
# Fuego
# Status and Roadmap
## December 2017

Tim Bird

Fuego Maintainer

Sony Electronics


# Fuego

# Status and Roadma

## December 2017

Tim Bird

Fuego Maintainer

Sony Electronics


# Outline

Introduction

Status

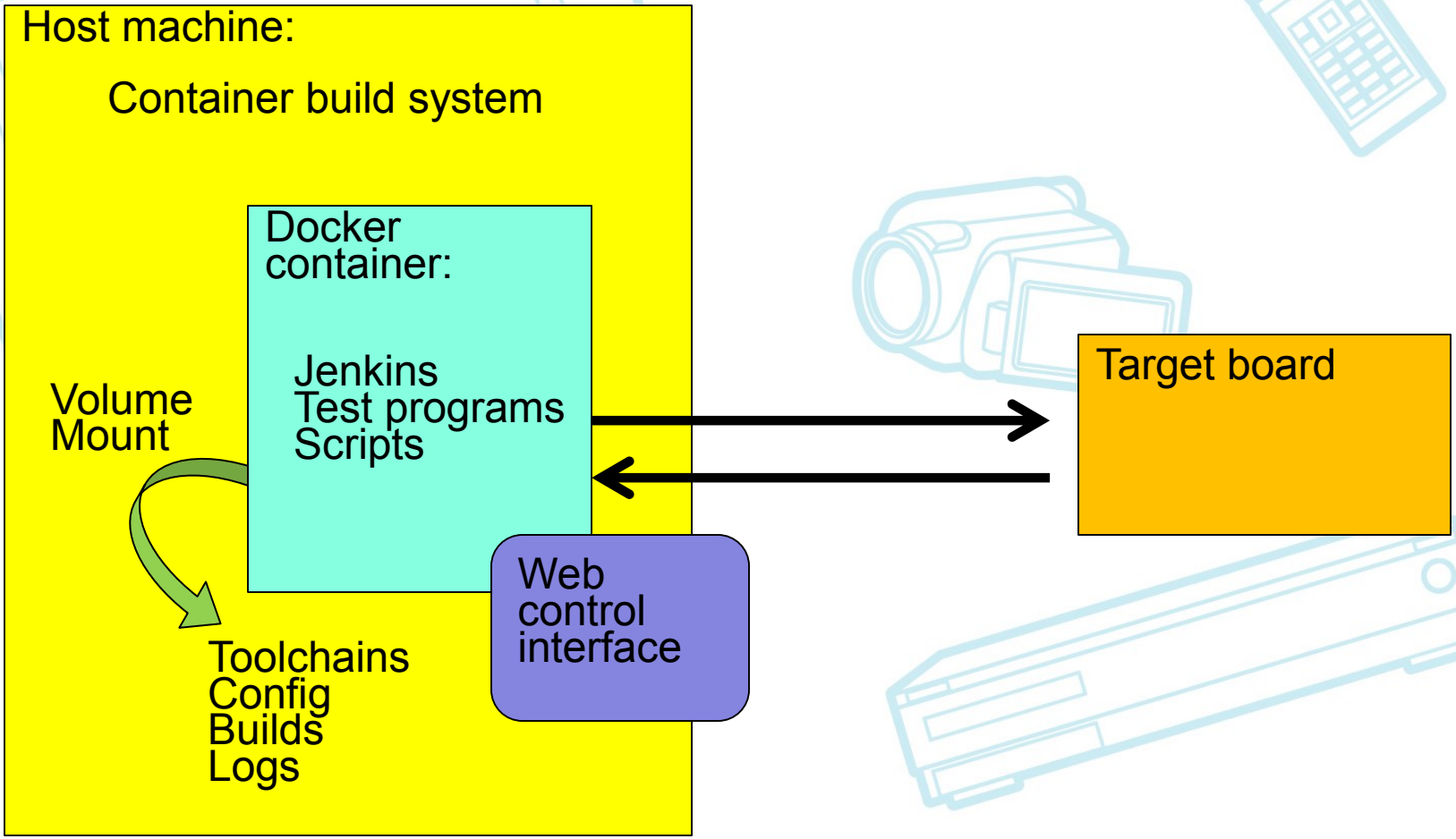
Projects

Roadmap

# Micro-Introduction

Fuego = (Jenkins + host scripts + pre-packaged tests) inside a container

# Architecture Diagram



# Vision – super high level

Do for testing
what open source
has done for coding

- Significant parts of the test process are unshared, ad hoc, private, etc.
  - For no good reason – most QA doesn't need to be proprietary
  - There are OSS frameworks and test programs but parts are missing to create a open testing community.
- Fuego Goal:
  - *Promote the sharing of tests, test methods, and results, the way code is shared now*
    - Make it easy to create, share and discover tests
    - Make test results easy to share and evaluate

# Core principles

- Actually finds bugs
- Allows sharing
- Usable by wide audience
  - Minimal requirements
  - Customizable
- Applicable to embedded
- Easy to use
- Scalability via decentralization


# Outline

Introduction

Status

Projects

Roadmap

# Status

- 1.2.0 (“Combustion”) released Oct 12, 2017
  - Lots of work over the summer for this release
- 1.2.1 released Nov 15, 2017
  - Bugfixes and cleanup
  - also Functional.kselftest
- New web site:
  - http://fuegotest.org/
  - wiki: http://fuegotest.org/wiki
- Miscellaneous projects

# pre-1.2 Feature list

- Jenkins front end
  - Also has a command line interface (“ftc”)
- Containerized
- Overlay system, for customization
  - Boards, distros, specs, plans
- Build system
- Tests are driven from host
- Multiple Transports
- Collection of Tests
- Results parsing and post-processing

# Version 1.2.1 Features

- Unified Output Format
- Test dependency system
- Complex pass criteria handling
- Dynamic board variables
- Charting refactoring
- Test source from git repositories
- Transport modifications
- Test improvements

# Unified Output Format

- Every test creates a “run.json” file
- Has meta-data for the test run, as well as results
  - meta-data:
    - start time, board, kernel version, etc.
- Test results are organized into:
  - test sets
  - test cases
  - measurements (numeric results)
- Format is modeled after KernelCI API
- Purpose is to allow consistent handling of test results

# Test dependency system

- Test declares pre-requisites in fuego_test.sh
- Fuego evaluates dependencies and aborts test if they are not met
- Expressed in 2 ways:
  - NEED_ variables
    - NEED_MEMORY, NEED_KCONFIG, etc.
  - test_pre_check – arbitrary code
    - Usually sequence of calls to *is_on_target* and *assert_define*
- Purpose is to prevent costly build and execute phases
- Can also use (in future) to select tests for a board

# Complex pass criteria handling

- Pass criteria is expressed in a JSON file
  - Allows for complex results determination
    - e.g. threshold for benchmarks, list of allowed failures
- Can be customized per board
- Can be shared with others
- Can (in future) be written automatically, based on current results
  - e.g. board filesystem performance threshold = current results + 5%
- Purpose is to allow specifying status determination, for complex tests (eg. LTP)
  - Not all tests can be expected to pass

# Dynamic board variables

- Users and Fuego can add additional board variables at runtime
- Saves persistent information about a board
- Automatically included in test variables for future tests
- Purpose is to allow communication between tests, and automated test customization
  - eg. Functional.kernel_build could populate FUEGO_KERNEL_CONFIG_PATH

# Charting refactoring

- Fuego charting was changed in 1.2 release
- Mostly internal reorganization of code
  - Some changes to support future report generation features
- Now support 3 chart types:
  - HTML table of testcase results
  - HTML table of test set (aggregate) results
  - Javascript plots of measurement data (for benchmark)
- Create “chart_config.json”, to allow for control of default visualization for test

# Test source from git repositories

- Can retrieve test source from git repositories
  - Previously, only source from tarball was supported
- Specify source with:
  - gitrepo and gitref variables
    - gitref indicates a particular commit, tag or version
- Purpose is to provide greater flexibility and easier maintenance for managing test source
- Note: This requires external network access during the test build
  - Source from integrated tarballs doesn't require this

# Transport modifications

- New overlay functions for connect and disconnect to board
  - ov_transport_connect
  - ov_transport_disconnect
- Can be used for session setup and teardown
- Can also be used for provisioning the board or for reservation in an external system (eg. LAVA)

# Test improvements

- Added support for aarch64 toolchains
- Added dependency information to some tests
- Some older tests were fixed to address build issues with newer toolchains
- The source for some test programs was updated to newer versions
- Parsers were added for some Functional tests, to give results for individual testcases
- Parser improvements in general
- Lots of other bugfixes were made

# LTP test improvements

- Support for pre-installed LTP test binaries
  - In case you already have LTP packaged in your embedded distribution
- Simplified support for pre-installing Fuego LTP binaries
  - Significantly reduces test execution time, by skipping deploy phase
- Allow re-use of code between LTP test jobs
  - Avoid rebuilding the same LTP code, if toolchain and target architecture is the same
  - Reduces disk space for board farms

# Fuego Project Processes

- Communication
  - Mailing list
  - Monthly conference call (AGL/CIAT)
  - Fuego hackathon
- Need contributor guidelines
  - Developer Certificate of Origin (Signed-off-by)
  - Code style guide
    - Mostly indentation (4 spaces, no tabs)
  - Patch submission tips


# Outline

Introduction
Status
Projects
Roadmap

# Projects

- Board automation standards
- Linux Foundation funding
  - Fuego release self-test
  - Fuego Test Server
- China hackathon
- Japan hackathon
- Features in progress

# Board automation standards

- Presentation at Linaro Connect
  - See http://fuegotest.org/ffiles/Test-Standards-LC-2017.pdf
- Lots of meetings at ELCE on this
  - Pengutronix introduced labgrid
  - Linutronix demonstrated r4d and libvirt
  - BOF resulted in some collaboration:
    - See https://elinux.org/Board_Farm
    - Mailing list for discussion:
      - https://lists.yoctoproject.org/listinfo/automated-testing
- Please join this discussion

# Linux Foundation Funding

- Release self-test (funded)
  - To use Fuego to do continuous integration for itself, and for release testing
- Test server hardware (funded)
  - To replace virtual machine with dedicated hardware
- LAVA test-level integration (not funded)
- Documentation conversion (not funded)

# China Hackathon

- Was held in Shanghai at Fudan University on Nov 3-5
- Fujitsu employees were mentors at the event
  - Liu Wenlong, Bao Fei
- 5 Students worked on Fuego
- See http://fuegotest.org/wiki/HACKxFDU_2017_planning
- Website and logo completed

# Japan Hackathon

- Will be held tomorrow (Dec 2) at Sony headquarters in Shinagawa
- See http://fuegotest.org/wiki/Japan_Fuego_Hackathon_2017

# Features in progress

- Fuego release self-test
  - Implement Fuego release test as a Fuego test
  - Use multiple containers
    - Current container and container-under-test
  - Add support for evaluating web results
    - Compare browser images using Selenium HQ
      - Will help us with other image comparison tests
- Fuego centralized test server
  - Share ad-hoc test (test package)
  - Request test on someone else's board
  - On backburner at the moment


# Outline

Status
Projects
Vision
Roadmap

# Roadmap

Recent past → Near Future → Long Term

# Roadmap

- Recent past:
  - Priority was stuff affecting test API or test packaging
    - Needed before big push for new tests
- Near future:
  - Documentation
    - Conversion to reStructuredText
    - Refactoring
    - Tutorials
  - New tests for AGL, LTSI, CIP
    - What tests to tackle next?

# Roadmap (cont.)

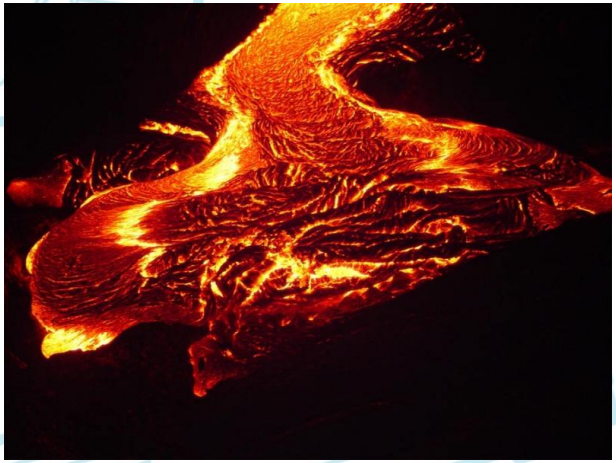
- Near future (cont.):
  - Testplan enhancements
    - Controlled test sequences
      - Similar to Jenkins pipelines
      - Processing multiple steps (provisioning, testing, notifications, report generation) in sequence
    - More fields for plan configuration
  - Report generator and more charting control
    - Now that we have unified output, we can do queries, and different output formats

# Roadmap (cont.)

- Near future (cont.):
  - System provisioning support
    - Install of software under test
      - Has been out-of-scope for Fuego
    - e.g. AGL image deploy, LTSI kernel update, etc.
    - Full automation requires board management API
    - Looking at labgrid as possible solution
- Long-term
  - Distributed test network
  - Hardware testing

# Other Priorities

- LAVA integration
  - We have everything needed for transport integration
  - Need test-level integration
    - Separate build phase
    - Deploy to LAVA server
    - Create LAVA test that does:
      - Execute test on board
      - Collect results



# Resources

- Fuego web server:
  - http://fuegotest.org/
  - wiki: http://fuegotest.org/wiki
- Mailing list:
  - https://lists.linuxfoundation.org/mailman/listinfo/fuego
- Repositories:
  - https://bitbucket.org/tbird20d/fuego
  - https://bitbucket.org/tbird20d/fuego-core


Fuego