linaro™

Arm Solutions at Lightspeed

Faster Linux Kernel Testing with Linaro's Open-Source Tools and LKFT Automation

Anders Roxell
anders.roxell@linaro.org

CI Systems for Linux Kernel Testing

Why Kernel Testing Still Sucks:

- CI is fragmented
- Testing setup are painful
- Feedback takes too long
- Reproducibility is hard
- Many devs still don’t test enough

Problem we are trying to address

For developers:

- Testing is hard to fit into daily work
- Setup is slow and manual
- Feedback takes too long

For testing reliability:

- Limited access to broad test coverage
- Reproducibility is inconsistent

Linaro's Open Source Projects with Testing Focus (1)

- TuxMake
  - CLI tool to build test the Linux Kernel with multiple toolchains
- TuxRun
  - CLI tool for testing Linux on virtual devices, using curated Rootfs and Test Suites
- TuxLAVA
  - A CLI tool to generate the LAVA Job definition easily with curated Rootfs and Test Suites
- TuxSuite CLI
  - CLI tool to submit builds/tests to TuxSuite Cloud Service
- TuxTrigger
  - A CLI tool to monitor remote tree and trigger a TuxSuite Plan on updates
- Tuxpkg
  - Release automation tool for Python projects, creates pypi, deb and rmp pkgs
- TuxBake
  - A CLI wrapper around bitbake tool to make it easier to build OE/Yocto

Linaro's Open Source Projects with Testing Focus (2)

- LAVA
  - Is a scheduler to schedule test jobs on physical and virtual hardware
- Test-definitions
  - A set of testing scripts designed to work with LAVA and standalone
- SQUAD
  - A Software Quality Dashboard, store's all the test results and logs
- lavacli
  - Is a command line tool to interact with one or many LAVA instances using XML-RPC
- SQUAD Client
  - Is a tool for accessing data from a SQUAD instance through its API. The main purpose of this tool is to ease report customization

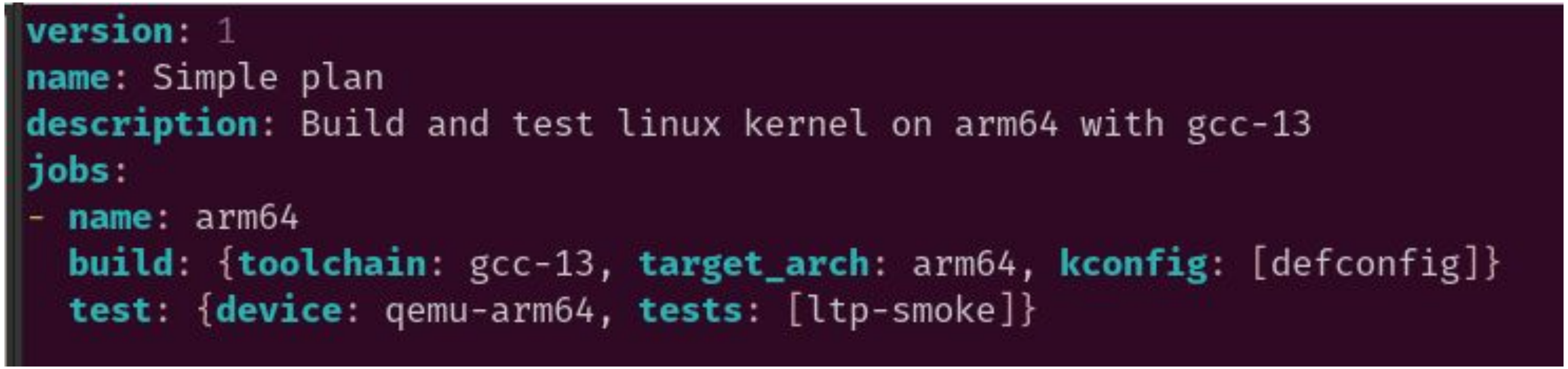
LKFT (Linux Kernel Functional Testing) Linux Stats - 2024

- LTS Releases: 271
- Regressions: 116
- Total Tests: 204,487,984
- Kernel triggers: 1,229
  - Builds: ~400 builds, ~2500 boots
- Builds: ~430k
- Boots: ~3,000k

https://stats.lkft.linaro.org/

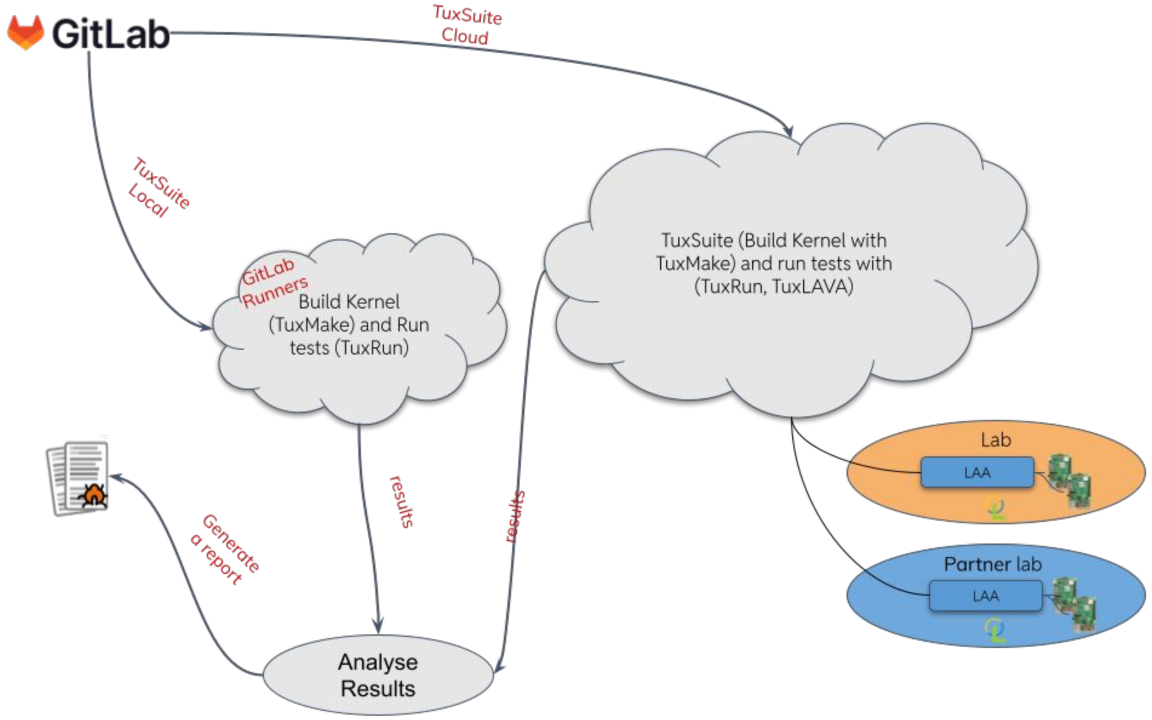
What is a TuxSuite Plan?

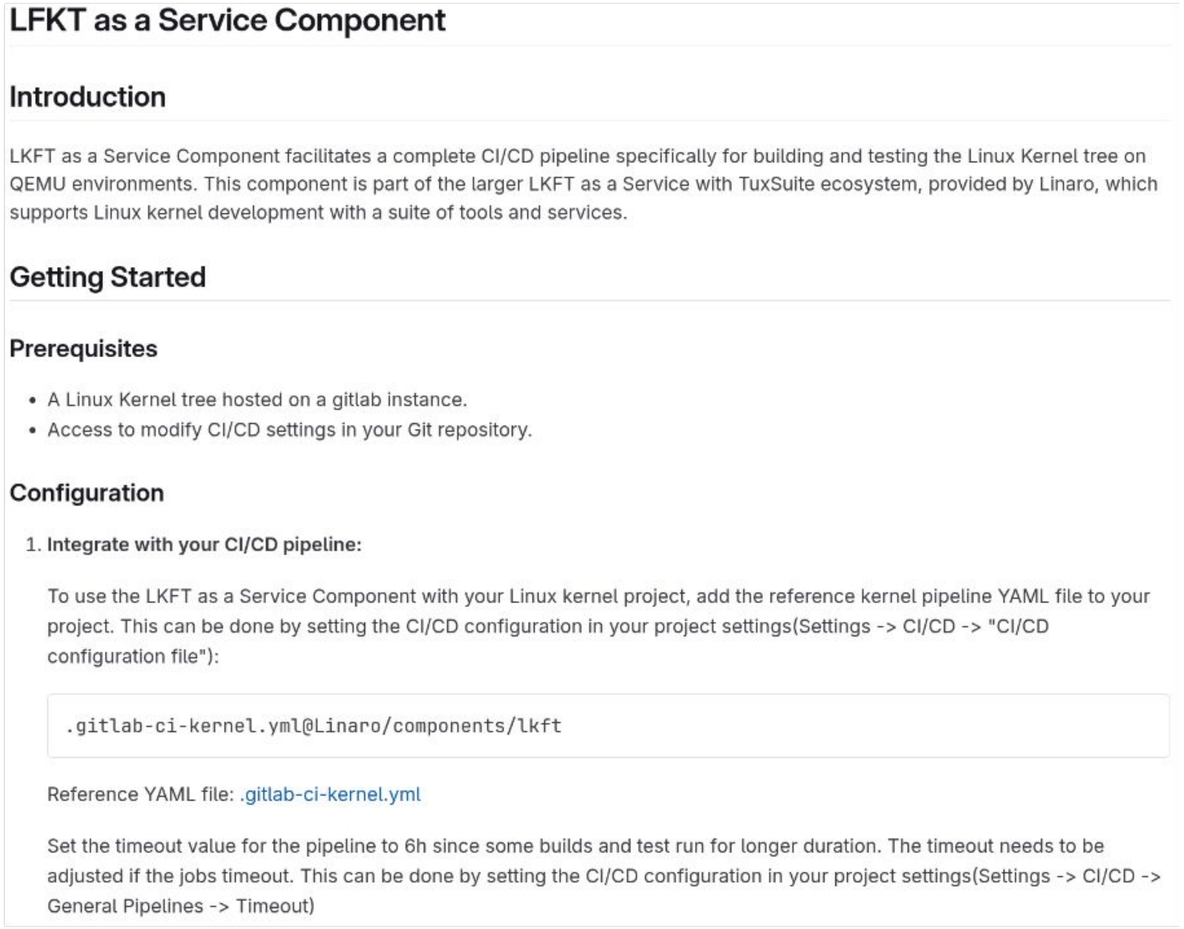
- A Yaml file describing a combination of builds and tests.
- A plan could be made of
  - standalone builds
  - standalone tests
  - set of builds and set of tests to be run for each build
  - combination of all the above



```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]}
```

LKFT GitLab Component


GitLab
TuxSuite Cloud
TuxSuite Local
GitLab Runners
Build Kernel (TuxMake) and Run tests (TuxRun)
TuxSuite (Build Kernel with TuxMake) and run tests with (TuxRun, TuxLAVA)
Lab
LAA
Partner lab
LAA
results
results
Generate a report
Analyse Results



LFKT as a Service Component

Introduction

LKFT as a Service 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 LKFT as a Service with 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 LKFT as a Service 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/lkft
   ```

   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)

https://gitlab.com/Linaro/components/lkft

LKFT Gitlab Component Features

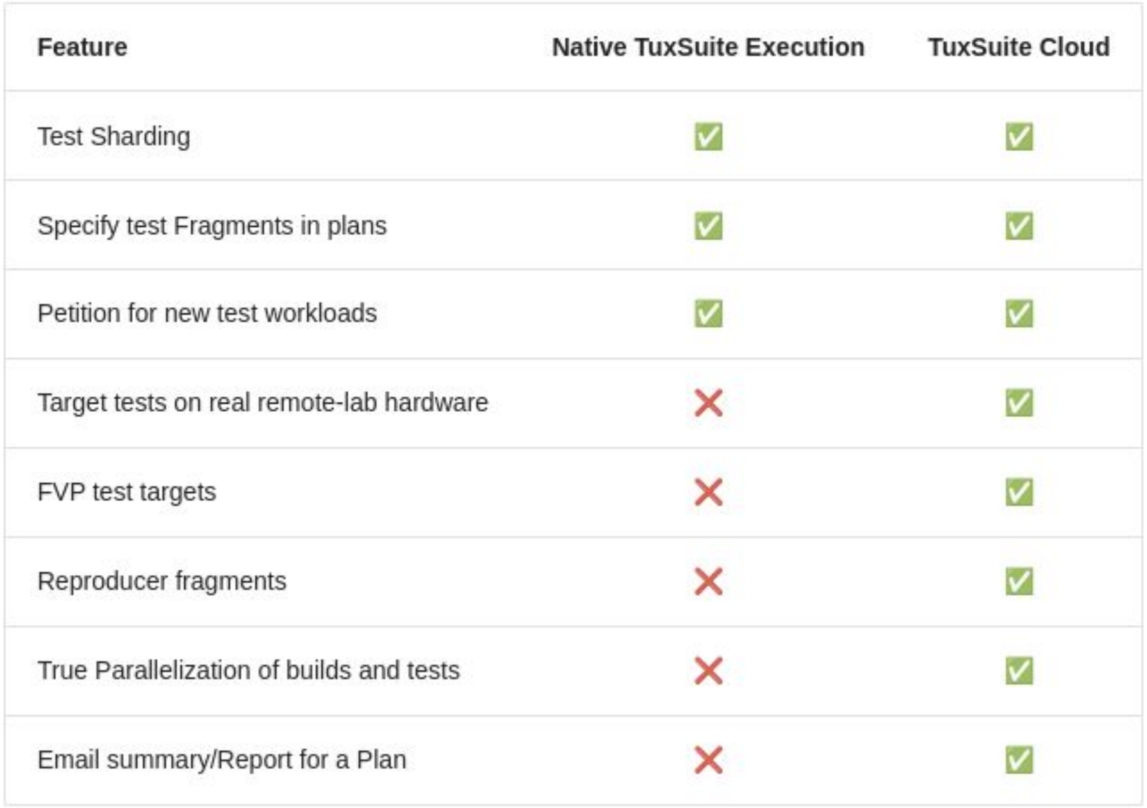
- This is the easiest way to plug into LKFT's testing power
- Default Plan builds and boot test
  - Latest toolchain gcc-(11/12/13) available for the architecture
  - QEMU - arm, arm64, x86_64, i386, riscv, mips, sh, s390x, ppc, sparc64

- RockPi4 example
  - You can override the plan via Git push options. No YAML edits needed – just a git push -o

```
$ git push -o
ci.variable="PLAN=https://people.linaro.org/~anders.roxell/demo-plans/rockpi4-preempt-rt.yaml"
-f origin hw
```

- LKFT Local Execution
- LKFT Cloud Execution

Local vs Cloud Execution

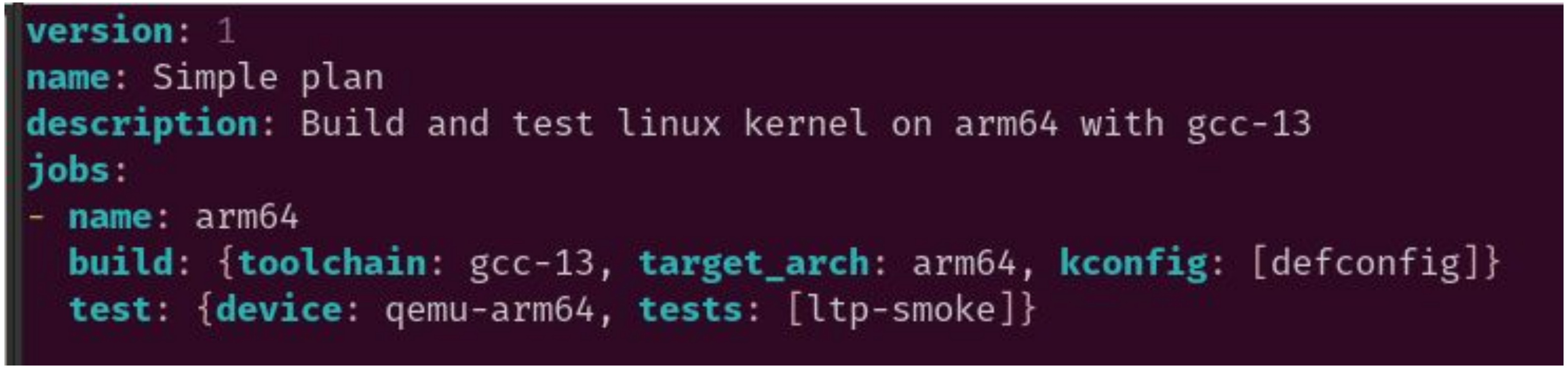


Feature	Native TuxSuite Execution	TuxSuite Cloud
Test Sharding	✅	✅
Specify test Fragments in plans	✅	✅
Petition for new test workloads	✅	✅
Target tests on real remote-lab hardware	❌	✅
FVP test targets	❌	✅
Reproducer fragments	❌	✅
True Parallelization of builds and tests	❌	✅
Email summary/Report for a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

Cloud Execution

and obviously: scalability!

Bisecting locally with 'tuxsuite plan execute'



```
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]}
```

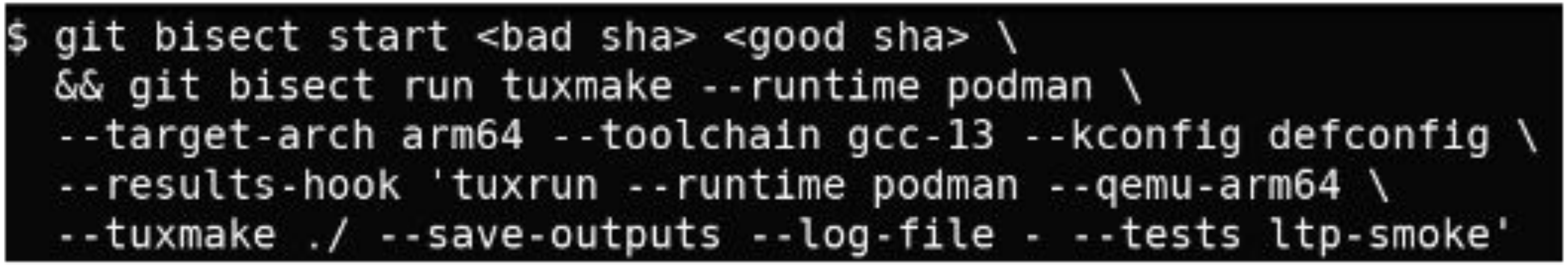
$ cd /path/to/kernel-tree

$ git bisect start <bad sha> <good sha> && git bisect run tuxsuite plan execute --job-name arm64 ~/src/components/lkft/templates/boot/plan.yml

More information: How to bisect with tux tools

Bisecting with TuxMake & TuxRun

- Bisect with it



```
$ git bisect start <bad sha> <good sha> \
  && git bisect run tuxmake --runtime podman \
  --target-arch arm64 --toolchain gcc-13 --kconfig defconfig \
  --results-hook 'tuxrun --runtime podman --qemu-arm64 \
  --tuxmake ./ --save-outputs --log-file - --tests ltp-smoke'
```

More information: [How to bisect with tux tools]

Time For Lunch!

Want to dig deeper into any of the tools, workflows, or LKFT?

Let’s chat after the talk or drop me a line.

Anders Roxell

anders.roxell@linaro.org