TOSHIBA

Leading Innovation

Recent Toshiba Work on FUEGO

Fuego jamboree 2 (23-June-2018)

Panasonic Laboratory, Tokyo

http://fuegotest.org/wiki/Fuego_Jamboree_2

Daniel Sangorrín (ダニエル)

Open Source Technology Dept.
Corporate Software Engineering & Technology Center
Research & Development Division
TOSHIBA corporation

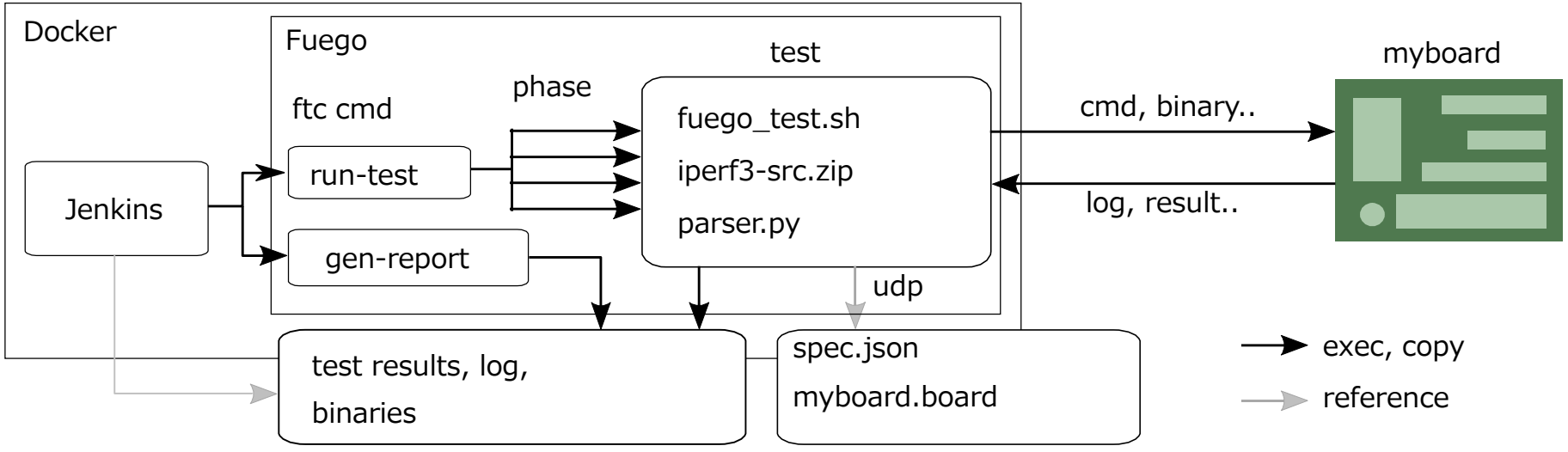
Topics

- Calling Fuego from Jenkins
- Problems with Jenkins
- Visualizing Fuego test results
- FTC code improvements
- Trinity (fuzzing testing)
- Dynamic variables
- Useful scripts
- Ideas

Warning: this information is for developers. A lot of this functionality has not been merged or published yet. Interfaces may change in the future.

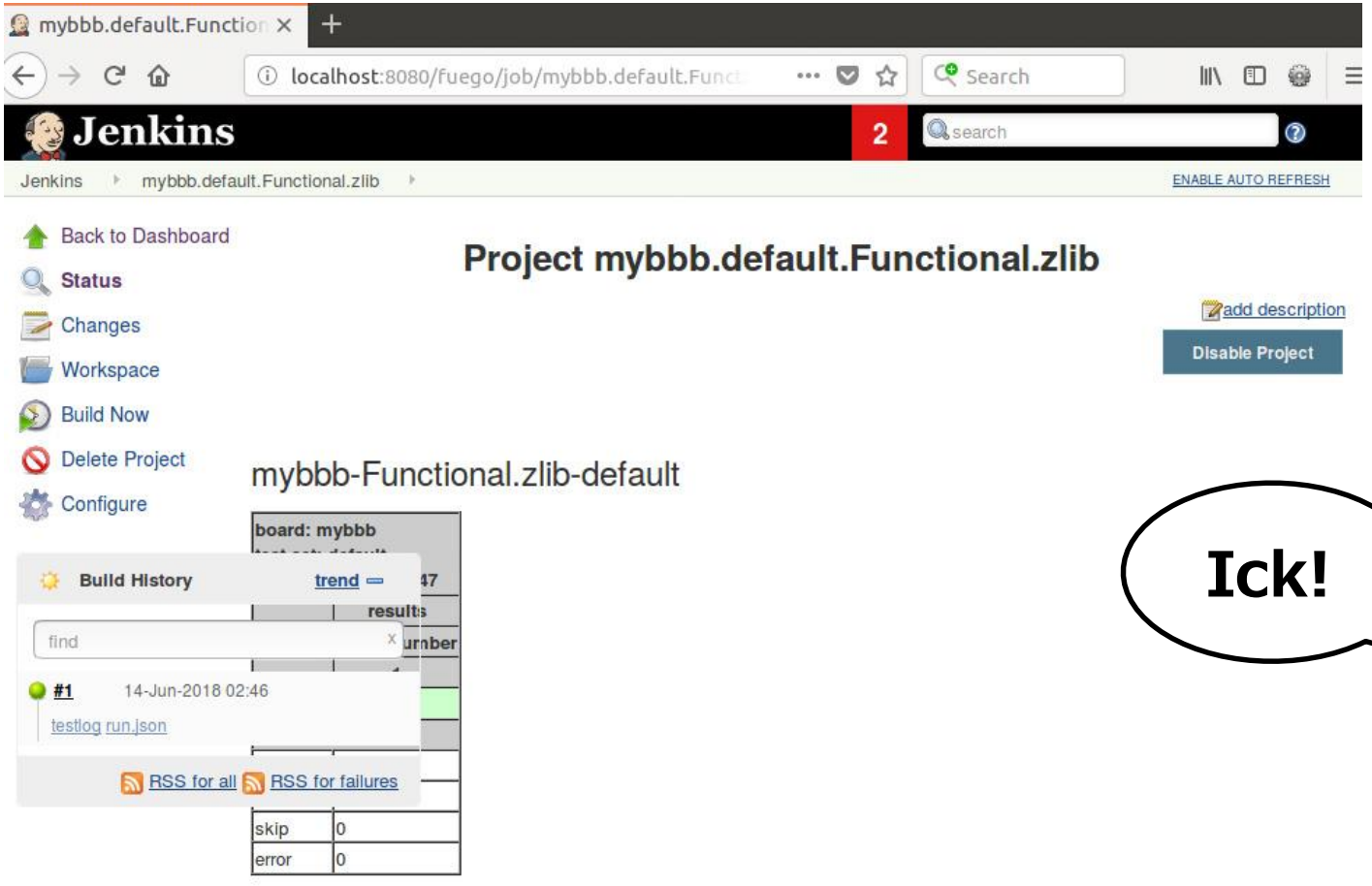
Calling Fuego from Jenkins

- **Main changes**
  - ftc run-test now supports all test flags (reboot, timeout, ..)
    - Jenkins is not needed to run tests.
  - Jenkins can call ftc run-test instead of main.sh
  - ftc run-test updates NextBuildNumber correctly
    - Unfortunately Jenkins does not display cli builds yet



Problems with Jenkins

- **Jenkins plugins are not reliable**
  - Each update causes many breaks
  - Our Flot plugin and HTML tables also have issues



Problems with Jenkins

- **What to do?**
  - Restrict Jenkins to job building with 0 plugins
    - Restricted visualization
      - PASS/FAIL circles
      - Complete log
    - Users can still use Fuego without Jenkins through the CLI
      - Create useful scripts that exploit the flexibility of the CLI
  - Move visualization to a web service
    - Usable by local developers and for sharing results on a centralized server.

Visualizing Fuego test results

- **What do we want to visualize**
  - Fuego test results: run.json
  - Fuego artifacts: logs, reports, etc
- **Command line interface**
  - Run IDs current format: test-spec-buildno-board
    - Problem: some specs contain hyphens (-)
  - Allow using --where clauses on put-run

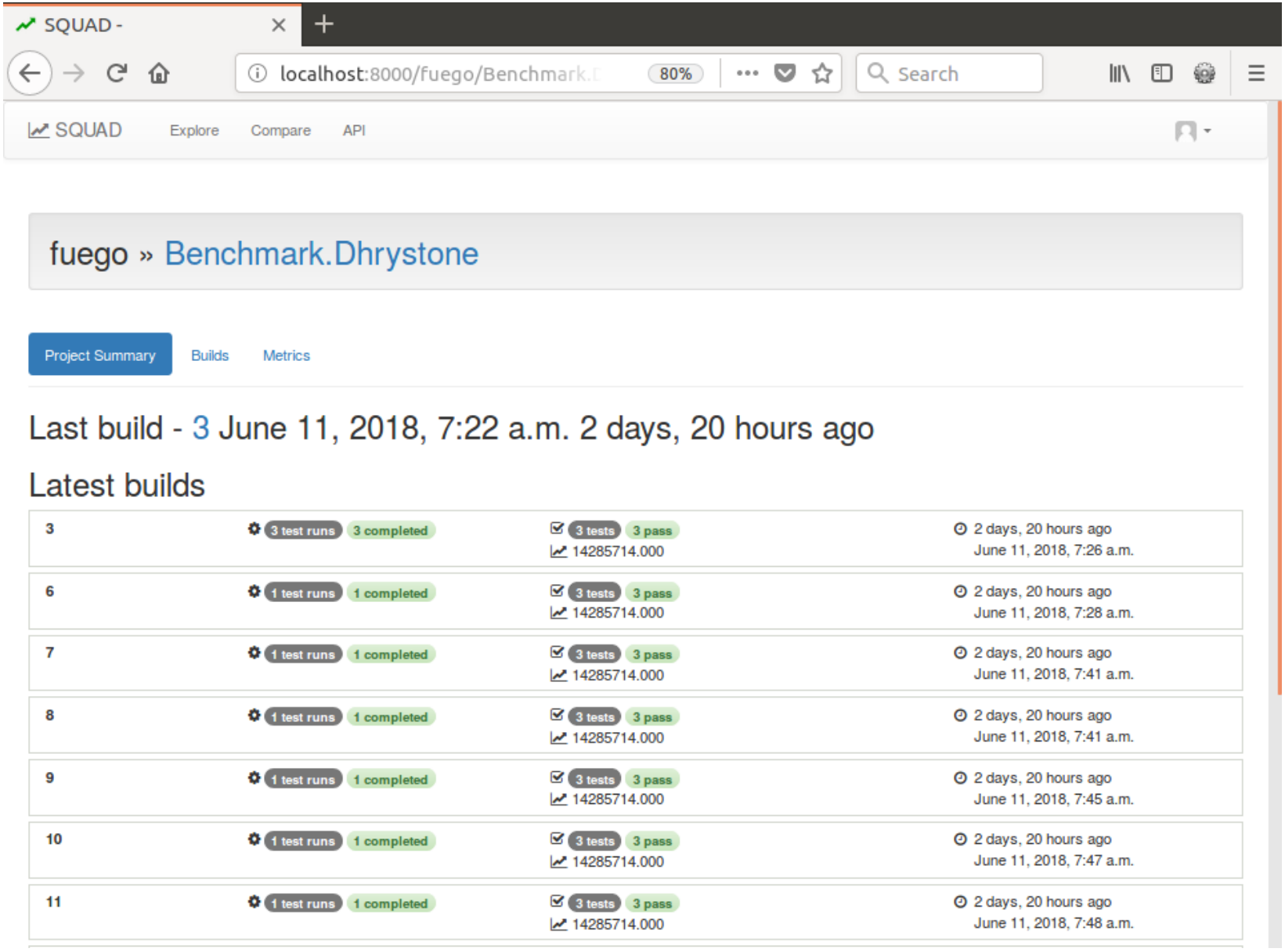
```
$ ftc list-runs -q -where test=Benchmark.Dhrystone
  Functional.bc-mult-1-bbb
  Functional.bc-add-1-raspi
  Functional.bc-sub-1-raspi
$ ftc put-run -r Functional.bc-mult-1-bbb --backend squad
$ ftc put-run --where board=raspi --backend kernelci
```

Visualizing Fuego test results: SQUAD

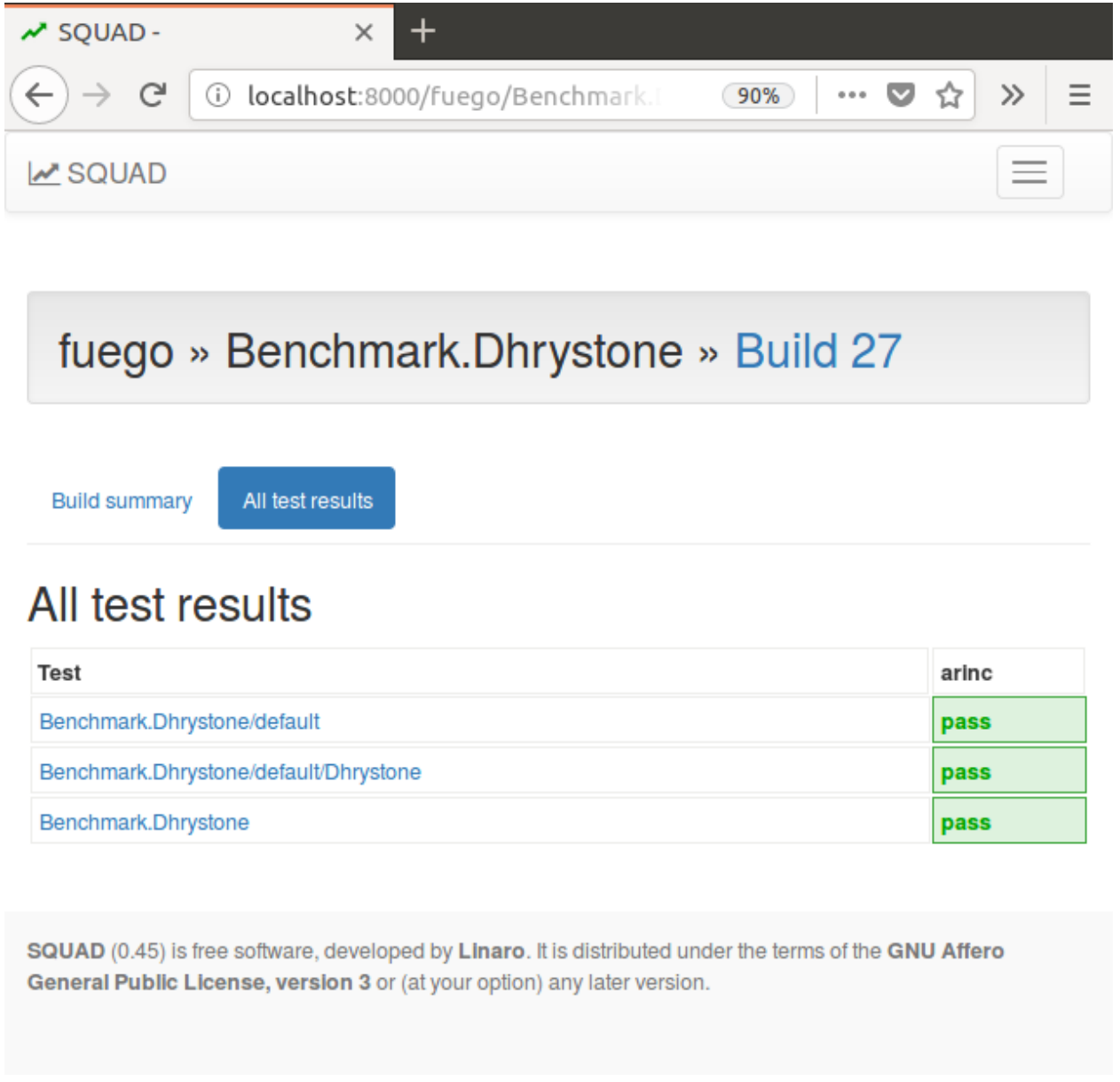
- **SQUAD**
  - Django-based test dashboard with POST/GET API
  - https://squad.readthedocs.io
  - https://qa-reports.linaro.org/
- **Easy to install locally**

```
$ sudo apt-get install rabbitmq-server
$ git clone https://github.com/Linaro/squad
$ cd squad
$ mkvirtualenv --python=python3  mysquad
$ pip3 install -r requirements-dev.txt
$ ./manage.py migrate
$ ./manage.py createsuperuser user user@mail.com password
$ ./manage.py runserver
$ firefox http://127.0.0.1:8000/
```

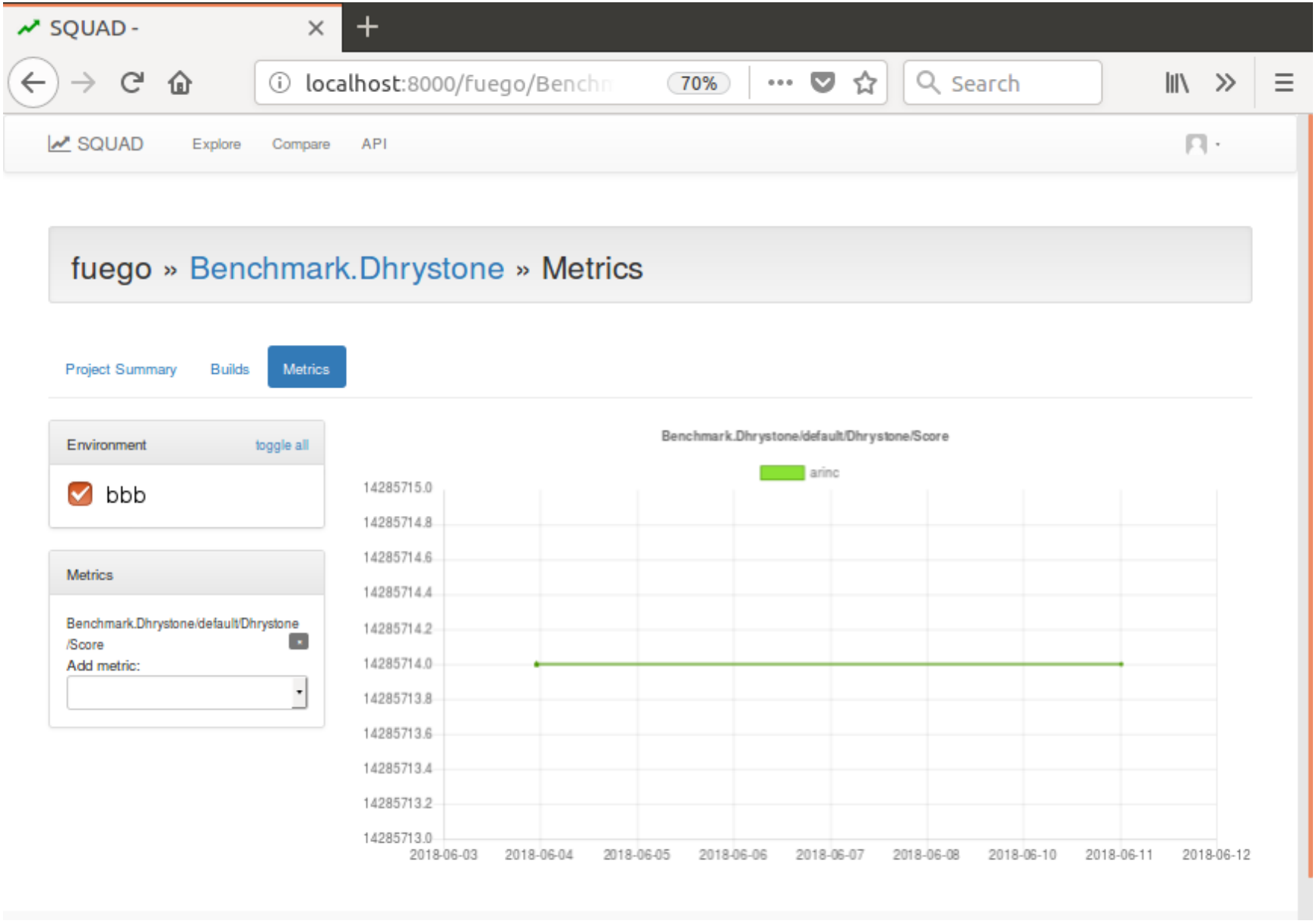
Visualizing Fuego test results: SQUAD



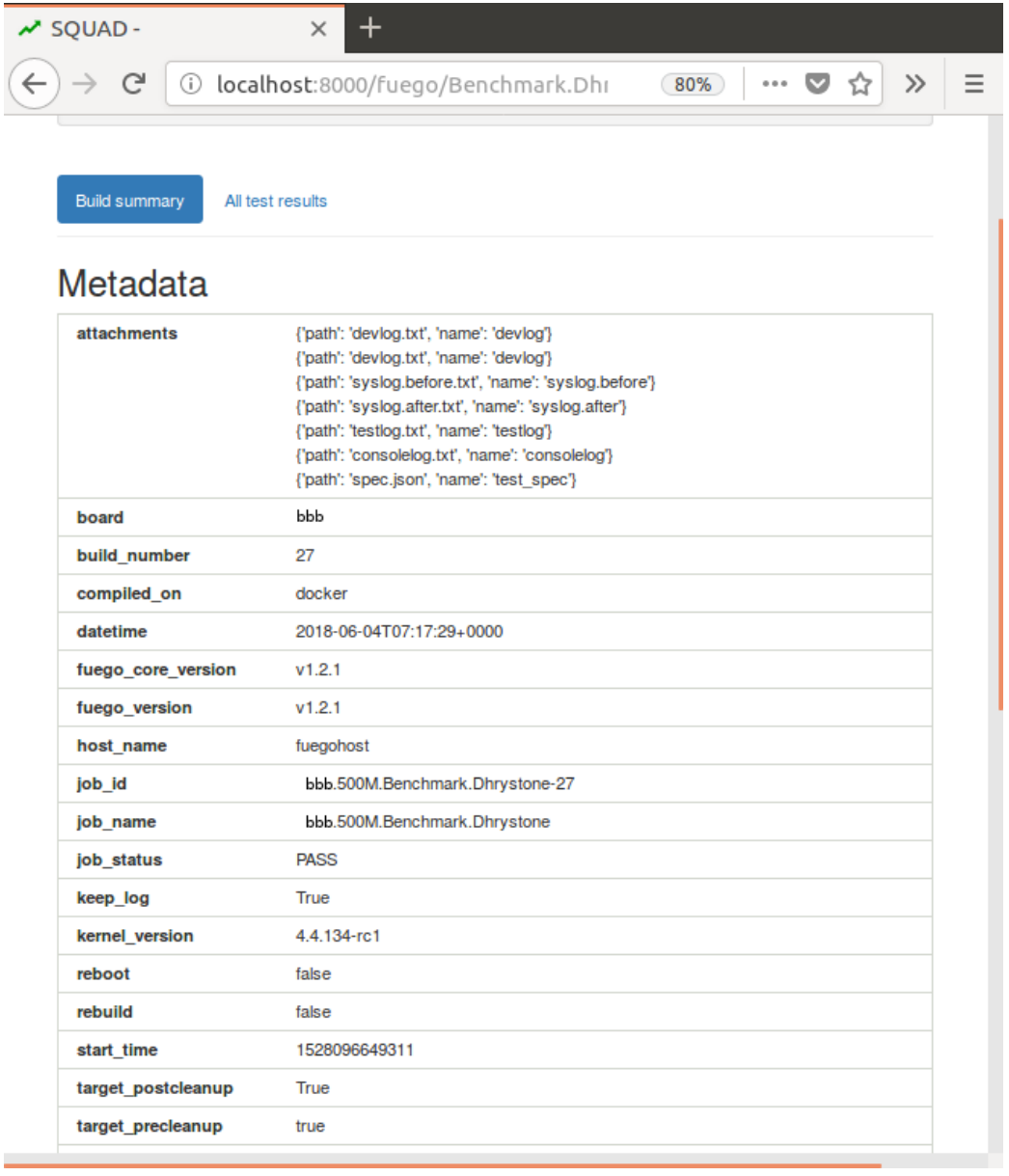
Visualizing Fuego test results: SQUAD



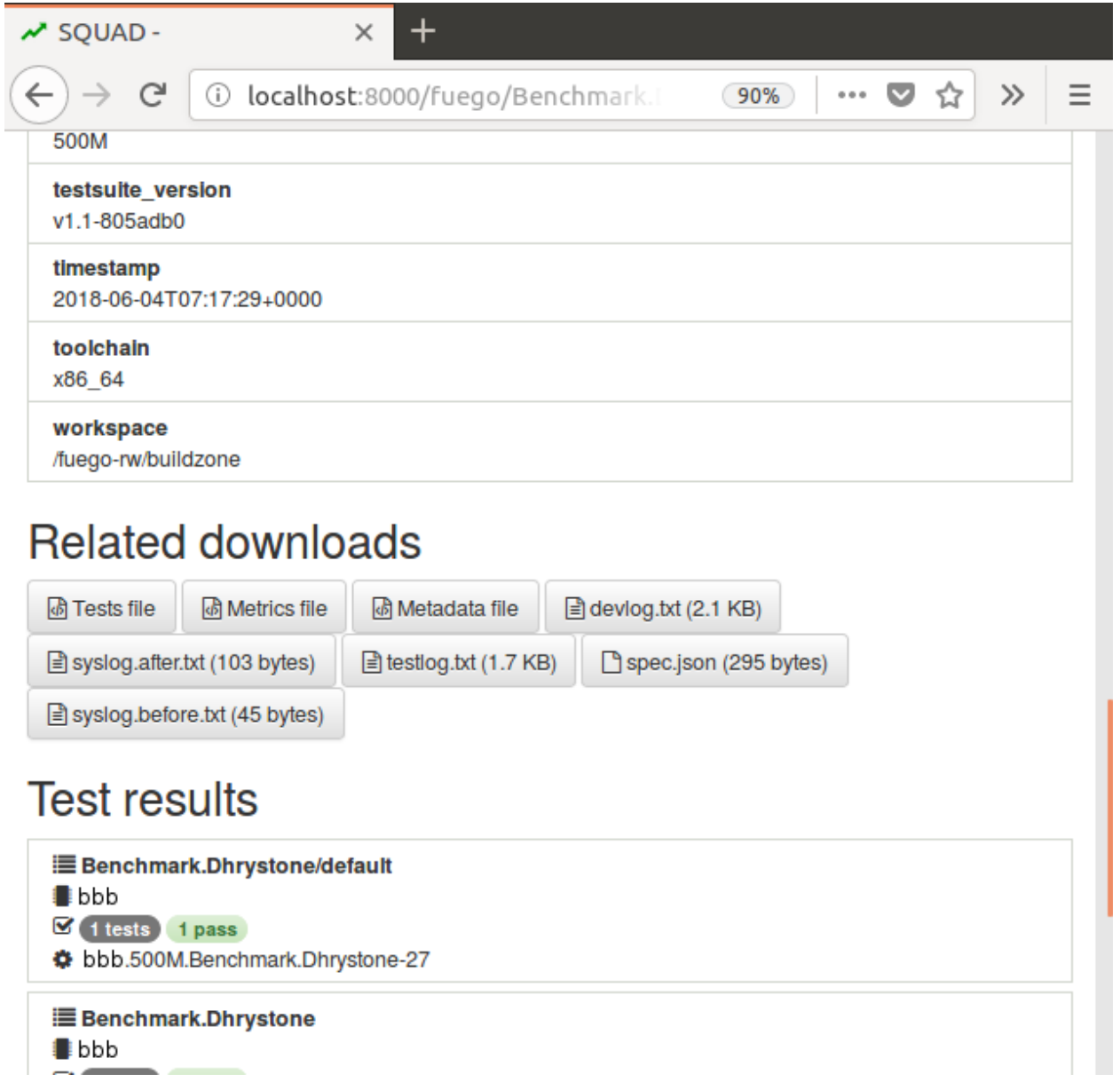
Visualizing Fuego test results: SQUAD



Visualizing Fuego test results: SQUAD



Visualizing Fuego test results: SQUAD



Visualizing Fuego test results: KernelCI

- **KernelCI**
  - Flask-based test dashboard/backend with POST/GET API
  - https://kernelci.org/
- **Became easier to install recently**

```
$ git clone https://github.com/lucj/kernelci-docker
$ cd kernelci-docker
$ git submodule init
$ git submodule update
$ git submodule foreach git checkout master
$ git submodule foreach git pull origin master
$ ./dev-start.sh
$ firefox http://127.0.0.1:8080/
```

Visualizing Fuego test results: KernelCI

- **In general Squad is very similar to KernelCI**
  - Personally I prefer Squad
- **A few problems I found in KernelCI**
  - Submitting a test requires a build_id
    - This build_id represents a kernel build test you need to do in advance.
    - That makes sense for their use case but not for us.
  - Their API does not seem to include status for test_sets or for the test_suite, only for test_cases.
  - Submitting POST requests is a bit more complicated than in Squad
  - Squad API is much simpler and flexible

FTC code improvements

- **Argparse**
  - Improves maintainability, code reuse and quality.
- **Configparser**
  - No need to reinvent the wheel.
- **Modularization**
  - Separate Jenkins related code on a different module
- **Deadcode**
  - Separate it on a different file or mark it as dead
- **Towards Python 3.x**
  - Python 2.x EOF: January 1, 2020
  - 2to3 tool might be useful

Trinity (fuzzing testing)

- **Prepared a new test wrapper for trinity**
  - It works well but..
- **Problem**
  - How to decide whether the test passed?
  - For example, check that the kernel did not panic?

Dynamic variables

- **Allow overriding test parameters**
  - Adds flexibility to the current spec.json based testing
- **Examples**

```
$ ftc run-test -b bbb -t  Functional.LTP_one_test ¥
  --dynamic-vars “{‘TEST’:’access01’, ‘scenario’:’syscalls’}”

$ ftc run-test -b bbb -t Benchmark.iperf3 ¥
  --dynamic-vars “{‘client_params’:’-u -t 10 -b 100M’}”
```

Useful scripts

- **Performance regressions**
  - Goal: check the evolution of a Benchmark's score along time
- **For any source code (e.g.: iperf)**
  - Check the evolution in performance of a test

```
$ ./fuego-test-evolution -b board ¥
  -t Benchmark.iperf3 -s udp ¥
  --commits asdf687234,asfdw913,234we5sf,sdf235d
```

- **For the Linux kernel**
  - Check the evolution in performance of the Linux kernel

```
$ ./fuego-kernel-evolution -b board ¥
  -t Benchmark.iperf3 -s udp --kernel-spec "lts-4.4.y" ¥
  --commits as2dg567,sa234tswr,496bf5782,234345
```

Useful scripts

- **Simple example in a shell script**
  - Check iperf3 performance for 3 stable kernels to compare
  - Dynamic variables become handy

```
BOARD=mypc
GITREPO=git://myserver/kernel/linux-stable.git
GITREFS=("linux-4.4.y", "linux-4.9.y“, “linux-4.14.y”)
CONFIG=/fuego-ro/boards/myconfig
DEPLOY_METHOD=scp
TEST_NAME=Benchmark.iperf3
TEST_SPEC=udp

for GITREF in "${GITREFS[@]}"; do
   ftc run-test --rebuild true -b $BOARD -t Functional.kernel_build ¥
      --dynamic-vars "{'gitrepo': '$GITREPO', 'gitref': '$GITREF', 'config':
          '$CONFIG', 'deploy_method': '$DEPLOY_METHOD'}“
   ftc run-test --reboot true -b $BOARD -t $TEST_NAME -s $TESTSPEC
done
```

Useful scripts

- **Functional bisect**
  - Goal: find commit that caused a test to fail
- **Example**
  - Find commit that caused a an LTP test to fail

```
$ ./fuego-kernel-evolution -b board ¥
  -t Functional.LTP_one_test ¥
  --dynamic-vars="{'TEST':'fnctl35'} ¥
  --kernel-spec "lts-4.4.y" ¥
  --commits as2dg56wr, HEAD
```

Ideas

- Add interactive mode

```
$ ftc run-test –i
What test do you want to run? LTP_one_test
Which board? Myboard
Myboard is not available, do you want to create one? y
How can I connect to your board (ssh)? help
   Please choose from ssh, serial, ttc, local.
How can I connect to your board? ssh
What is the IP address of your board? 192.168.1.55
User: root
Password: root
What spec do you want to use (default)?  Fcntl35
Do you want to reboot the board? Y
I couldn’t find a way to reboot it. Please reboot manually and
press enter when your board is ready [ENTER]
OK, you are ready to run a Fuego test. Press [ENTER] to start
```

Ideas

- **Add tab completion**

```
$ ftc run-test –b myboard –t Benchmark.<tab>
$ ftc run-test –b myboard –t Benchmark.Dhrystone –s <tab>
```

- **Reorganize fuego**
  - Move tarballs to a separate repo
    - Add them when creating the pre-built docker image
  - Move testplans to fuego-ro/testplans
  - Fuego-core should only have code and configuration
    - Add requirements.txt to install python dependencies on any OS (not just debian or docker).
  - Put test build dependencies into their yaml files
    - Dockerfile can remain as it is for convenience

Thanks for your attention

ご清聴ありがとうございました

Gracias por su atención