Linaro Connect
MADRID 2024 | MAY 14-17 2024

Modern CI Solutions for Linux Kernel Testing

Vishal Bhoj <vishal.bhoj@linaro.org>

Agenda

- Who am I ?
- Problem we are trying to address ?
- Benefits of Modern CI Platforms
- What is TuxSuite?
- TuxSuite OpenSource Tools
- Tuxsuite Plans
- TuxSuite Gitlab Component
- Custom TuxSuite Plan as input
- Future work

Who am I ?

- Tech Lead, Microservices CI Team in Linaro
- 16 years in building embedded software and CI/CD solutions.
- Maintain the TuxSuite open source software tools and cloud services.

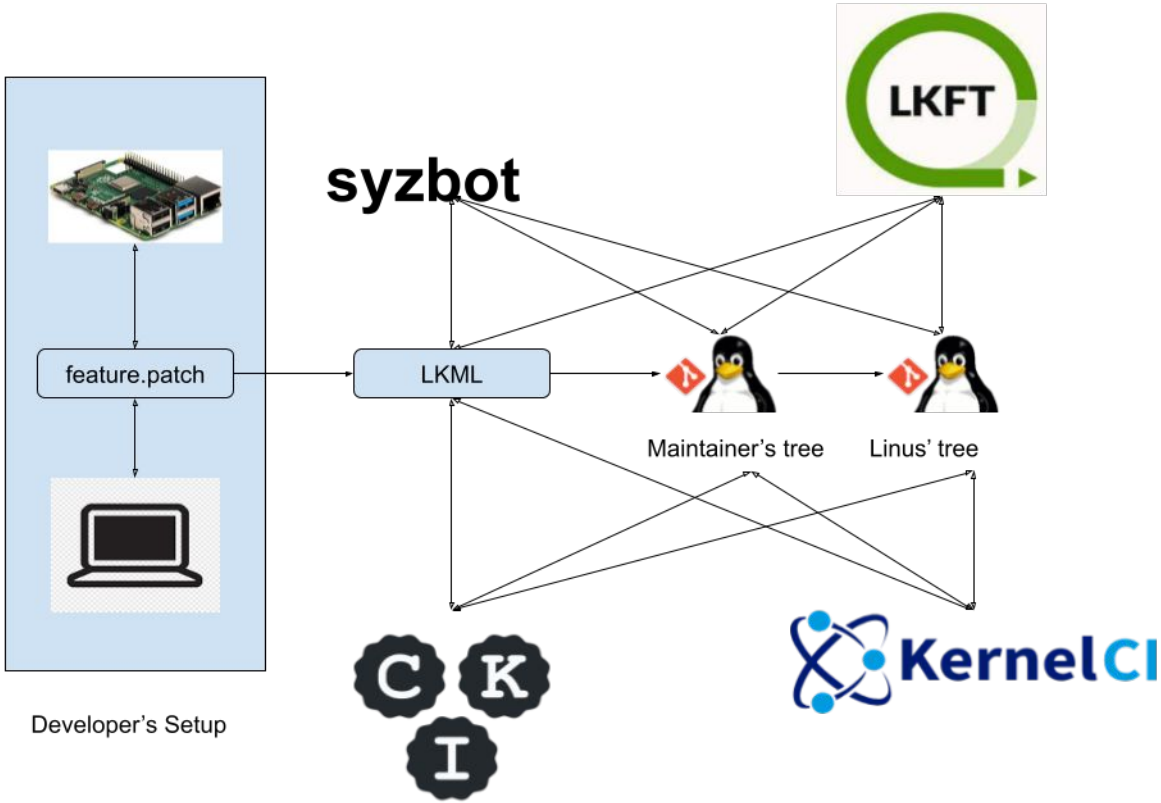
Disclaimer

- I am not a kernel developer
- I might be unaware of all the Linux testing Projects and CI infrastructures that are available as of today.

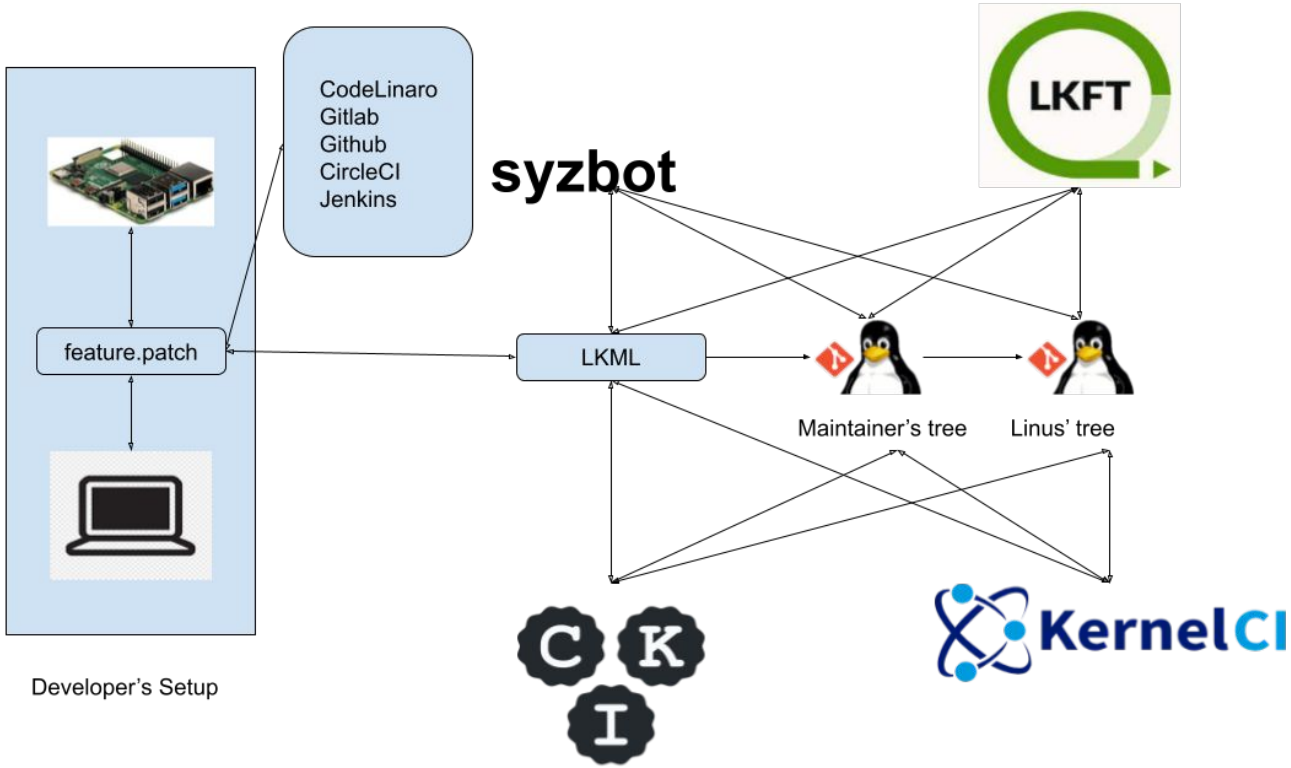
Problem we are trying to address

- Improve testing in Individual Kernel developer’s workflow
- Simplify build and test setup
- Access to a larger sanity testing framework for kernel developers
- Reduce feedback time in Kernel Testing as much as possible

Current Workflow



Proposed Workflow



Benefits of Modern CI systems

- Automated Testing
- Enhanced Collaboration
- Scalability
- Real-time feedback
- Dashboards

Modern CI systems used in Linux kernel Testing

- Gitlab CI
  - * LKFT: https://gitlab.com/Linaro/lkft/mirrors/stable/linux-stable/-/pipelines
  - CKI: https://gitlab.com/redhat/red-hat-ci-tools/kernel/cki-public-pipelines/-/pipelines
  - DRM CI: https://gitlab.freedesktop.org/mesa/mesa/-/pipelines
- Github
  - * ClangBuiltLinux: https://github.com/ClangBuiltLinux/continuous-integration2/actions
  - BPF: https://github.com/kernel-patches/bpf/actions

* Powered by TuxSuite

But, who is going to implement and maintain the CI Frameworks ?

- Replicating the CI setup is difficult
- Kernel developers may not want to learn about how to setup and maintain the CI infrastructure
- Who will maintain and update the build environments and dependencies continuously?
- Who will maintain the DUT target setup ?
  - Baseline Rootfs
  - Test Suites and their dependencies

TuxSuite comes to the rescue!

- What is TuxSuite ?
  - Suite of open source tools to simplifying builds, boots, and tests (on QEMU).
    - Supports Linux, OE, AOSP
    - TuxSuite Plans to describe end-to-end test plans
  - Cloud services
    - To scale for enterprise needs
    - Test on remote hardware targets
    - Test on FVPs
    - Test reports
- Tools built with Goals
  - Reproducibility
  - Containerised environments
  - Simplicity
  - Scalability

TuxSuite OpenSource Tools

Tools	Functionality
TuxMake	Linux Kernel build engine and toolchain containers
TuxRun	Testing engine and test-suite containers for testing on Qemu and FVP
TuxBake	Build engine and containers for Yocto/OE
TuxSuite Plans	Define build, boot, and test plans for multiple targets in one file
TuxSuite API specification	Build and run tests (on virtual targets) in the cloud, in parallel.
TuxSuite cli	Cli to Execute TuxSuite Plan **Locally** or invoke TuxSuite API in the cloud
TuxTrigger	Daemon to monitor Git trees and trigger TuxSuite plans in the cloud

What is a TuxSuite Plan?

- A Yaml file describing a combination of builds and tests.
- A plan could be made of
  - stand alone builds
  - stand alone tests
  - set of builds and set of tests to be run for each build
  - combination of all the above

```yaml
version: 1
name: Simple plan
description: Build and test linux kernel on arm64 with gcc-13
jobs:
- name: arm64
  build: {toolchain: gcc-13, target_arch: arm64, kconfig: [defconfig]}
  test: {device: qemu-arm64, tests: [ltp-smoke]}
```

Local vs Cloud Execution

Feature	Native TuxSuite Execution	TuxSuite Cloud
Test Sharding	✅	✅
Specify test Fragments in plans	✅	✅
Petition for new test workloads	✅	✅
Target tests on real remote-lab hardware	❌	✅
FVP test targets	❌	✅
Reproducer fragments	❌	✅
True Parallelization of builds and tests	❌	✅
Email summary/Report for a Plan	❌	✅

Supported Toolchains and Tests

- Toolchains
  - gcc-8 to gcc-13
  - Clang-10 to clang-18/(19-nightly)
  - Korg-clang (clang/llvm toolchains hosted on kernel.org)
  - Rust (rustgcc, rustllvm)
- Tests
  - kselftests
  - kunit
  - mmtests
  - pm-qa
  - libgpiod
  - libhugetlbfs
  - ltp
  - perf
  - rcutorture
  - v4l2
  - vdso
  - xfstests

But, how will TuxSuite help in CI ?

- Gitlab Components
  - A CI/CD component is a reusable single pipeline configuration unit.
  - Use components to create a small part of a larger pipeline, or even to compose a complete pipeline configuration.

TuxSuite Gitlab Component available!

https://gitlab.com/explore/catalog/Linaro/components/tuxsuite

TuxSuite Component

Introduction

TuxSuite Component facilitates a complete CI/CD pipeline specifically for building and testing the Linux Kernel tree on QEMU environments. This component is part of the larger TuxSuite ecosystem, provided by Linaro, which supports Linux kernel development with a suite of tools and services.

Getting Started

Prerequisites

- A Linux Kernel tree hosted on a gitlab instance.
- Access to modify CI/CD settings in your Git repository.

Configuration

1. **Integrate with your CI/CD pipeline:**

   To use the TuxSuite Component with your Linux kernel project, add the reference kernel pipeline YAML file to your project. This can be done by setting the CI/CD configuration in your project settings(Settings -> CI/CD -> "CI/CD configuration file"):

   ```
   .gitlab-ci-kernel.yml@Linaro/components/tuxsuite
   ```

   Reference YAML file: .gitlab-ci-kernel.yml

   Set the timeout value for the pipeline to 6h since some builds and test run for longer duration. The timeout needs to be adjusted if the jobs timeout. This can be done by setting the CI/CD configuration in your project settings(Settings -> CI/CD -> General Pipelines -> Timeout)

2. **Submit Custom TuxSuite Plan:** The default TuxSuite Plan that is run on the kernel is here. It builds the kernel and runs ltp-smoke test on QEMU using gcc compiler for these architecture: armv5, armv7, arm64, i386, mips, powerpc, riscv, sparc and x86_64.

   The user can choose to host their own TuxSuite Plan covering the tests suitable for their usecase and pass it to the pipeline. We will pass a TuxSuite Plan to run kselftest-cgroup during git push as shown below:

   ```
   git push -o ci.variable="PLAN=https://gitlab.com/Linaro/tuxsuite/-/raw/master/examples/kselftest-cgroup-test.yaml?ref_type=heads"
   ```

TuxSuite Gitlab Component Features

- Default Plan builds and boot test
  - Latest toolchain gcc-(11/12/13) available for the architecture
  - QEMU - arm, arm64, x86/i386, riscv, mips, sh, s390x, ppc, sparc64

- User defined Plan Support to boot test on rpi

```
$ git push -o
ci.variable="PLAN=https://people.linaro.org/~vishal.bhoj/bcm2711-rpi-4-b-build-
boot-plan.yaml" -f origin hw
```

- TuxSuite Local Execution
- TuxSuite Cloud Execution

Demo Pipeline on Linux next tree

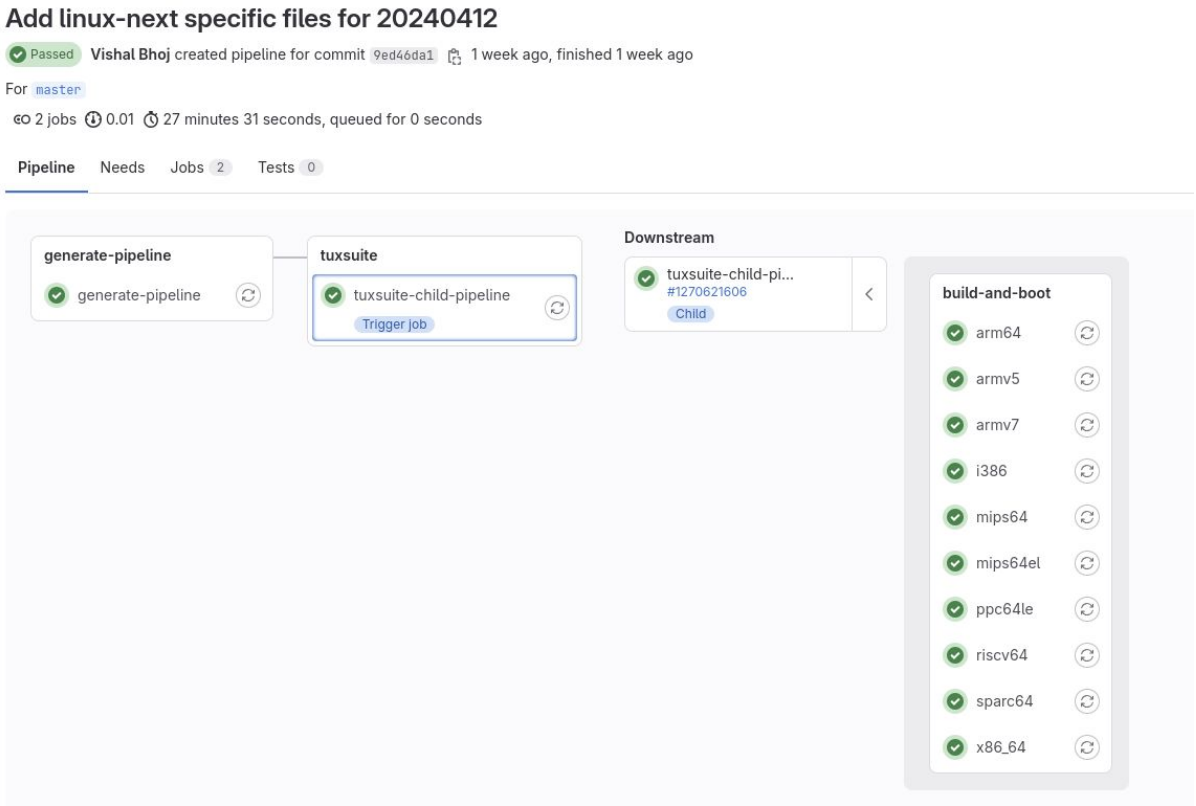
Vishal Bhoj / linux-next / **Pipelines**

All 100 | Finished | Branches | Tags | Clear runner caches | CI lint | Run pipeline

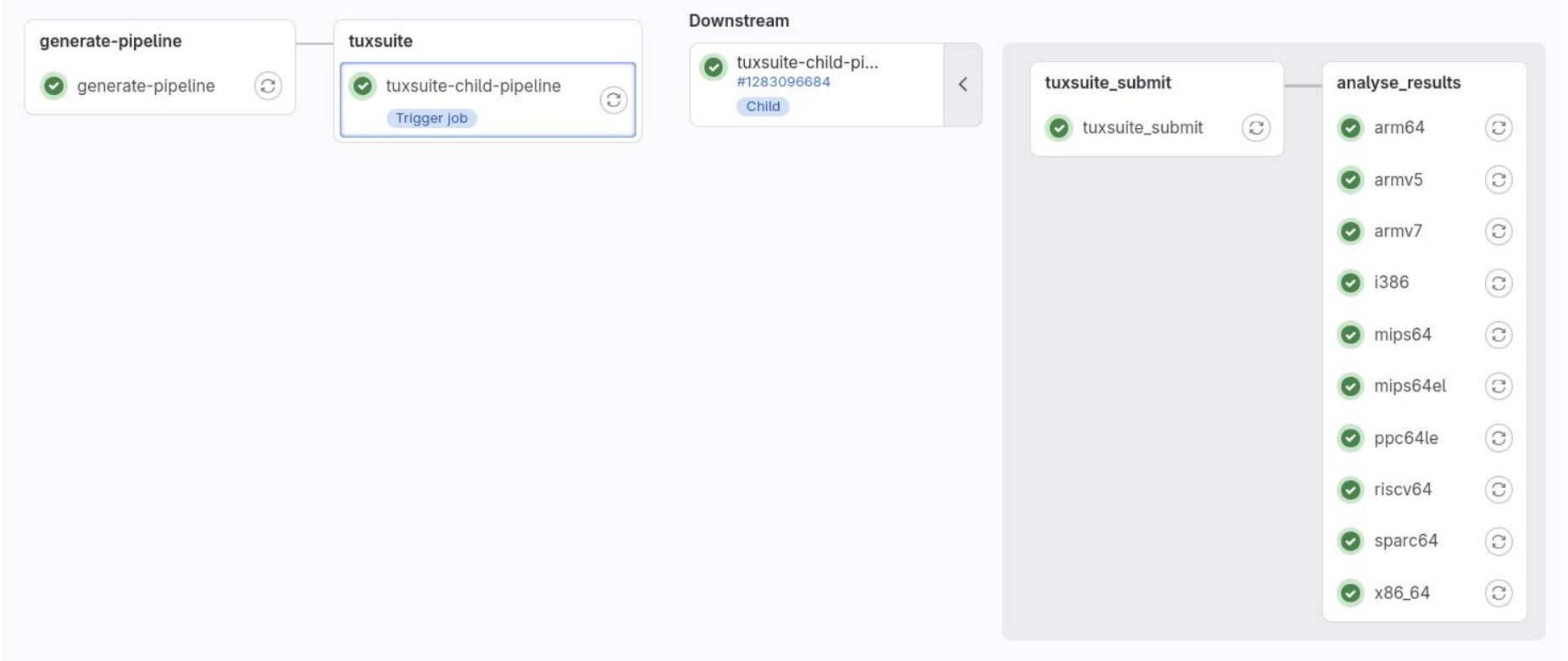
Filter pipelines | Show Pipeline ID

Status	Pipeline	Created by	Stages
Passed 00:22:16 2 hours ago	Add linux-next specific files for 20240508 #1283096117 master e7b4ef8f latest		
Failed 02:00:15 1 day ago	Add linux-next specific files for 20240412 #1281431463 test 9ed46da1 latest		
Failed 03:00:46 1 day ago	break kernel #1281336100 test c432f115		
Failed 03:00:45 1 day ago	Add linux-next specific files for 20240412 #1280555205 test 9ed46da1 latest		

Pipeline with TuxSuite local execution



Pipeline with TuxSuite Cloud Execution



But, what about running specific test ?

- Define custom Plan and submit it.
- TuxSuite Plan file from clangbuiltlinux to run all builds against korg-clang-14 toolchains

```
$ git push -o
ci.variable="https://people.linaro.org/~vishal.bhoj/clang-builtlinux-plan.yml"
-f origin test-2
```

- A single Plan with 56 builds

GitLab <gitlab@mg.gitlab.com>
to vishalbhoj

✖ Pipeline #1281432943 has failed!

Project	Vishal Bhoj / linux-next
Branch	test
Commit	9ed46da1 Add linux-next specific files for 20240412 Sig...
Commit Author	Stephen Rothwell

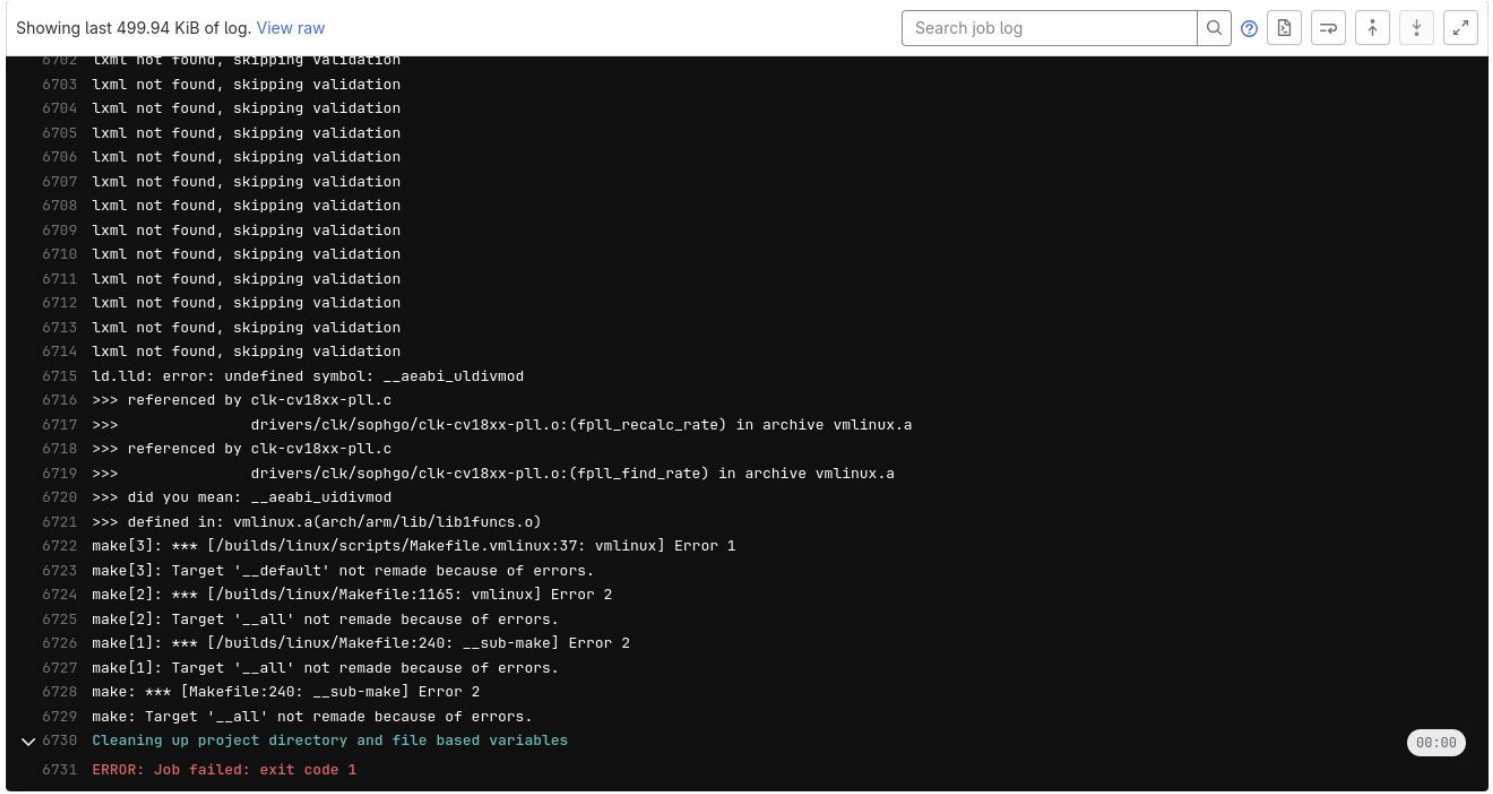
Pipeline #1281432943 triggered by Vishal Bhoj
had 1 failed job

Failed job

✖ analyse_results allconfigs

GitLab

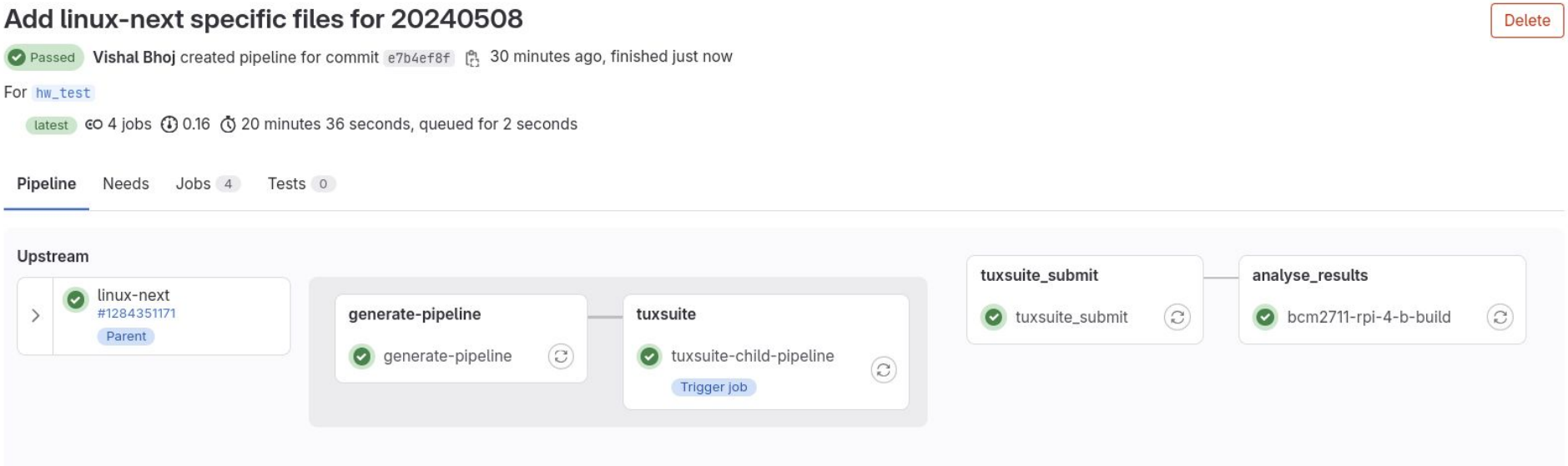
Failure logs in a gitlab pipeline



Testing on Rpi4

- User defined Plan Support to boot test on rpi

```
$ git push -o
ci.variable="PLAN=https://people.linaro.org/~vishal.bhoj/bcm2711-rpi-4-b-build-
boot-plan.yaml" -f origin hw
```



Gitlab minutes story...

- Please fork the linux kernel from Gitlab backed Linux kernel Tree
  - https://gitlab.com/linux-kernel/linux
- 400 free Gitlab minutes(units) for CI/CD pipelines Monthly / User.
- Gitlab provides additional minutes on forks of Gitlab backed Open Source project.
- The users get additional minutes by forking their development tree from the above kernel tree.
- Users can add their own machine as Gitlab Runners
  - https://docs.gitlab.com/runner/register/

Advantages of TuxSuite Cloud Services

- Scalability
- Access to wide variety of physical test target hardware
  - Rpi , db845, rb5, Graviton 3 and many more
- FVP test targets
- Easily Integrate with Remote Labs Hosted inside OEM’s infrastructure
- Email Reporting from TuxSuite
- True parallelism

Future Work

- We are looking for Beta Testers
- Need help from the kernel Community to understand useful Tests to enable.
- Enable the tuxsuite component on other CI infrastructures
  - CodeLinaro, GitHub Action, Jenkins, Jupyter Notebook, ...
- Enable other toolchains hosted on kernel.org
- Add static analyser tools
- Plan Repository
- Dynamically select plan based on patches in the subsystems
- Analytics across large test plans

For Feedback and Queries

tuxsuite@linaro.org

References

- https://docs.tuxsuite.com/plan/kernel
- https://docs.tuxsuite.com/plan/kernel/subcommands/#execute
- https://tuxmake.org
- https://tuxrun.org
- TuxSuite Cloud access Request
- How to use TuxSuite plan locally?

Linaro Connect
MADRID 2024 | MAY 12-17 2024

Thank you