tmt

FREEDOM FOR TESTS
COMFORT FOR USERS

František Nečas • Miroslav Vadkerti • Petr Šplíchal

DevConf 2022

introduction

WHO

Who are we?

- Miroslav Vadkerti — Testing Farm Team
  - thrix / mvadkert@redhat.com
- František Nečas — Packit Team
  - FrNecas / fnecas@redhat.com
- Petr Šplíchal — Operating System CI
  - psss / psplicha@redhat.com

AGENDA

What to expect from this talk?

- Short introduction about tmt
- Hands on!
  - Feel free to try tmt with your components/packages
  - Ideally have your Fedora account ready
  - Please, ask questions any time
- What's new since the last devconf?
- Plans for the near and far future

See the sched event to download slides and the tmt cheat sheet

TEST MANAGEMENT TOOL

What is it and why was it created?

- User-friendly way to work with tests
- Human-readable concise configuration to run tests in various environments
- Configuration stored in **fmf** format (based on YAML), makes use of 3 levels of metadata:
  - L1 = tests (e.g. summary, require, duration)
  - L2 = plans (how the tests are to be executed)
  - L3 = stories
- Introductory guide, inspirative examples, metadata specification

FREEDOM FOR TESTS

- Tests do not need to rely on internal infrastructure
- No dependency on a particular test case management system (TCMS)
- Stored inside git repositories, convenient to make open source
- Easier cooperation between upstream and downstream
- Integration testing across components

COMFORT FOR USERS

- All test metadata stored inside a single git repository
  - Simple human-readable configuration based on YAML
- Consistent configuration across
  - Packit GitHub/GitLab, Fedora CI, CentOS Stream, RHEL CI
- Execute tests according to your preferences
  - Virtual machine? Container? A physical server? No problem
  - Flexible interface
- Easier cooperation between devel and QE on test code

first steps

INSTALL

Install tmt on your laptop. Try core/full package, selected provision plugins.

```
# basic features, executing tests on localhost
sudo dnf install -y tmt

# all available subpackages including all dependencies
sudo dnf install -y tmt-all


# experiment safely and easily inside a container
podman run -it --rm quay.io/testing-farm/tmt bash

# full container image with all packages (large)
podman run -it --rm quay.io/testing-farm/tmt-all bash
```

INITIALIZE

Initialize a metadata tree

```
mkdir tmt_workshop && cd tmt_workshop

# Initialize a tree with sample metadata
tmt init --template mini

# When in doubt, refer to help
tmt init --help
```

SMOKE

Let’s look at the created test and execute it in different environments

```
summary: Basic smoke test
execute:
    script: tmt --help
```

```
# Run it locally
tmt run --all provision --how local

# Run it in a VM (requires tmt-provision-virtual subpackage)
tmt run
tmt run --all provision --how virtual --image fedora-34
```

MULTIPLE TESTS

Execute multiple tests inside a single plan using discover step

```
# create a new test using the beakerlib shell framework
tmt test create —-template beakerlib tests/smoke


summary: A simple test
test: ./test.sh
framework: beakerlib
```

COMBINE

Make use of the created test inside a plan

```
# Look for tests in the local fmf tree
discover:
    how: fmf
execute:
    how: tmt
```

let’s get our hands dirty!

EXPLORE

Clone the git repository, look around

```
fedpkg clone fmf
or
git clone https://src.fedoraproject.org/rpms/fmf

cd fmf

tmt
tmt plans
tmt plans ls
tmt plans show
tmt plans show smoke
```

DISCOVER

Discover what tests would be run

```
tmt run discover

# select plan
tmt run plan --name upstream discover

# verbose
tmt run plan --name upstream discover -v

# debug
tmt run plan --name upstream discover -vd
tmt run plan --name upstream discover -vdd
tmt run plan --name upstream discover -vddd
```

RUN TESTS

Execute tests, choose your preferred environment

```
# dry run
tmt run --dry

# run it in a vm
tmt run plan --name smoke

# run in a container
tmt run --all plan -n smoke provision --how container
tmt run ... --image fedora:fresh

# run it on your laptop (if you feel safe)
tmt run --all plan -n smoke provision --how local
```

CREATE A SIMPLE PLAN

Clone the repo, initialize, adjust simple plan, try

```
# let's try on the did tool or your favourite package
fedpkg clone did
cd did
git checkout -b smoke-test

# use the minimal template, rename the plan if you like
tmt init -t mini
mv plans/example.fmf plans/smoke.fmf

# adjust as necessary & try out!
vim plans/smoke.fmf
tmt run plan --name smoke
```

PULL REQUEST

Fork the git repository, push changes into a new branch, create the pull request

```
Click [Fork] on https://src.fedoraproject.org/rpms/did/
Go to the fork page
Copy ssh address from the [Clone] button

git remote add fork ssh://yournick@pkgs.fedoraproject.org/...
git add .fmf plans
git commit -m "Add a simple smoke test"
git push fork -u smoke-test

Follow url printed in the terminal
Create the pull request
```

CREATE MORE TESTS

More complex tests deserve a separate metadata

```
git checkout -b plugin-test

tmt plan create --template base plans/plugins
vim plans/plugins.fmf

tmt test create --template beakerlib tests/bugzilla
vim tests/bugzilla/*

tmt run plan -n plugins
```

REPRODUCE

- Vision
  - Provide a easy way to reproduce problems locally in all environments
- Initial version
  - A code snippet you can copy paste into your localhost
  - Only available for Fedora CI and CentOS Stream CI
- What you get
  - Same test code, same context, environment variables and image
- What is missing
  - Without artifacts installation - https://github.com/psss/tmt/issues/1018
- Examples:
  - http://artifacts.dev.testing-farm.io/12dca44f-bc3f-44cc-8eea-de37f8165949/
  - http://artifacts.dev.testing-farm.io/476bb42d-6803-4721-ab07-545c09c7a053/
  - http://artifacts.dev.testing-farm.io/c8fcf7b8-8015-4448-a0c7-a2b8dfd8298b/

DEBUG TEST CODE

Fast way to re-execute modified test code

```
# run all steps until test execution
tmt run --until execute

# repeat test execution as needed
tmt run --last execute --force

# log in to the guest to adjust what's needed
tmt run --last login

# apply test code changes, execute again
tmt run --last discover --force execute --force
```

MIGRATE

Migrate an old Beaker/STI test to tmt

- A couple of examples documented to help with migration from STI
  - How do I migrate STI tests to tmt?
- Easy conversion of old Beaker Makefiles thanks to import:

```
# including tcms integration
tmt test import

# process just the Makefile
tmt test import --no-nitrate
```

MORE

A couple more useful commands

```
# check run status
tmt status

# clean runs, guests, images
tmt clean

# verify config against the specification
tmt lint

# track implementation, test and docs coverage
tmt stories
```

NEWS

What's new since last year?

- Zuul integration: CentOS Stream 9 / Fedora
- Features
  - Support for reboot during the test execution
  - Support for plan parametrization from environment
  - Integration tmt test export --fmf-id | workflow-tomorrow --fmf-id -
  - Discover tests from sources: dist-git-source
- Command line experience
  - Implement tmt clean, tmt lint
  - Exit after the first failure
  - Progress bar to execute --how tmt
- Docs improvements
  - Second chapter of the Guide: Under The Hood
  - Docs for migrating from Standard Test Interface

FUTURE

What's ahead of us?

- Next steps
  - Documentation cleanup, extend the guide
  - Multihost test support, hardware requirements specification
  - Transfer some tasks from Testing Farm to tmt, consistent guest setup
  - Improve test debugging and usability (aliases, wizard mode)
  - Implement many nice ideas (custom test templates)
  - And, of course, fix a lot of bugs :-)
- How to get involved?
  - Submit bugs and ideas for improvement
  - Pick a good first issue and create pull request
  - Join the #tmt channel on IRC to discuss the design

LINKS

Here’s a bunch of useful links

- tmt docs
  - tmt.readthedocs.io
  - tmt cheat sheet
  - the guide
- fmf docs
  - fmf.readthedocs.io
- fedora quick start guide
  - docs.fedoraproject.org/en-US/ci/tmt
- packit & testing farm
  - packit.dev/testing-farm

questions?

thank you!