

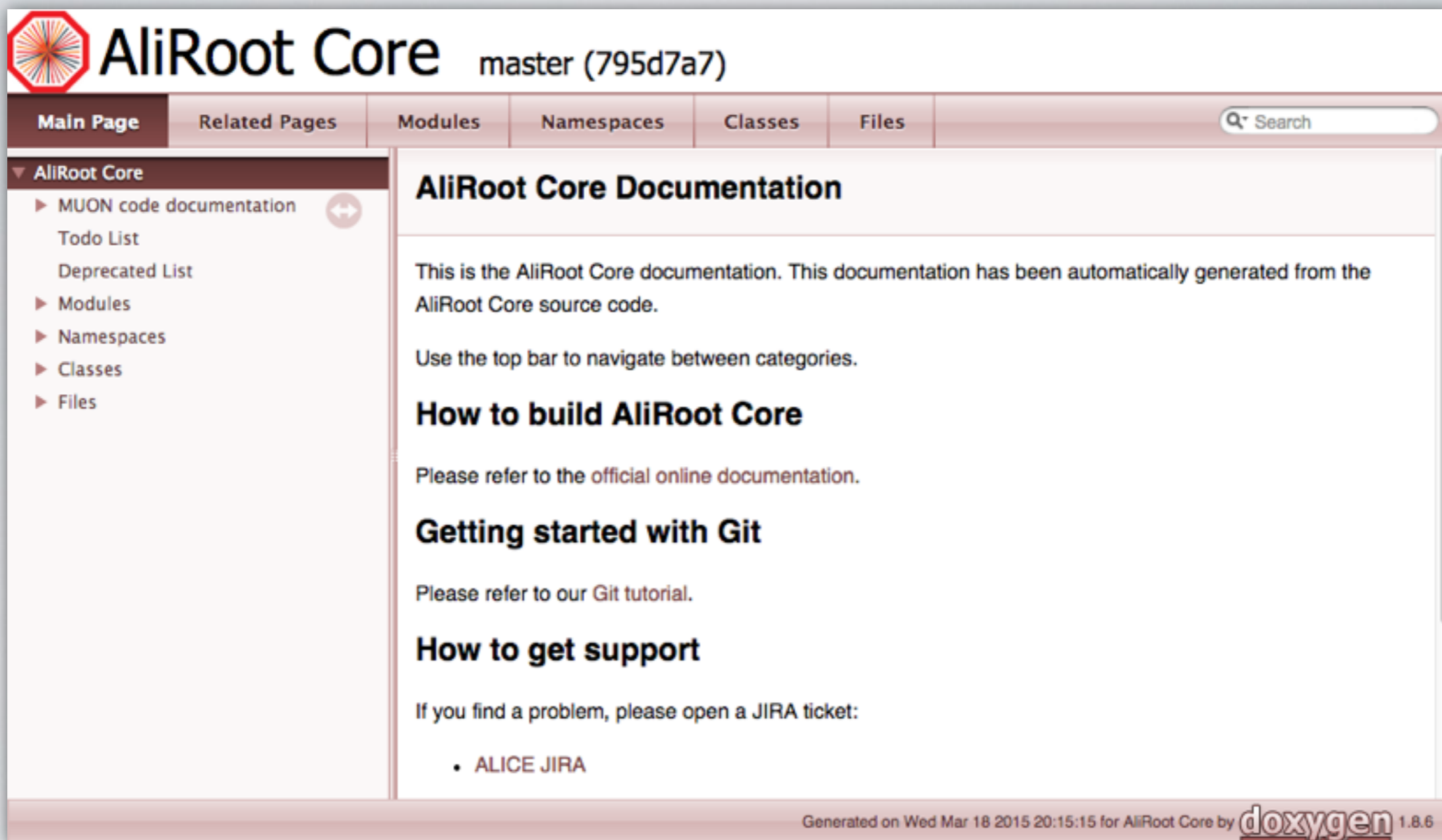
Documentation status, automatic build tests with CDash and daily tagging strategies

Dario Berzano
ALICE Offline - CERN

ALICE Offline Week - 19.03.2015

Documentation

- AliRoot code had **diverse** documentation formats:
 - Official: **THtml**
 - Some private parts: **Doxygen** was already used
 - Most code documented with non-specially formatted comments
- Why Doxygen
 - **No ROOT required** to generate doc
 - Has a **cache**: continuous **incremental** generation is fast
 - More **robust**



The screenshot shows the AliRoot Core documentation page. At the top, there is a navigation bar with tabs for 'Main Page', 'Related Pages', 'Modules', 'Namespaces', 'Classes', and 'Files'. A search box is located on the right side of the navigation bar. Below the navigation bar, there is a sidebar on the left with a tree view of the documentation structure. The main content area displays the 'AliRoot Core Documentation' page, which includes a welcome message, instructions on how to build AliRoot Core, and how to get support. The footer of the page indicates it was generated on Wed Mar 18 2015 20:15:15 for AliRoot Core by doxygen 1.8.6.

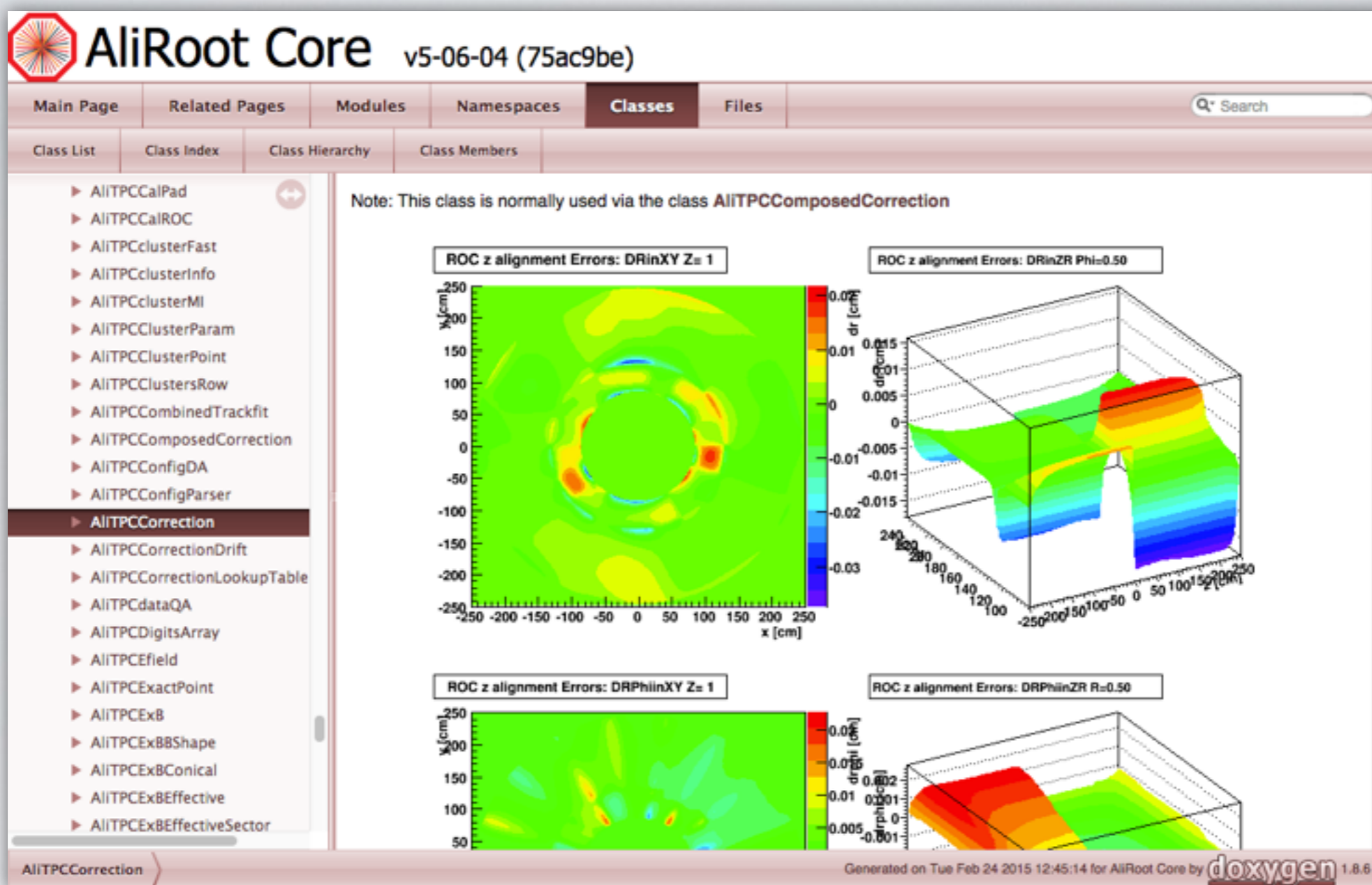
- Generated and archived for every tag
- Every two hours: generated from master

- Documentation conventions are available here:
 - <https://dberzano.github.io/alice/doxygen>
- What is covered:
 - [Class and macro](#) general description
 - [Function](#) comments
 - Commenting class [data members](#)
- Markup language used: [Markdown](#) - much easier than HTML
- If you write a new class, please document it from the beginning

- We thought we could run a script and convert it in one go
- ...but AliRoot did not follow clear conventions: **automation impossible**
- ~~Solution: an artificial intelligence doing the work for us~~
- Other solution: a script doing most of the work for you
 - We wrote the **script** and its doc
 - It does **most of the work**
 - You **check** its output and make minor adjustments
 - When happy you commit

- Python script *helping** with the conversion
- Based on [libclang](#)'s Python bindings
- *Must be used [carefully](#)...
 - Lots of dirty coding
 - Always [check the diff](#) before committing!
 - Very sensitive to libclang's version: Python API [changes frequently](#)
- ...but at the end of the day it saves from lots of manual editing
- How to use it: https://dberzano.github.io/alice/doxygen/#convert_existing_documentation_to_doxygen
- Helping ROOT converting their doc to Doxygen

- THtml supports on the fly image generation via macros
 - Macro block: `BEGIN_MACRO ... END_MACRO`
 - Internally, macro is run and the pad is printed to an image file
- In ALICE we want to `generate doc without having ROOT`
 - Macro extracted to .C file and substituted with Markdown tag:
`![Picture from ROOT macro](TBlah_h_foobar.png)`
 - Both `image` and `source` macro committed
 - Manually exec a helper script to regenerate image



- Images are PNG files committed under `<yourdir>/imgdoc`
- Utility to (re)generate images from ROOT macros

the binsize of the generated histogram, -1 means, that the maximal reasonable stepsize is used The actual work is done on the array.

$$f(x, \mu, \sigma) \Rightarrow S(t, \mu, \sigma) = \int_{-\infty}^{\mu+t\sigma} f(x, \mu, \sigma) dx / \int_{-\infty}^{+\infty} f(x, \mu, \sigma) dx$$

- Both THtml and Doxygen support LaTeX:
 - ROOT: `BEGIN_LATEX ... END_LATEX`
 - Doxygen: `\f[... \f]`
- Symbols like `#sigma` are replaced with the native `\sigma`
- Doxygen supports `MathJax`: no need to pre-render formulas!

- In order to avoid confusion, only the directories converted to Doxygen are considered when generating documentation
- When you are done, you must **add the directory explicitly**
 - This is explained here too: https://dberzano.github.io/alice/doxygen/#adding_your_directories_and_images_to_doxygen
 - Probably you don't have the permissions to do that: open a JIRA ticket asking to add the directory to Doxygen

Check result before pushing

- Generating documentation is easy:

make doxygen

- Generates HTML files locally
- You can easily test it before pushing
- Only [Graphviz](#) and [Doxygen](#) required
- This is explained here: https://dberzano.github.io/alice/doxygen/#run_doxygen

- Inline data member comments in Doxygen can have two formats:

```
int a;    ///< Description of a  
int b;    ///< Description of b
```

- This used to be [compatible with ROOT 5's special comments](#):

```
int a;    ///< ROOT non-transient and valid Doxygen  
int b;    ///< ROOT transient and valid Doxygen
```

- ROOT 6 broke this compatibility (///
is OK, but ///
is not):
 - `int a; ///
ROOT non-transient` and valid Doxygen
 - `int b; ///
Valid Doxygen, but non-transient!`
 - `int c; ///
Ignored by Doxygen, ROOT transient`
- ROOT 6 interprets ///
as a Doxygen-only marker and does not mark the variable as transient any longer!
- Issue opened to ROOT: <https://sft.its.cern.ch/jira/browse/ROOT-7125>
- Workaround: transient member comment in Doxygen, ROOT 5 and 6:
 - `int a; ///
Valid Doxygen, transient in both ROOT 5 and 6`
- Notice the double exclamation mark!

Who writes the doc?

- **AliRoot**
 - **TPC**: we have partly done it as an example
 - Lots of images and formulas: it was a good testing
 - **MUON**: it was already in Doxygen format (thanks Ivana!)
 - **STEER, ANALYSIS,...**: to be done by the **Offline** (*will do ASAP*)
- **AliPhysics**
 - PWG members should do that
 - In general every user is responsible of her own code

Who writes the doc?



I WANT YOU

- We have a sensible documentation format and conventions
- [Automatic generation](#) and instructions are in place
- [You](#) need to [convert](#) and/or [write](#) the doc

CTest and CDash

[Login](#) [All Dashboards](#) Wednesday, March 18 2015 22:23:03 UTC

AliRoot

[Dashboard](#) [Calendar](#) [Previous](#) [Current](#) [Next](#) [Project](#)

No file changed as of **Tuesday, March 17 2015 - 01:00 UTC**

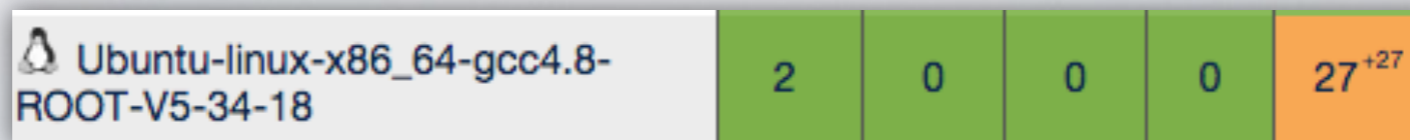
3 hours ago: 2 files changed by ppilot
 7 hours ago: 1 file changed by morsch
 13 hours ago: 1 file changed by shahoian
 1 days ago: 1 file changed by rgrosso
 1 days ago: 1 file changed by rgrosso

[See full feed](#)

Continuous

Site	Build Name	Update	Configure		Build		Test			Build Time
		Files	Error	Warn	Error	Warn	Not Run	Fail	Pass	
alinsure.cern.ch	Ubuntu-linux-x86_64-gcc4.8-ROOT-V5-34-18	1	0	0	0	0	0	0	0	Mar 17, 2015 - 17:49 UTC
alinsure.cern.ch	Ubuntu-linux-x86_64-gcc4.8-ROOT-V5-34-18	1	0	0	0	0	0	0	0	Mar 17, 2015 - 15:42 UTC
alinsure.cern.ch	Ubuntu-linux-x86_64-gcc4.8-ROOT-V5-34-18	1	0	0	0	0 ⁻²⁷	0	0	0	Mar 17, 2015 - 15:28 UTC
alinsure.cern.ch	Ubuntu-linux-x86_64-gcc4.8-ROOT-V5-34-18	2	0	0	0	27 ⁺²⁷	0	0	0	Mar 17, 2015 - 14:28 UTC
alinsure.cern.ch	Ubuntu-linux-x86_64-gcc4.8-ROOT-V5-34-18	12	0	0	0	0 ⁻²	0	0	0	Mar 17, 2015 - 14:14 UTC
alinsure.cern.ch	Ubuntu-linux-x86_64-gcc4.8-ROOT-V5-34-18	2	0	0	0	2 ⁺²	0	0	0	Mar 17, 2015 - 13:56 UTC
alinsure.cern.ch	Ubuntu-linux-x86_64-gcc4.8-ROOT-V5-34-18	2	0	0	0	0	0	0	0	Mar 17, 2015 - 13:35 UTC
alinsure.cern.ch	Ubuntu-linux-x86_64-gcc4.8-ROOT-V5-34-18	1	0	0	0	0	0	0	0	Mar 17, 2015 - 12:56 UTC
alinsure.cern.ch	Ubuntu-linux-x86_64-gcc4.8-ROOT-V5-34-18	1	0	0	0	0	0	0	0	Mar 17, 2015 - 11:07 UTC

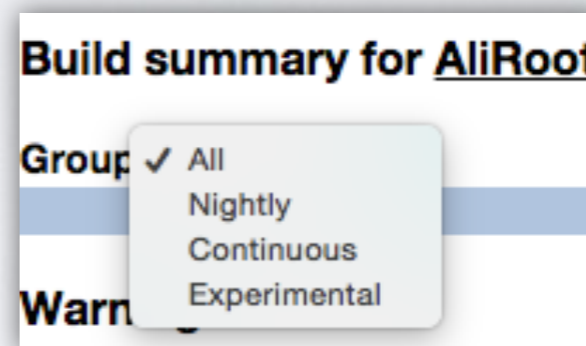
- [CTest and CDash](#): test your build and display results on a dashboard
- Dashboard: cdash.cern.ch



- Display [status of builds](#) on a shared dashboard
 - How many more or less [warnings/errors](#) with respect to last build
 - Build status on different platforms [at a glance](#)
 - Email [notifications](#) on new warnings/errors
- Also display results from [custom unit and functional tests](#)
- [Independent from the build system](#):
 - CDash is only used to *display* results from CTest
 - CTest can be [manually launched, or automated](#)

What we are currently testing

- We are testing **AliRoot Core** only on Ubuntu 14.04 LTS
 - **Continuous** builds (incremental “make” after each single commit)
 - **Nightly** builds (*from scratch*)
 - Those are the “certified” builds
- In addition to Continuous and Nightly, there’s an **Experimental** section
 - **Everybody** can launch a test: results [published on cdash.cern.ch](http://cdash.cern.ch)
 - Users/sites with special builds/needs can monitor and share results on the same dashboard
- *Thanks to **Mohammed** for putting it in place!*



What we plan to have

- [AliRoot](#) and [AliPhysics](#)
- Automatic Continuous and Nightly builds on several platforms
 - Ubuntu, OS X, SLC/CC, Fedora
- Automatic [email notifications](#) to the committer that broke a build
 - They immediately know what did not work and on what platform
- Add basic [functional tests](#)
- Display the results from builds coming from our next build system
 - *See [Giulio's presentation on Jenkins](#)*

You can test it now

- Works on AliRoot Core
- Create a config file containing the lines:

```
export LINUX_FLAVOUR=ubu1404 # arbitrary
export ROOTSYS=path_to_root
export BUILDDIR=tmp_dir_where_to_build_aliroot
export SOURCEDIR=aliroot_source_dir
```

- Launch it - from the AliRoot Core source:

```
./Dart.sh <path_to_config_file>
```

- Results published at the end on cdash.cern.ch under Experimental

Daily tagging strategies for AliPhysics (*discussion*)

- AliPhysics has **daily tags**
 - At 4pm every day a script builds and tags
 - **Trains** can be started with the new tags
- This is a **porcelain** system
 - Users rushing to catch the train may **break all** at the last minute
 - We are skipping daily tags if *one* user broke all: **unfair for the others**
- **Recover** from broken builds
 - Lots of daily **tedious manual work** to revert commits before 4pm
 - At least thanks to the split, problems are **confined to AliPhysics**

Lots of room for automating

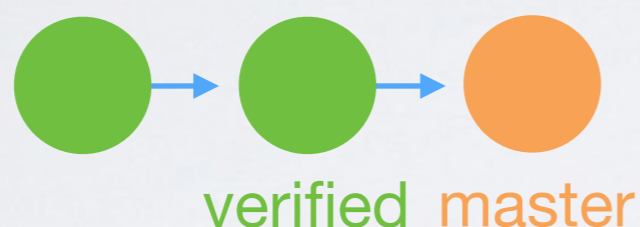
- Rationale:
 - Do not skip daily tags
 - Do not involve manual retagging
- Two major proposals plus one (collected during weekly meetings):
 - Some form of automatic revert of broken commits
 - Do not tag at HEAD but at the closest compiling commit
 - Use pull requests and label them accordingly (*with GitHub/GitLab*)

Automatic revert

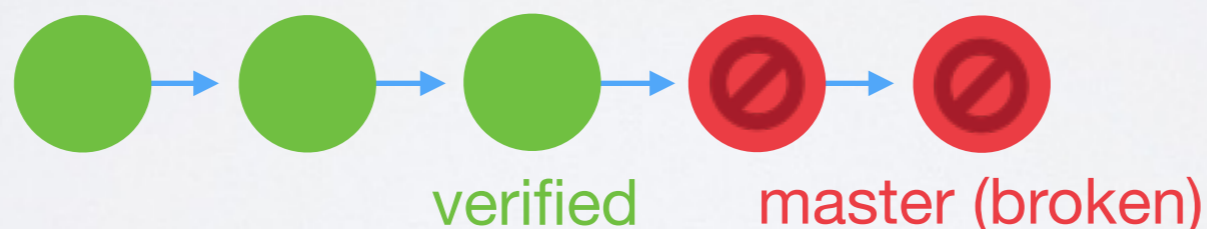
- Order-0 idea: if a commit is broken, `git revert <commit>`
- Problem: lots of commits in AliPhysics
 - Many of them don't compile: we would **litter the repository** with plenty of "revert commits"
 - It is not always possible to automatically revert a commit: there might be **merge conflicts** due to other commits!
 - Even if the revert applies, we might accidentally apply a revert *after* an appropriate fix has been committed already
- IMHO automatic revert is **complicated** and **risky** to implement

Verified branch

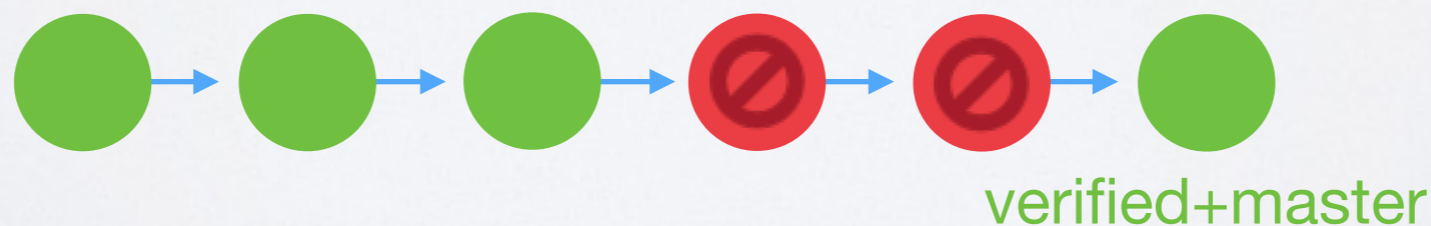
- Separate branch where no one can push: call it **verified**
- Every commit from **master** is ported to verified only if it compiles
 - No cherry-pick: **verified** and **master** share history (**verified** behind)



- If a commit breaks, stop porting new commits to **verified**



- If a fix commit appears, port it along with all the skipped ones:



- Advantages:
 - We tag from verified: no skipping tags, no manual work
- Objections collected so far:
 - *"Users will not know if their code made it to the daily tag"* → check the **Git history**
 - *"It's complicated for users to check the history"* → send users an email with a clear **résumé of today's commits**
 - *"Users will have to wait for commits to be checked"* → incremental builds are fast, it's a little more wait for a good cause

- Assumptions:
 - Using [GitHub/GitLab](#)
 - Every user has [her own remote repository](#) and pushes to it only
 - Users make a [“pull request”](#) to the main repository
- Automatic build system [checks and labels](#) pull requests:
 - *e.g.* [build-osx-ok](#) [build-ubu1404-broken](#) *etc.*
- Administrators of main repo know if the pull request cleanly compiles
 - They know if they can merge it safely
- *CMS workflow - see presentation from [Giulio](#)*

Thank you