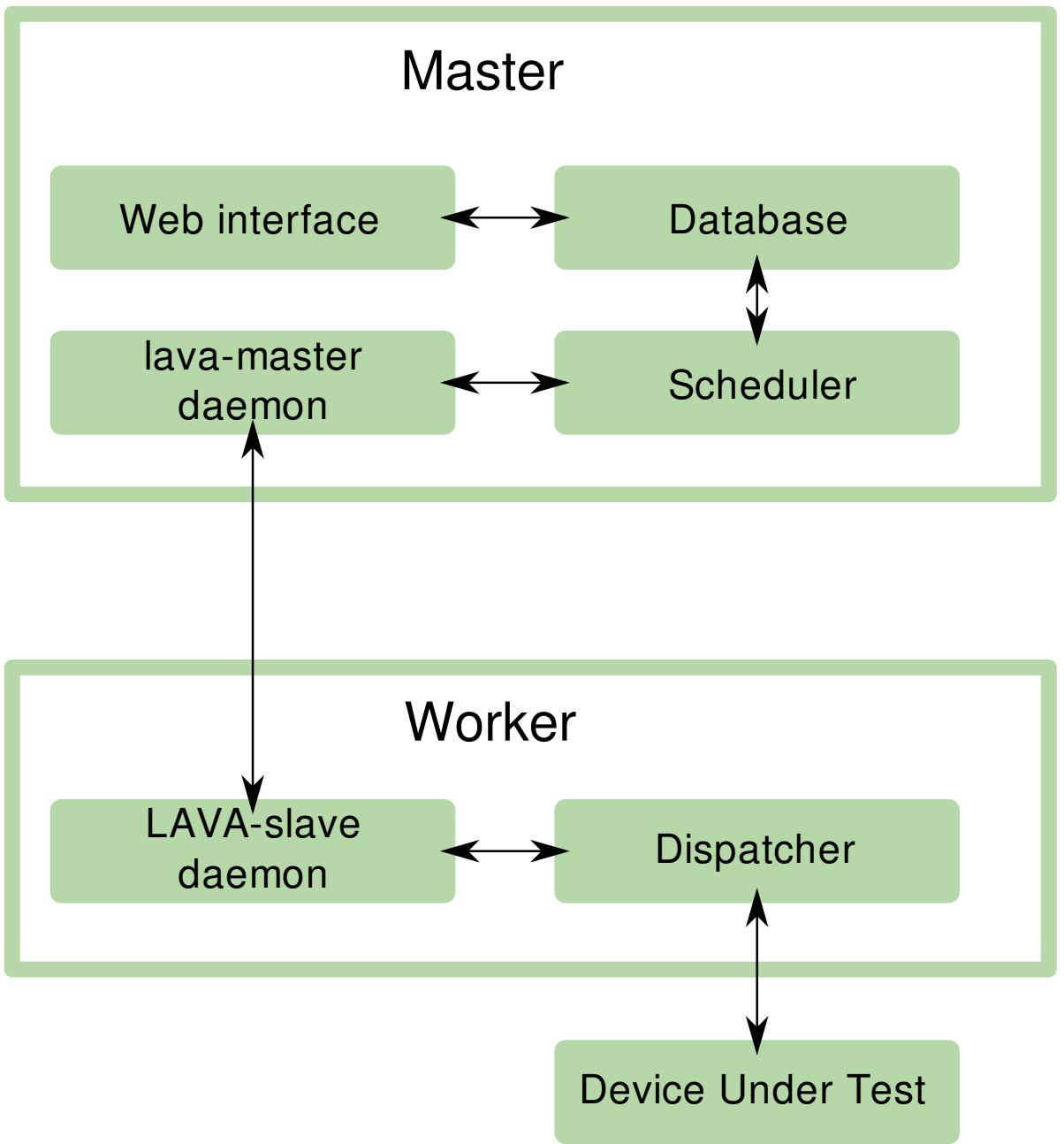
What is LAVA?

- Linaro Automated Validation Architecture
- Runs your tests on the target hardware

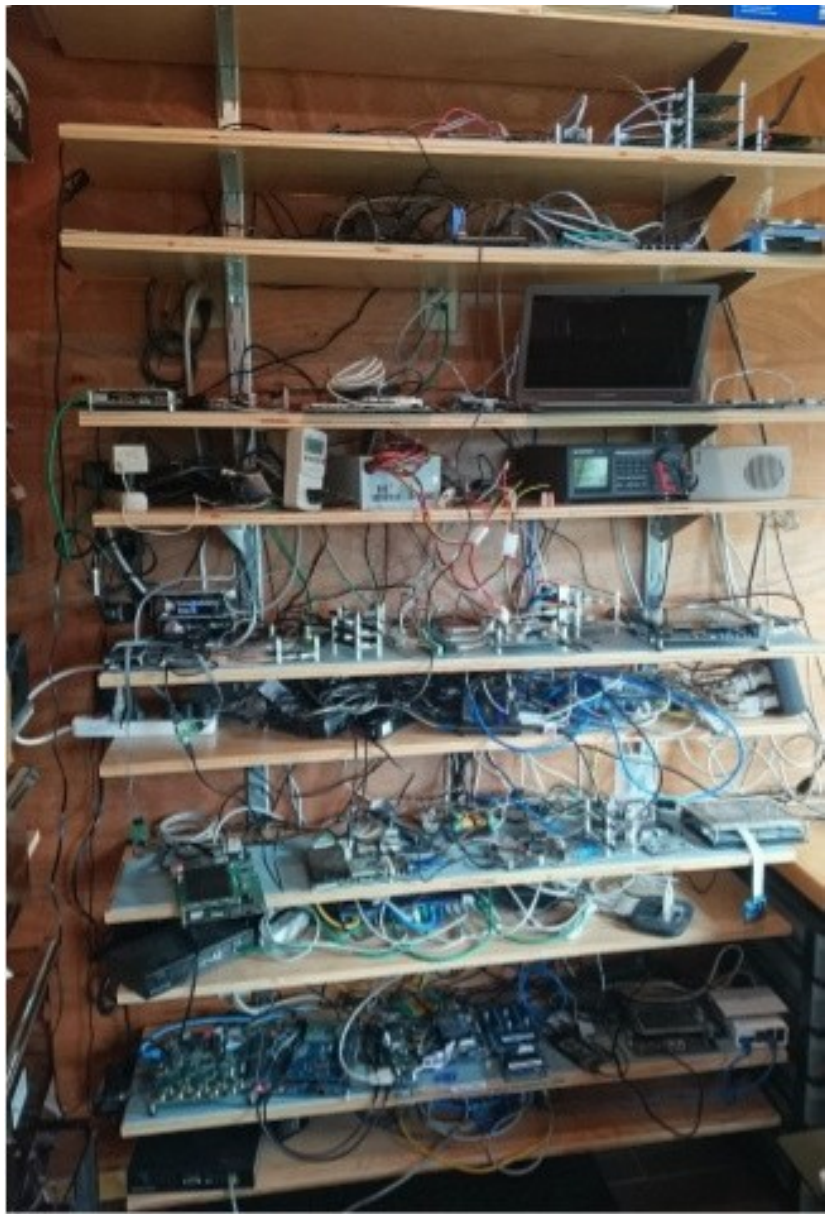
LAVA overview

- Power on a device
- Boot into the target OS
- Run tests
- Collect results

LAVA overview

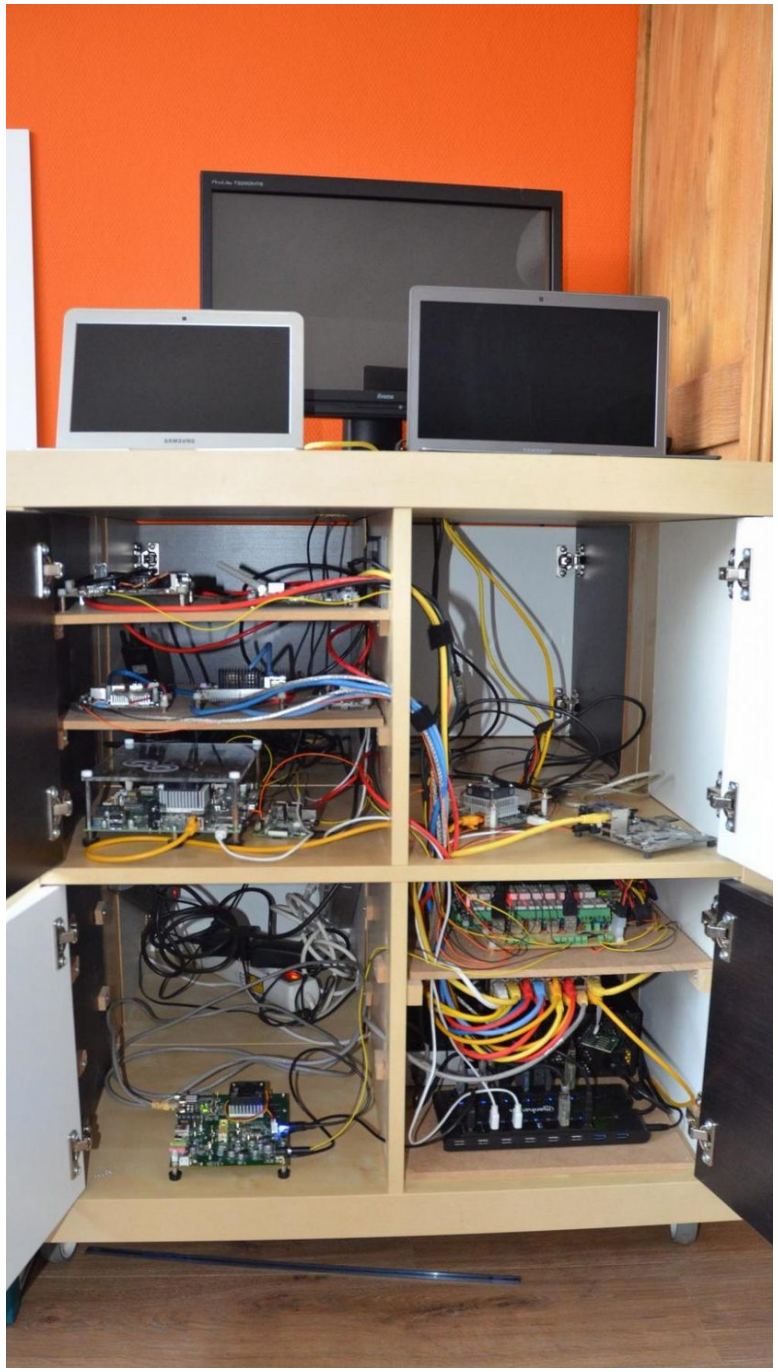


LAVA test labs



LAVA test labs

LAVA test labs



LAVA jobs

Pipelines defined in YAML:

- `boot`
- `deploy`
- `test`

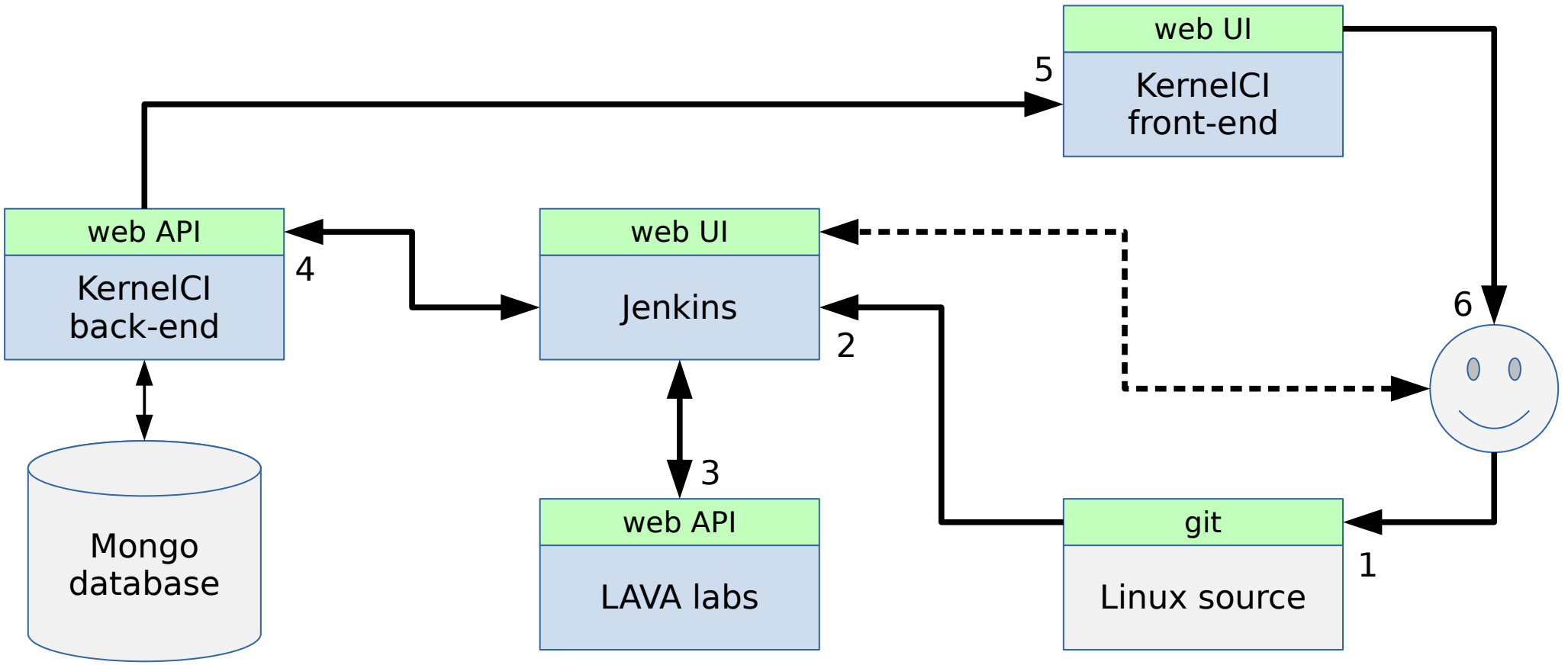
Jobs specify high-level details only

LAVA ‘knows’ the low-level details

KernelCI

- Build all new commits for a number of architectures
- Boot them on the actual hardware
- Report all failures

KernelCI


web UI
KernelCI
front-end
5
web API
KernelCI
back-end
4
web UI
Jenkins
2
6
3
Mongo
database
web API
LAVA labs
git
Linux source
1

Why use Kernel CI

- Detect failures early
- Test on many devices
- Test branches before they’re merged
- Automatic bisection (WIP)

How to use LAVA

```
1 device_type: bcm2836-rpi-2-b
2 job_name: boot test
3 visibility: public
4
5 timeouts:
6   job:
7     minutes: 10
8
9 actions:
10 - deploy: …
11 - boot: …
12 - test: …
```

Deploy

```
- deploy:
    to: tftp
    kernel:
      url: https://server.org/path/to/zImage
      type: zimage
    dtb:
      url: https://server.org/path/to/raspberry-pi-2b.dtb
    ramdisk:
      url: https://server.org/path/to/ramdisk-armhf.cpio.gz
      compression: gz
    os: oe
```

Boot

```
- boot:
    method: u-boot
    commands: ramdisk
    auto_login:
      login_prompt: 'root login:'
      username: root
    prompts: ['# ']
```

Test

```
- test:
    timeout:
      minutes: 15
    name: sanity-check
    definitions:
      - repository: https://git.server.org/tests/tests
        revision: master
        from: git
        path: sanity-check.yaml
        name: sanity-check
```

More complex setups

- Need to boot a full OS image?
- Two-stage setup:
  1. Boot minimal OS
  2. Flash the image
  3. Reboot into the image

Deploy the second stage

```
- deploy:
    namespace: system
    timeout:
      minutes: 20
    to: usb
    device: sd-card
    tool:
      prompts: ['copying time: [0-9ms\.\ ]+,
        copying speed [0-9\.]+ MiB\/sec']
  images:
    image:
      url: "https://…/apertis_18.09-target-arm64-\
          uboot_20180801.0.img.gz"
    bmap:
      url: "https://…/apertis_18.09-target-arm64-\
          uboot_20180801.0.img.bmap"
```

Deploy the second stage (part 2)

```
1       os: ubuntu
2     writer:
3       tool: /usr/bin/bmaptool
4       options: copy {DOWNLOAD_URL} {DEVICE}
5       prompt: 'bmaptool: info'
```

How to define tests?

YAML!

Test definition: metadata

```
1 metadata:
2   name: check-dbus-services
3   format: "Lava-Test-Shell Test Definition 1.0"
4   description: "Sanity-check all installed D-Bus services"
5   maintainer: "simon.mcvittie@collabora.co.uk"
6   scope:
7   - functional
8   devices:
9   - i386
10  environment:
11  - lava-test-shell
```

Test definition: dependencies

```
1 install:
2   deps:
3   - apertis-tests
4   - dbus-tests
```

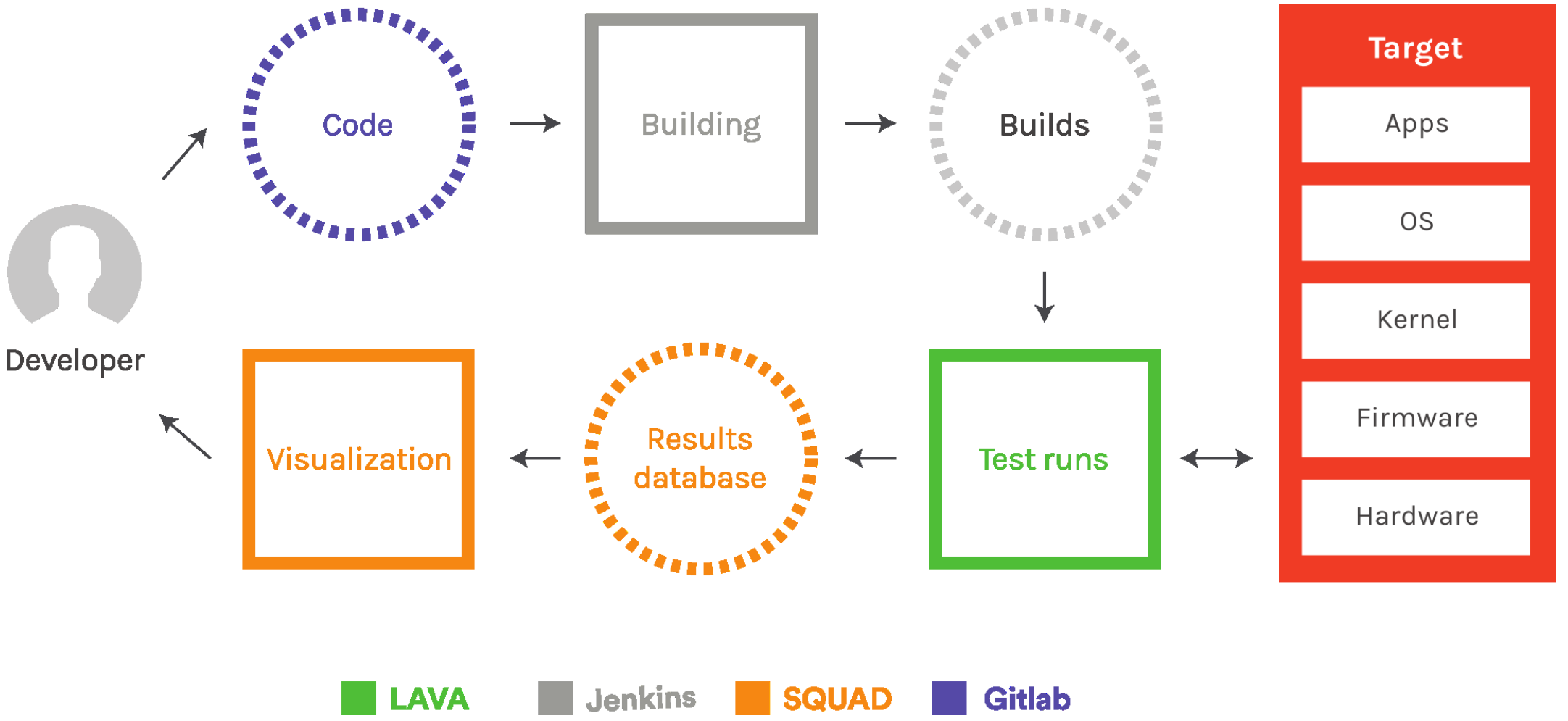
Test definition: run stuff

```
1 run:
2   steps:
3   - common/run-test-in-systemd --user=user
4         dbus/check-dbus-services
5   - common/run-test-in-systemd dbus/check-dbus-services
```

Test definition: results parsing

```
parse:
  pattern: 'RESULT:(?P<result>\w+):(?P<test_case_id>[^:]+):'
  # LAVA doesn't seem to have the concept of
  # an expected failure, so calling it skipped
  # is the next best thing
  fixupdict:
    xfail: skip
```

Full CI loop with LAVA



Thanks to

Guillaume Tucker

We are hiring!

www.collabora.com/careers

COLLABORA