



# Fuego Test System Status Summary

June 2018

Tim Bird

Fuego Maintainer

Sony Electronics

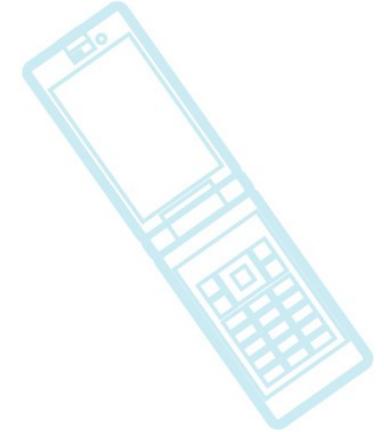
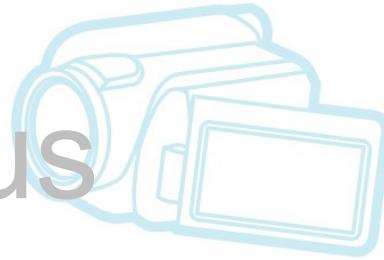
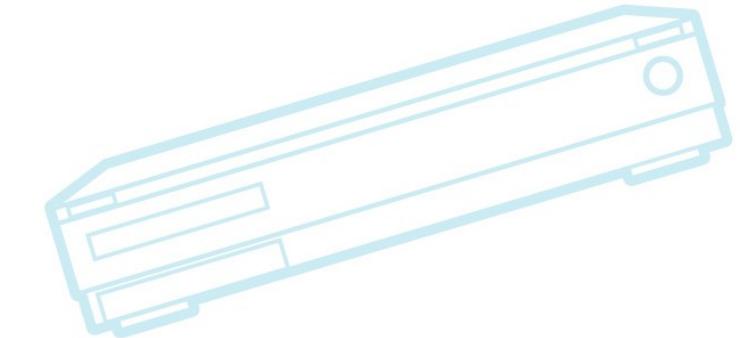
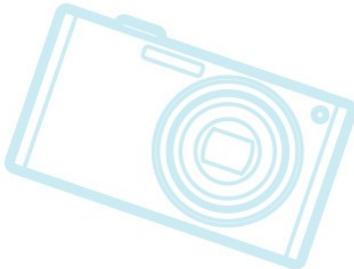


# Outline

Introduction

Feature List

Feature Status



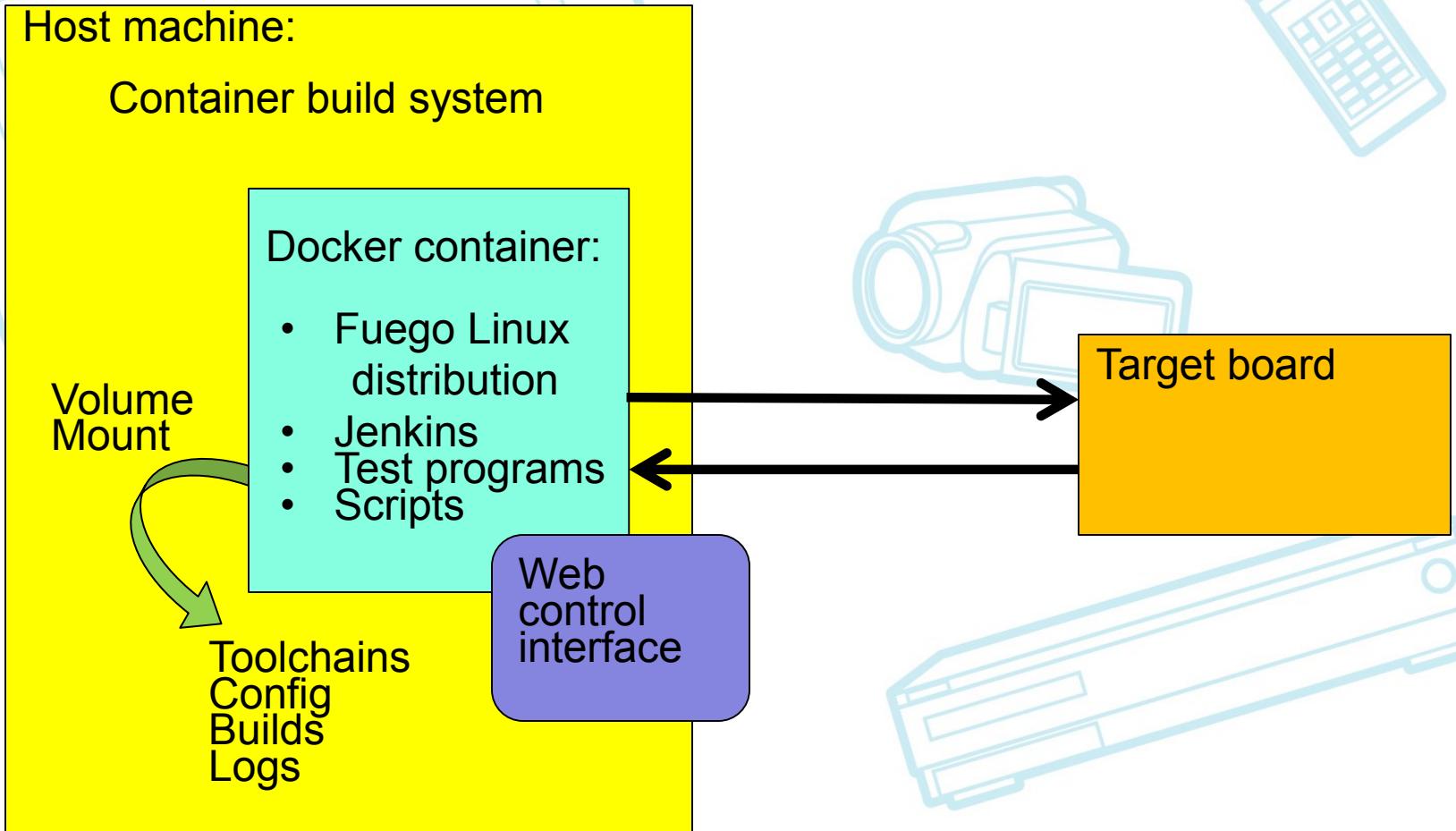


# Micro-Introduction

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



# Architecture Diagram





# Core features

- Distribution of Linux for testing
- Build system
  - Architecture-neutral & inherently cross-platform
- Includes a collection of tests
  - Scripts for test execution
  - Results parsing, analysis, and visualization
- Report generation
- Multiple transports
- Jenkins front end
  - Also has a command line tool

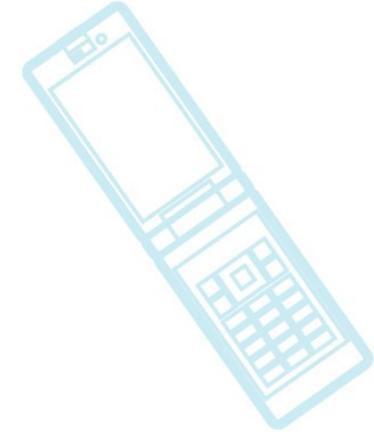
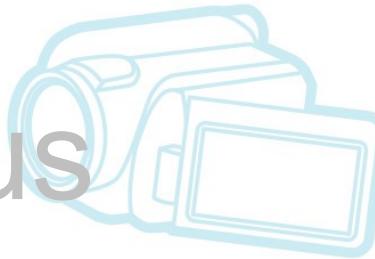
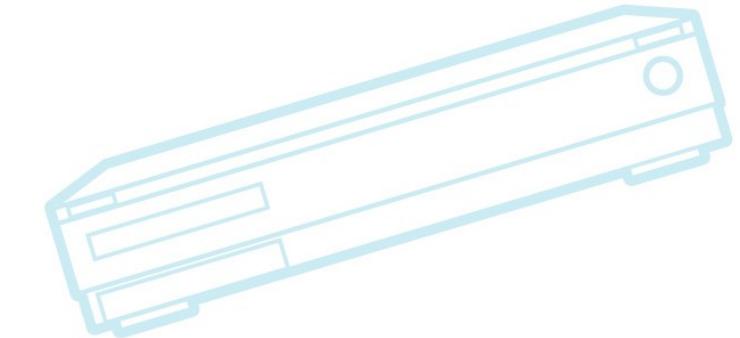
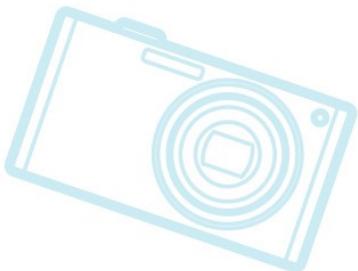


# Outline

Introduction

Feature List

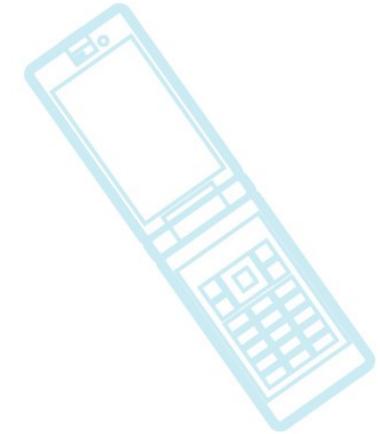
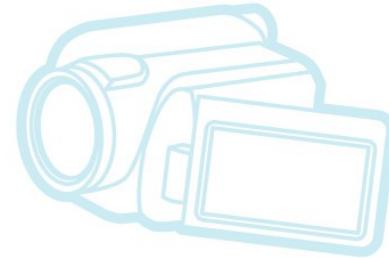
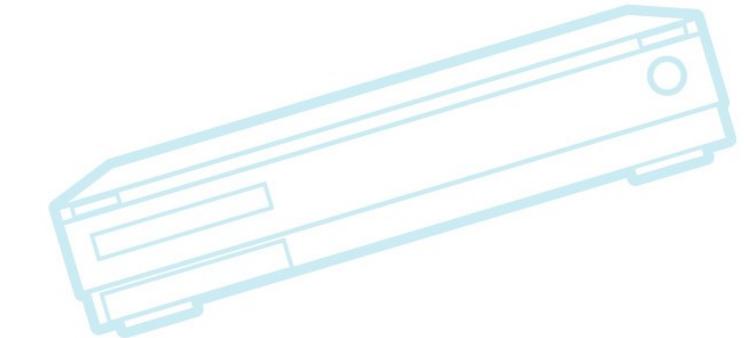
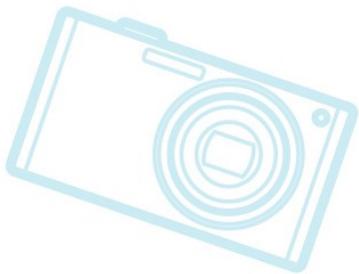
Feature Status





# Slide key

- *Italic header* = reference material





# Feature Areas

- Installation
- Test Execution
- Results Analysis
- User Interface
- Visualization
- Reports
- Tests
- Fuego Distribution
- Infrastructure
- Usability
- Documentation
- Marketing



# Installation

- Docker build
  - Pre-built Docker image (partially completed)
    - Problems with container customization
- Adding a board
- Adding a toolchain
- Populating Jenkins artifacts
- Provisioning setup (**none**)



# *Pre-built docker image*

- Ability to use Fuego without building the docker image
  - Create a pre-built Fuego docker image, and host it at docker.io
  - e.g. “docker run fuego”
- Requires automatic container customization
  - Network proxy
  - User and group
  - Volume mount customization
- Includes refactoring the Fuego directory layout
  - Turned out to be too intrusive for 1.3 release



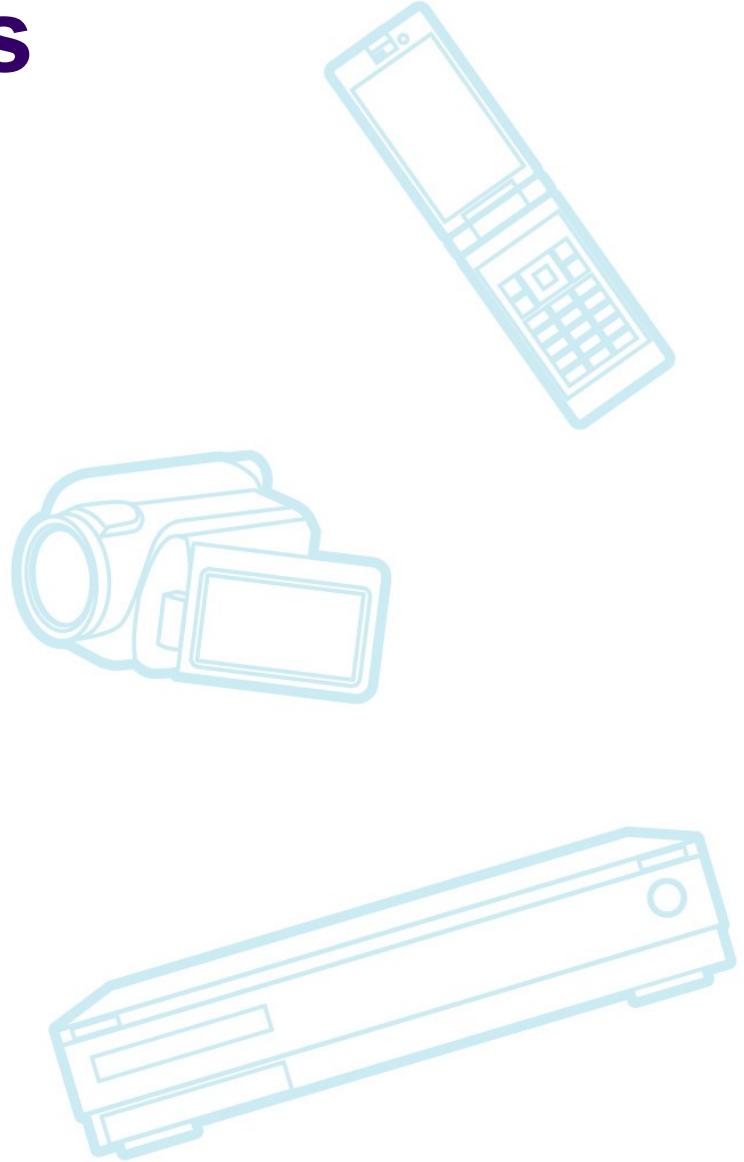
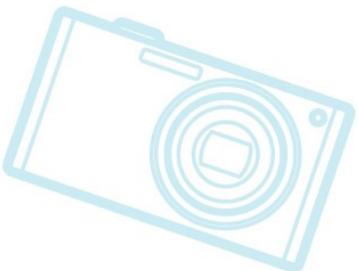
# Execution

- Launch – jenkins job or command line
  - main.sh, ftc run-test
- Test customization (plans, specs, criteria)
- Overlay system (ovgen.py)
- Test dependency system (need\_check.sh)
- Transports (serial, ssh, local, ttc, lava)
- Build system
  - Source retrieval
- Deployment
- Remote execution
- Log retrieval
- Error handling/Abort
- Timeouts



# Results Analysis

- Log parsing
  - Unified Output Format
- log\_compare
- Pass-criteria
  - per-board pass-criteria
- Jenkins history





# User Interface

- Jenkins UI
- Job launching
- ftc to manage Jenkins objects
  - ftc add-node, add-job, add-view
- History, status, result indicators
  - Not very rich (no icon for abort or skip)
- log links, results description
- Failure reason (**missing**)



# Visualization

- Charts
  - Tables
  - Plots (jquery flot module)
  - chart\_config
    - A few items missing from JTA (hide/unhide data sets)
- Log splitting by testcase



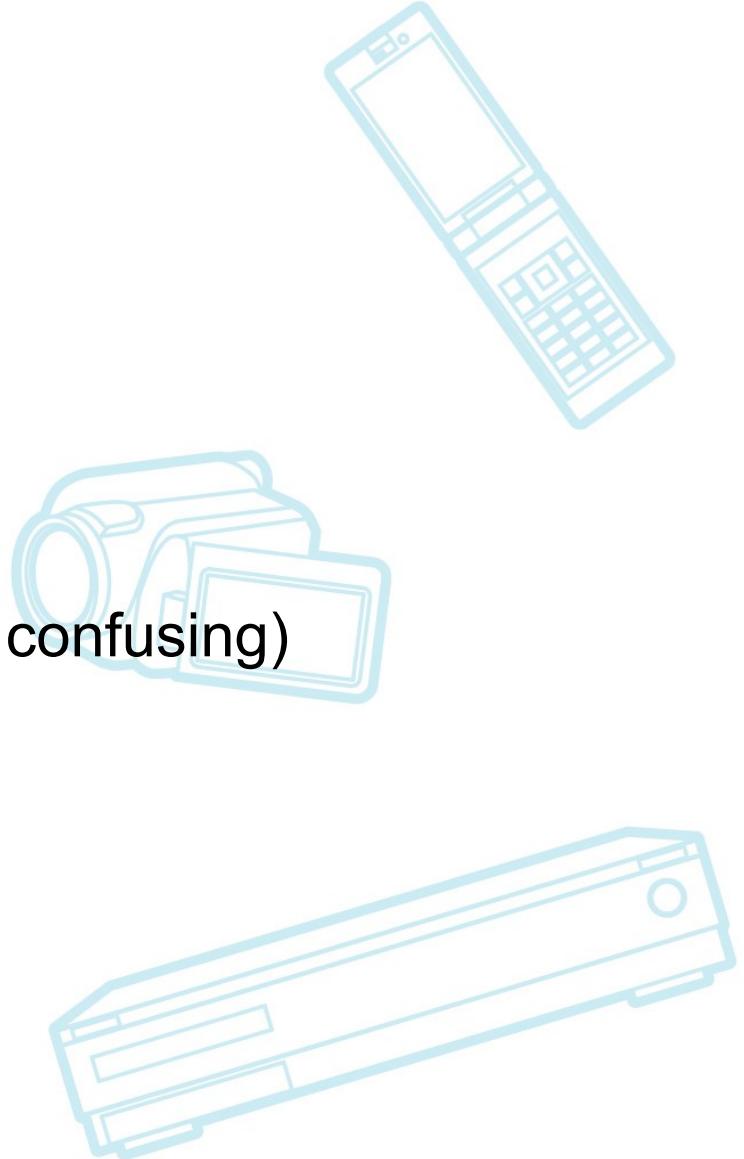
# *Log splitting by testcase*

- Ability to split the test log into pieces, according to testcase boundaries
- Only works for some logs
- Requires slight modification to test's parser.py
- Addition to UI
  - Can click on testcase in Jenkins UI, and see section of log related to that testcase
- Very handy for examining results details



# Reports

- ftc list-runs, gen-report
  - field control
  - output formats
  - --where (filtering)
    - (time comparisons are a bit confusing)
    - (could use a batch id)
- Dynamic documents
  - chart insertion (missing)
  - coverage (very little)





# *Report generation improvements*

- More output formats
  - html, rst, pdf, excel, csv
- Control of fields displayed
  - header\_fields
  - report fields
- More filtering (--where options)
  - especially tguid:result
  - Try this:
    - `ftc gen-report --where "tguid:result!=PASS"`



# Tests

- LTP – see next slide
- kselftest
- Realtime, Filesystem, Network
- POSIX
- System benchmarks
- Security/vulnerability
- autopkg
- ptest
- Fuego self-tests / release test



LTP

- lots of specs
- skiplists
  - Autodetect skiplist items
- supports separate build and manual deploy
- one\_test
  - execute a single test program
- Problems:
  - realtime parsing is not good
  - should separate POSIX from other tests



# New tests

- Realtime benchmarks:
  - backfire, deadlinetest, migratetest, pmqtest, ptsematest, sigwaittest, svsematest
- Other tests (updated or new):
  - dbench4, dd, iperf3, vuls, autopkgtest, year2038
  - LTP\_one\_test
  - fuego\_board\_status
- Fuego self-tests:
  - fuego\_lint, fuego\_tguid\_check, fuego\_ftc\_check
  - fuego\_release\_test



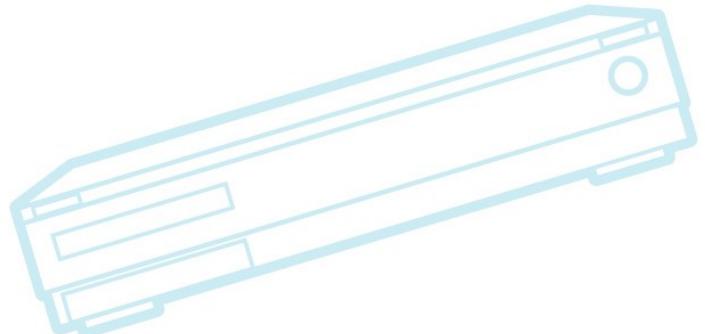
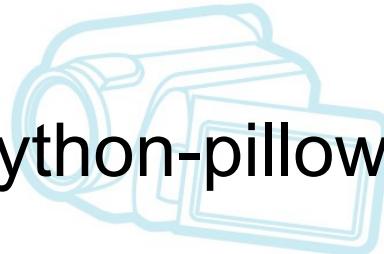
# Fuego release test

- Complicated test to do a full release test
  - Builds docker container
  - Runs docker container for “release under test”, alongside “test-runner” container
- Checks Jenkins web interface
  - Using HTML DOM element checks
  - Using comparisons of web page rendered images
- Adds capabilities to Fuego distribution for testing of other DUT web or image features



# Fuego Distribution Additions

- clitest
  - Tests expected output of commands
- Selenium
  - Web page HTML testing
- Chromium, imagemagick, python-pillow
  - Web page image testing
- pexpect, flake8 (python lint)
  - Fuego self-test
- python-reportlab
  - Report formats
- iperf3
  - Network testing





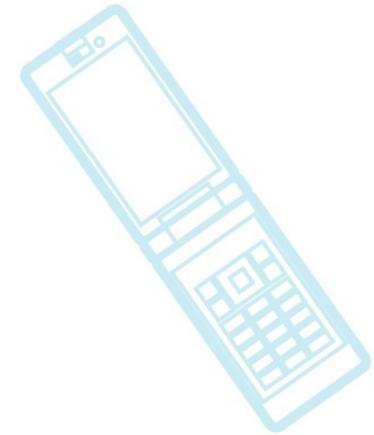
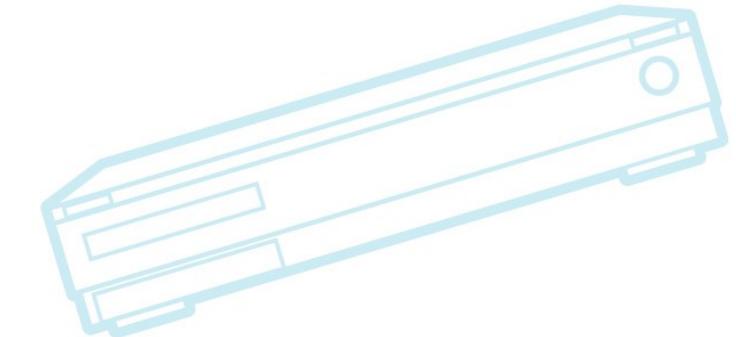
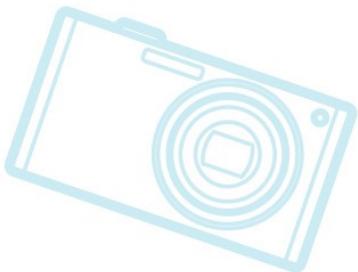
# *Web page and image comparison tools*

- Added Selenium to Fuego distribution
  - For web page testing
- Added Chromium to Fuego distribution
  - For web page rendering automation
- Added tools for:
  - Comparison of returned HTML
  - Web page image capture
  - Image comparison
    - With support for masked regions
- **Note: This is not generalized yet**
  - Need to read Functional.fuego\_release\_test scripts and use as example



# Infrastructure

- bitbucket git repositories
- Hardware board control
- Individual test phases
- ftc outside the docker container





# *Hardware board control*

- General feature is ability to control board under automation
  - Added in 1.3:
    - Hook for hardware board reboot
    - Shows method for adding board control hooks
  - Goal is to support provisioning and other hardware functions, as well
    - ex. off-DUT test hardware control and multiplexing
- Would rather re-use some other board control layer
  - e.g. LAVA, labgrid, Dryad (from SLAV), ttc, etc.



# *Individual test phases*

- Ability to run test phases individually
  - Main purpose is to allow separation of:
    - Test program build (on host)
    - Test execution on target
- Can use ftc option:
  - ftc run-test –p ‘pcb’
    - Executes pre\_test, pre\_check and build phases, then stops
- Can use environment variable
  - FUEGO\_TEST\_PHASES=“pre\_test pre\_check build”



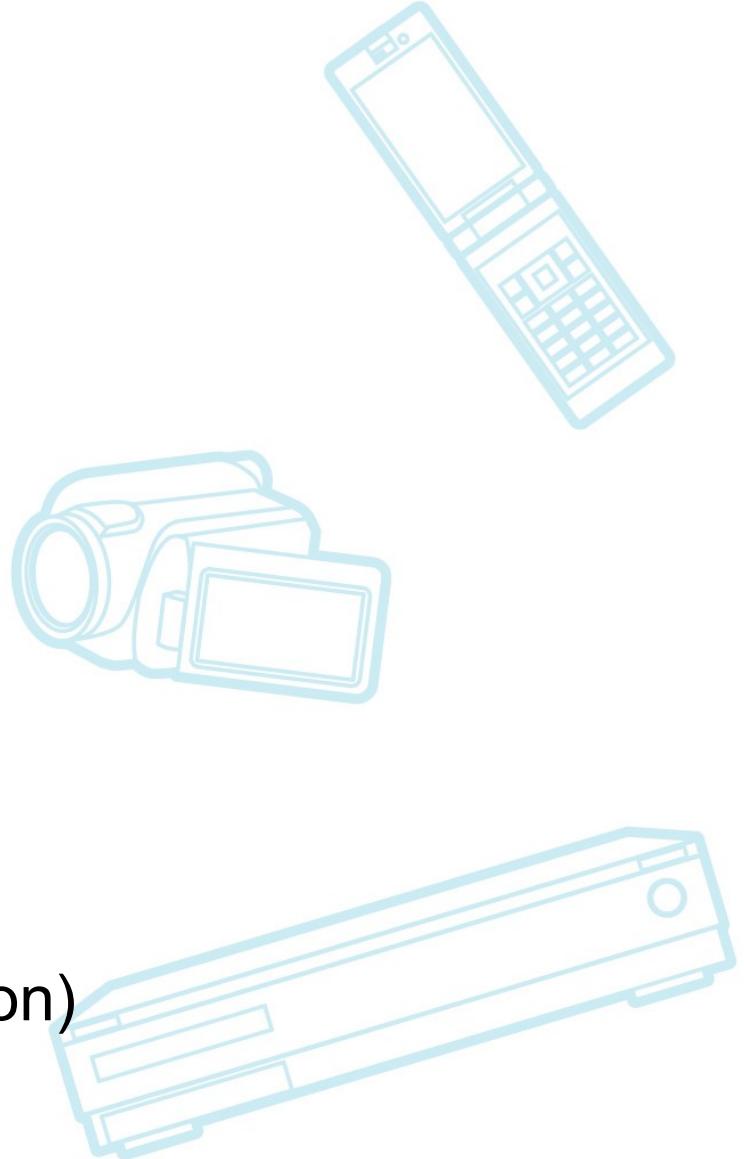
# *ftc outside the docker container*

- docker adds unneeded overhead to some commands
- Some commands can now be run outside the docker container
  - New fuego.conf file to specify directory locations
  - list-runs, gen-report can be done directly on host
- Partial step towards use of low-level Fuego functionality with alternate UIs and frameworks



# Usability

- End user
  - License
  - Web site
  - Ease of installation/use
    - Consistency
    - Obviousness
  - Robustness
    - Error Handling
- Developer support
  - Coding style
  - Languages (bash, python, json)
  - Debugging
  - Documentation





# Contributor guidelines

- Recently added to wiki
- Coding style guide
  - Mostly indentation (4 spaces, no tabs)
  - See [http://fuegotest.org/wiki/Coding\\_style](http://fuegotest.org/wiki/Coding_style)
- License guide
- Patch submission tips
- See  
[http://fuegotest.org/wiki/License\\_And\\_Contribution\\_Policy](http://fuegotest.org/wiki/License_And_Contribution_Policy)



# Documentation

- Installation (quickstart guide)
- End-user docs (wiki, pdf)
  - User's guide (adding...)
  - Walkthroughs (**none**)
  - Jenkins help (**none**)
    - Setting up triggers (needed)
    - Setting up notifications (needed)
- Developer docs (wiki)
  - Architecture
  - API reference
- Tutorials ??
  - (**Some presentations exist, but they're old**)



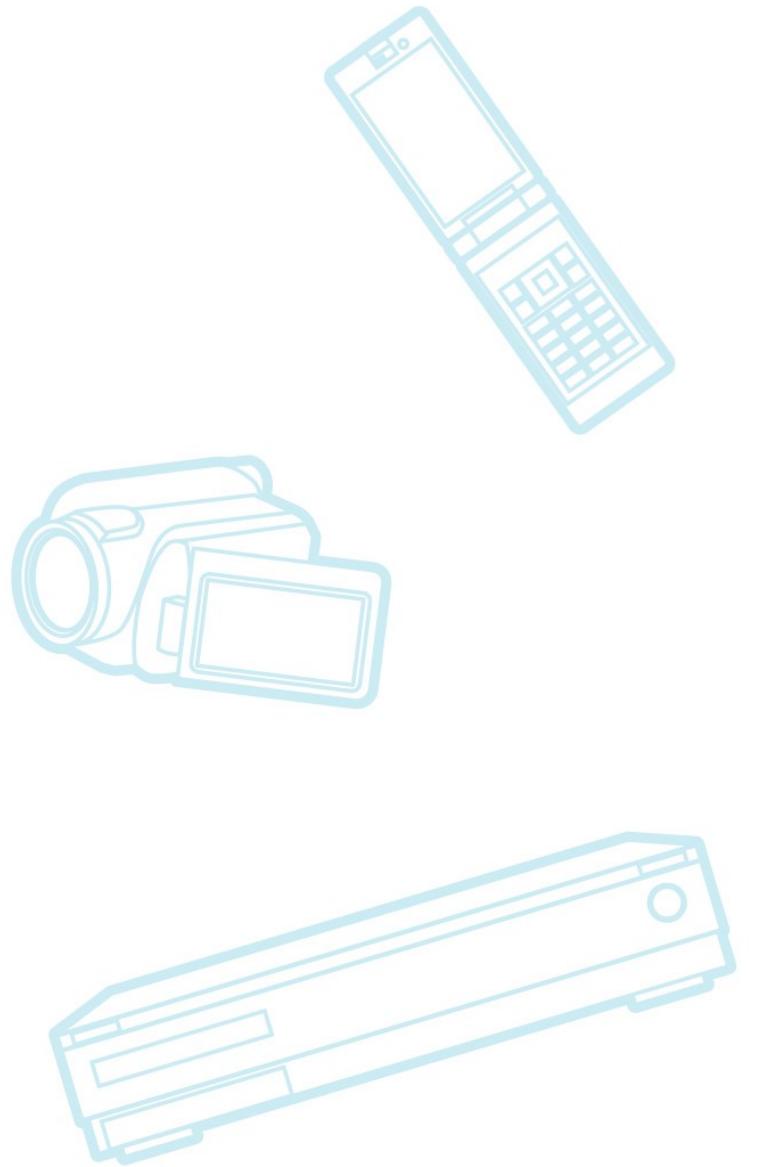
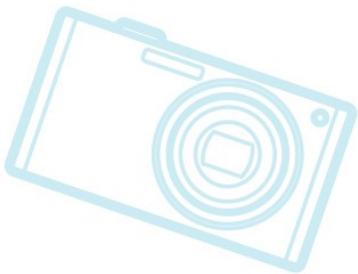
# *Documentation conversion*

- Conversion of docs to reStructuredText
  - Replace PDF and wiki docs with rst
  - Move all docs under source repository
  - Use sphinx to create multiple formats
  - Publish on [readthedocs.io](http://readthedocs.io)
- Made some progress
  - Have sphinx templates in place
- Got stuck on markup conversion
  - Considered automation, but hit some hurdles
- See [http://fuegotest.org/wiki/rst\\_docs](http://fuegotest.org/wiki/rst_docs)



# Marketing

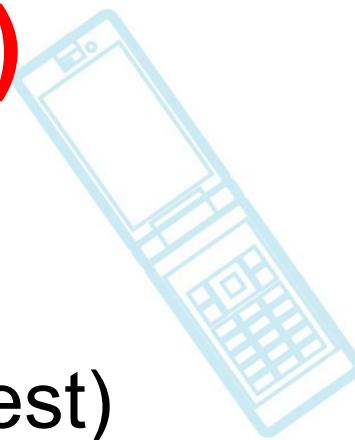
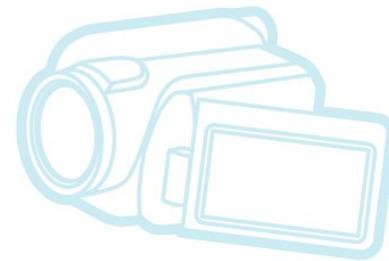
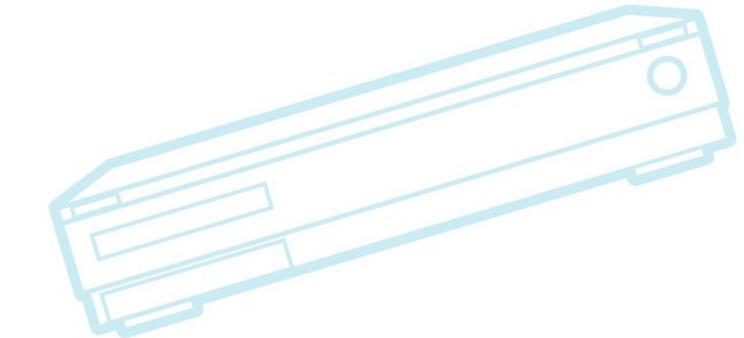
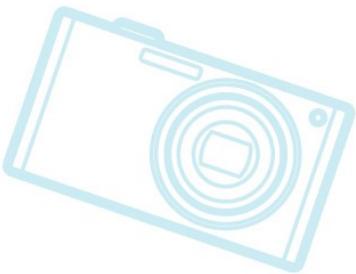
- Outreach
- Visibility
- User growth
  - (slower than expected)





# Test server (prototype)

- Test store (put-test, get-test)
- Distributed test requests (put-request)
- Aggregation
  - Results sharing (put-run)
- Data mining (nothing)



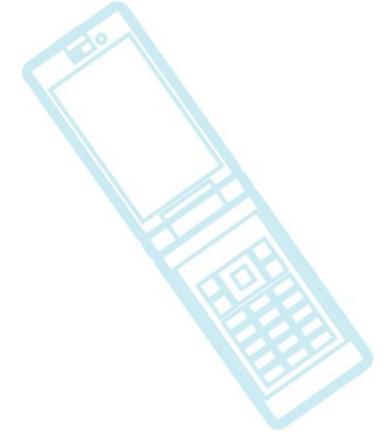
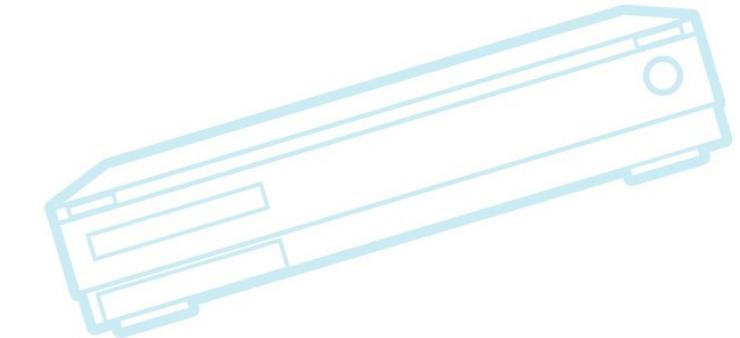
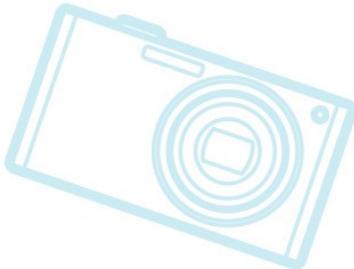


# Outline

Introduction

Feature List

Feature Status





# Feature Status

Feature Area	Status	Potential Improvements
Installation	Pretty Good	Pre-built image, provisioning help
Test Execution	Good	Handle timeouts better
Results Analysis	Very Good	Report of failure reason
User Interface	Good	??
Visualization	Good	Add more chart_config options
Reports	OK	Integration into dynamic docs
Tests	OK	More tests
Fuego Distribution	Good	Ability to load packages as needed
Infrastructure	OK	Move to gitlab
Usability	OK	??
Documentation	OK	Need tutorials and walkthroughs
Marketing	Good	??
Test server	Not Good	Finish implementation, provide server code



# Final Thoughts

- Don't want to focus on negatives
- Fuego is a very rich, capable, flexible system
- An amazing amount of features
  - It's often easy to implement a new feature, because of rich existing infrastructure
  - Can usually support what you want to do
- Very friendly community



**Fuego**