Improving ROOT Development Processes

Martin Storø Nyfløtt
Norwegian University of Science and Technology (NTNU Gjøvik)
Supervisor: Pere Mato
07.08.2017

Outline

1. Completely moving ROOT to GitHub
2. Pull requests, @phsft-bot and Jenkins Pipelines
3. Monitoring infrastructure with Grafana and Graphite
4. Communication on Mattermost
5. Daily Scrum Meetings
6. Shipping ROOT in Docker containers
7. Vision for the future

Completely moving ROOT to GitHub

- Early 2017: GitHub serves as a mirror of https://root.cern.ch/git/root.git
- Inconvenient to test and merge PRs
- https://github.com/root-mirror -> https://github.com/root-project
- Modernizing GitHub presence:
  - Updating readme.md: Screenshots, links, badges, build instructions
  - Better community interaction
  - “Modern” open source development

Pull requests, @phsft-bot and Jenkins Pipelines

- Pull requests are picked up by a Jenkins web-hook
- Security: Only whitelisted users get automatic building
  - Prevent unauthorized usage of our infrastructure
- Customization of platforms, compilers, and CMake flags through @phsft-bot
- Warnings, errors, and failing tests submitted as comments on PRs
- Implemented using Jenkins Pipelines
  - https://github.com/root-project/jenkins-pipelines

Disable 'from ROOT import *' in python>=3.6 (ROOT-8931) #825

Open **daritter** wants to merge 1 commit into `root-project:master` from `daritter:disable_importstar_py36`

Conversation 8 | Commits 1 | Files changed 1

daritter commented 6 days ago — Contributor

`from ROOT import *` leads to a segmentation violation when used with Python 3.6.

One could try fixing it instead of disabling it but the code which is used to install the lookup handler in the module `RootModule::SetRootLazyLookup` depends on internal CPython implementation details of the dict class which are not part of the public API. As a consequence keeping this alive will lead to very fragile code, require continuous effort to adapt to internal changes and cause a lot of #ifdef handling. (as a matter of fact, Python 3.7 would probably already require new changes to this code already).

Also there's a statement in the bug report that `from ROOT import *` is broken in Python 3 so I don't understand why it's not disabled as it will only confuse users: https://sft.its.cern.ch/jira/browse/ROOT-8931

As such I would propose to instead have a clear error message that `from ROOT import *` does not work. This pr adds an `ImportError` which is easy to handle but cannot be just ignored by the user.

phsft-bot commented 6 days ago — Member

Can one of the admins verify this patch?

etejedor was assigned by **dpiparo** 5 days ago

Teemperor commented 4 days ago — Member

@phsft-bot build

phsft-bot commented 4 days ago — Member

Starting build on `centos7 / gcc49`, `mac1012 / native`, `slc6 / gcc49`, `slc6 / gcc62`, `ubuntu14 / native` with flags `-Dvc=OFF -Dimt=ON -Dccache=ON`

How to customize builds

phsft-bot commented 4 days ago — Member

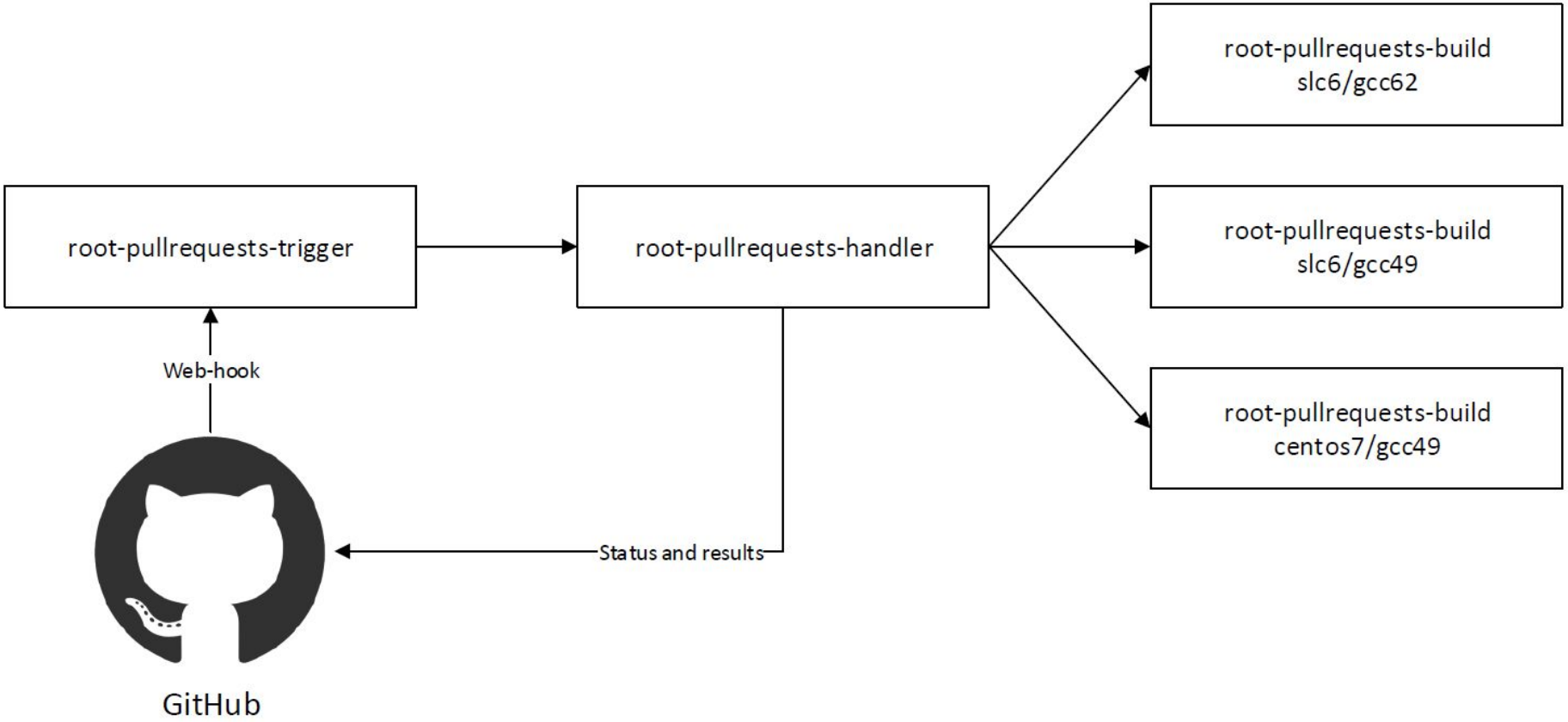
Build failed on mac1012/native.
See console output.

phsft-bot commented 4 days ago — Member

Build failed on slc6/gcc49.
See console output.

Failing tests:

- projectroot.test.test_TFormulaTests
- projectroot.roottest.root.dataframe.roottest_root_dataframe_test_snapshotNFiles


root-pullrequests-build
slc6/gcc62
root-pullrequests-trigger
root-pullrequests-handler
root-pullrequests-build
slc6/gcc49
root-pullrequests-build
centos7/gcc49
Web-hook
Status and results
GitHub

Pipeline root-pullrequests-handler

Recent Changes

Stage View

	Build - centos7-gcc49-Debug	Build - mac1012-native-Debug	Build - slc6-gcc49-Debug	Build - slc6-gcc62-Debug	Build - ubuntu14-native-Debug	Publish reports
Average stage times:	50min 2s	19min 2s	44min 54s	39min 27s	48min 11s	876ms
#806 PR #838 Aug 05 22:28 No Changes	45min 22s	6min 34s failed	1h 0min	48min 28s	39min 29s	840ms
#805 PR #793 Aug 05 20:39 No Changes	47min 17s	7min 54s failed	48min 42s	37min 13s	1h 9min	868ms
#804 PR #836 Aug 05 18:24 No Changes	40min 7s	11min 30s failed	47min 47s	44min 5s	35min 58s	989ms
#803 PR #837 Aug 05 17:34 No Changes	1h 2min	1h 7min	42min 57s	48min 1s	43min 36s	996ms

Pipeline root-pullrequests-build

Recent Changes

Stage View

	Checkout	Build	Test	Generate reports
Average stage times:	3min 32s	12min 29s	33min 17s	224ms
#3933 ubuntu14/native PR #838 Aug 05 22:28 No Changes	7s	5min 22s	33min 47s	231ms
#3932 slc6/gcc62 PR #838 Aug 05 22:28 No Changes	5min 47s	14min 7s	28min 23s	219ms
#3931 slc6/gcc49 PR #838 Aug 05 22:28 No Changes	4min 42s	17min 59s	37min 42s	224ms

Jenkins Pipelines

- New job type available after Jenkins 2
- Jobs are expressed through a script instead of a form
- Groovy: Simplified Java, without semicolons
- Benefits: Flexibility, version control of pipeline script, persistent between Jenkins restarts
- Limitations: No matrix support, integration with free-style job plugins may vary

Pipeline example

```
node('centos7') {
    stage('Checkout') {
        git 'https://github.com/someone/something.git'
    }
    stage('Build') {
        sh 'build.sh'
    }
    stage('Test') {
        sh 'test.sh'
    }
}
```

Useful resources when creating pipelines

- Pipeline Syntax-link under Pipeline script step:
  - Pipeline script generator
  - Global Variable Reference
- https://jenkins.io/doc/book/pipeline/
- Pipeline steps reference: https://jenkins.io/doc/pipeline/steps/
- Shared libraries: https://jenkins.io/doc/book/pipeline/shared-libraries/
- Jenkins Javadoc: http://javadoc.jenkins.io/

Monitoring infrastructure with Grafana and Graphite

- Report time usage to Grafana
- Implemented in incremental builds, nightlies, and PR builds
- Python script for GitHub statistics, Groovy for job-statistics
- Useful for discovering trends, and project/infra statistics


ROOT
Zoom Out
Last 7 days
Pullrequests
Last 14 days
55
50
45
40
7/26
7/29
8/1
8/4
Open PRs
45
Closed PRs
774
Total time (build + queue)
1 day
1 day
20 hours
13 hours
6 hours
0 seconds
7/31 00:00
8/1 00:00
8/2 00:00
8/3 00:00
8/4 00:00
8/5 00:00
8/6 00:00
Pullrequests
Incrementals
Nightlies
Build time
16 hours
13 hours
11 hours
8 hours
5 hours
2 hours
0 seconds
7/31 00:00
8/1 00:00
8/2 00:00
8/3 00:00
8/4 00:00
8/5 00:00
8/6 00:00
Pullrequests
Incrementals
Nightlies

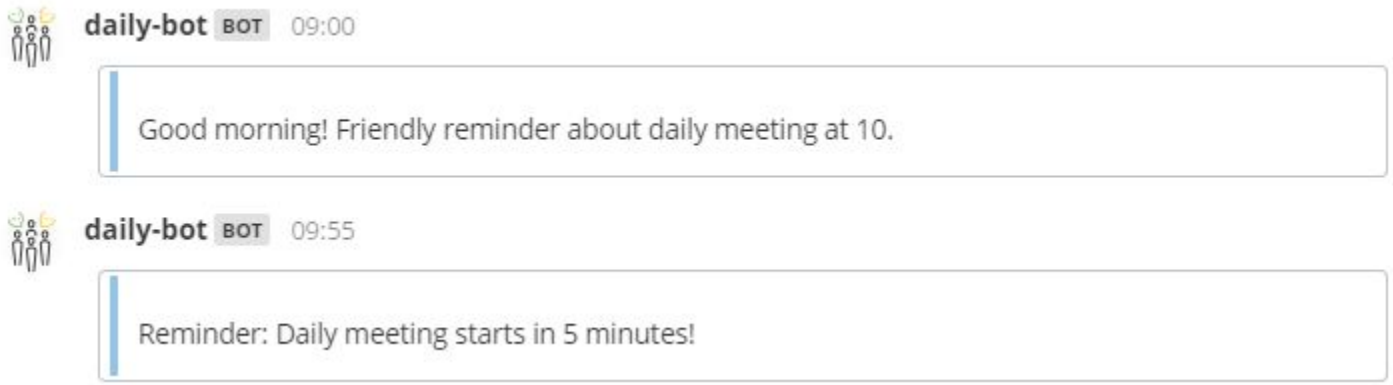
Communication on Mattermost

- Instant messaging service provided by CERN
- Open source alternative to Slack
- Improve team-communication
- Chat channels: Topic-oriented
- Bots and service integration: Jenkins, reminder for daily scrum meetings

https://mattermost.web.cern.ch

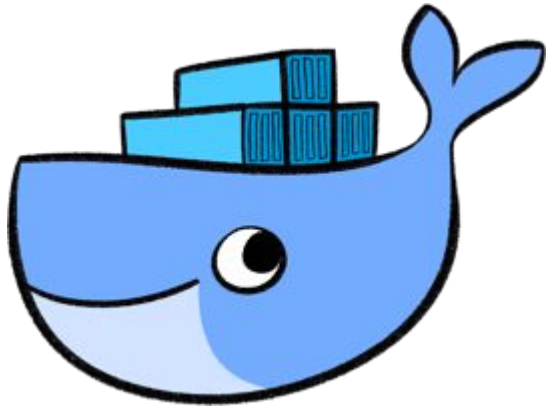
Daily Scrum Meetings

- Roundtable each monday: Not enough
- Meetup 10:00 each morning
- What you did yesterday, what are you going today?
- Time-boxing: 90 seconds, meeting ~10 minutes long
- Get better idea of what others are working on
- Get help from peers on known issues



Shipping ROOT in Docker containers

- Users can easily pull the latest version of ROOT
- No compiling needed
- Built on Ubuntu 16.04 base-image (root-ubuntu16-base)
- Continuous delivery:
  - Each passing incremental build => new snapshot container
  - Each passing 6.10-patches nightly => new 6.10 container
- What does this mean?
  - Low boundary to try out ROOT
  - Easier distribution
  - Larger userbase: Any platform where Docker runs is supported
  - Users can test out new features as they are developed
- https://root.cern/downloading-root



Demo:

Running ROOT on Windows 10 with Docker

Vision of the future

- More stable build infra
  - Disk space issues => Dockerization, push finished builds to Docker registry
  - Random crashes/exceptions =>
    - More conservative regarding plugins, follow up issues on issue trackers
  - Git checkout timeout => Retry count
- Every commit arrives as a pull request
  - Ensure this code is understood: Code review
  - Does it compile?
  - Does it break any tests?
  - Higher quality code, less test breakage, more stability

Vision of the future cont.

- Protected master branch: (https://help.github.com/articles/about-protected-branches/)
  - Can’t be force pushed, deleted, or edited through the web
  - Commits needs to pass status checks
  - Commits must pass review
  - Avoid by design that someone pushes a commit that breaks master, or a change that nobody else knows about or understands
- Person on shift should revert commits that breaks master
- More focus on process: Retrospective

Questions/comments

Big thanks to:

Pere Mato, Danilo Piparo, Vassil Vassilev, Javier Cervantes Villanueva,

and everyone else on the ROOT team!