http://www.geant4.org

ATLAS GIT developer model

Andrea Dotti (adotti@slac.stanford.edu) ; SD/EPP/Computing

U.S. DEPARTMENT OF ENERGY
Office of Science

SLAC NATIONAL ACCELERATOR LABORATORY

Goal

ATLAS has a similar development model as Geant4 (based on tags, and collection of tags)

They migrated recently to git from SVN

We could profit from their experience

See https://jira-geant4.kek.jp/projects/GIT/issues/GIT-9?filter=allopenissues for a complete discussion

ATLAS & git

Migrated to GIT from SVN

ATLAS had a very similar development model as Geant4:

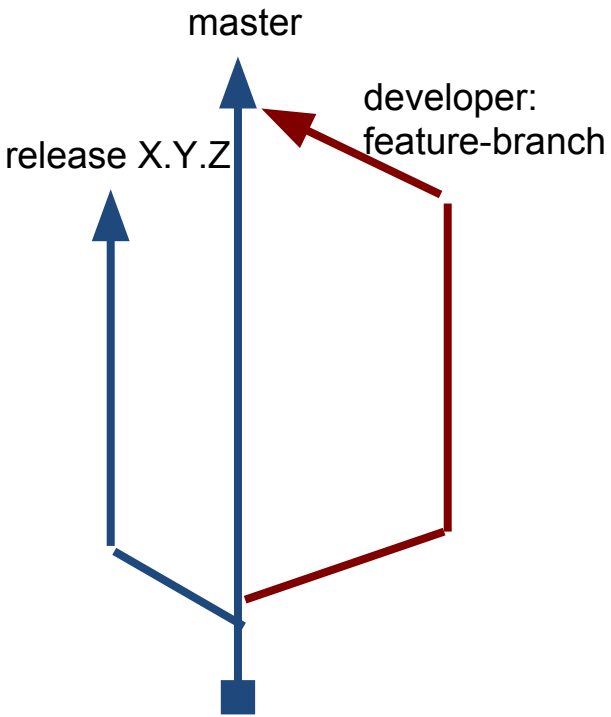
1. Based on SVN and tags per directory/package (i.e. equivalent to G4 categories)
2. A *tagcollector* (i.e. equivalent to our tags database) to collect tags for release
3. A testing system based on nightlies

Important difference ATLAS vs Geant4:
Very large code base (x5 times ours), very large developer community, thus many aspects of ATLAS development work-mode are in place to support this (e.g. three separate layers of *shifter*, equivalent to our system testing)

With the migration to git the concept of tags is replaced by **Merge Requests.** Built-in into git{hub,lab}

General feedback from ATLAS is very positive, despite some rough beginnings (one developer told me: “At beginning I was skeptical about the move from SVN to GIT for ATLAS, now I am fully supporting it”)

ATLAS development



Each developer forks repository (probably not needed for G4): cope with large number of people involved

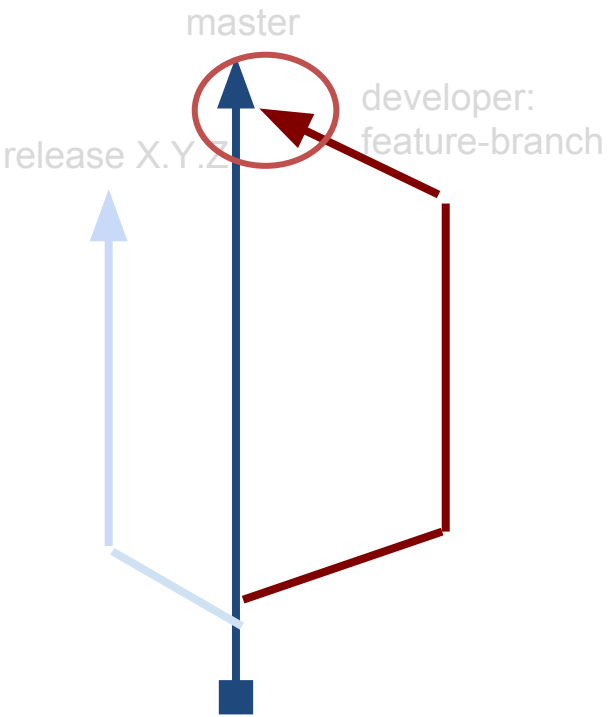
MRs are initiated **on the master of the shared** collaboration repo

- Branches exists for the various releases being supported

Master should always be stable (G4 equivalent: latest reference + all accepted tags)

- A git tag is done on master each evening

ATLAS development: proposing



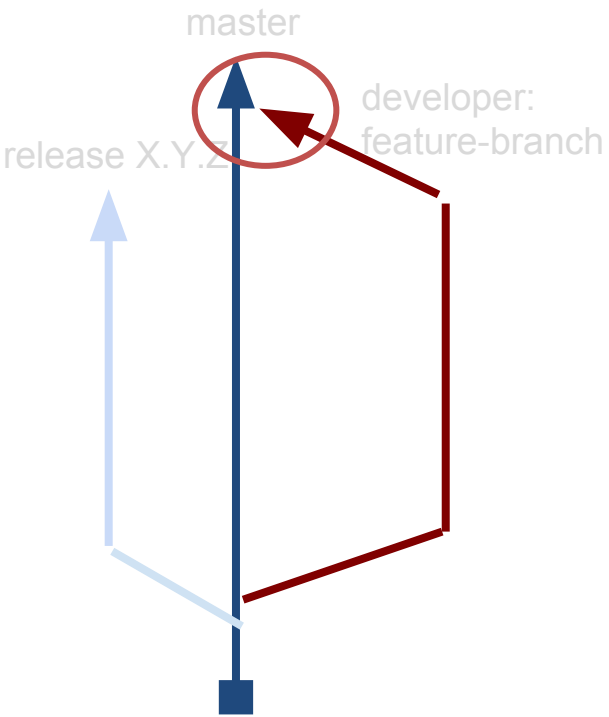
A MR is the act of **requesting the code from the developer to be merged back into the shared repository**

Important:

- The MR cannot be done if code conflicts are present (automatic procedure)
- The MR does not change the repository unless all tests are passed (a human must *accept* after reviewing code and testing results)

A MR is similar, in today system, to *proposing a tag*

ATLAS development: testing



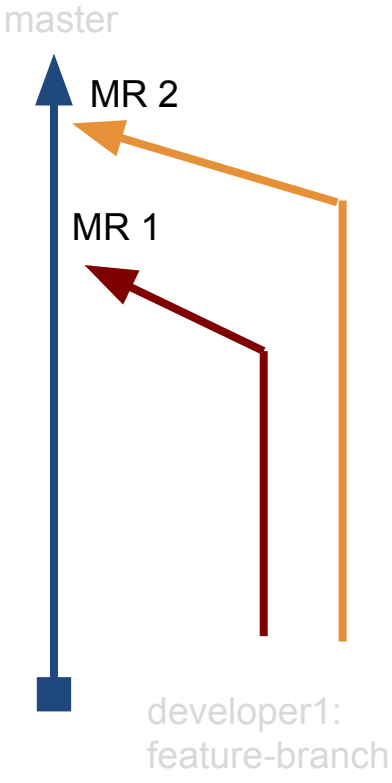
A MR will trigger automatic testing:
On a dedicated machine the master is checkout, the MR is applied, the code is built and relevant tests are run (our *continuous testing*). This is called CI (Continuous Integration)

If everything is green, the MR is reviewed by a shifter (in ATLAS there are 3 levels of shifters: non-expert, expert, guru; the equivalent of our stt shifter)

If the reviewer(s) gives also the green light, the MR has passed testing and goes to the final review by the release coordinator

This process is similar, in today system, to *selecting a tag*

ATLAS development: merging



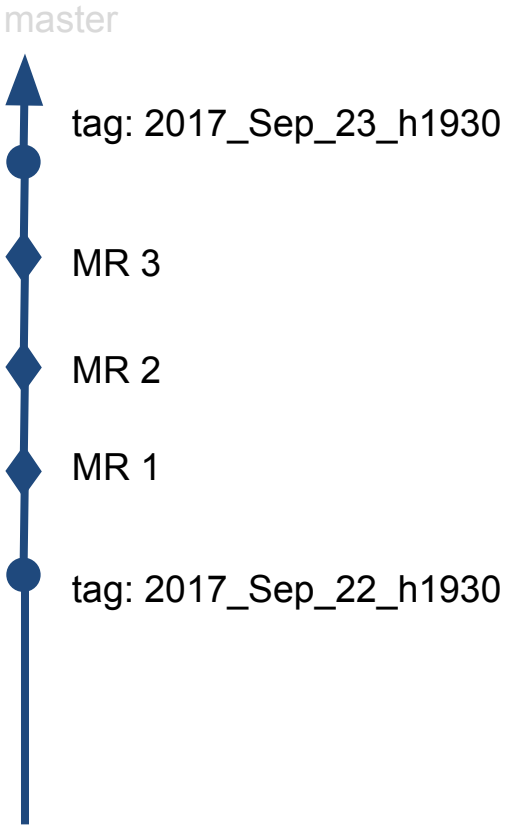
MR can be processed in **parallel**:

1. On one machine only MR1 is applied to the master and the procedure of the previous slides applies
2. On another machine only MR2 is applied to the master

The release coordinator accepts in sequence the MR in the master, but can accept MR2 before MR1 is accepted (e.g. if MR1 needs refinement from the author).
Possible conflicts may now arise, but these are rare even in the case of ATLAS.

Accepting the merge request by release coordinator is similar, in today system, to *accepting a tag*

The role of nightlies



Each evening a tag on the current master (i.e. including all accepted MR) is created and a full test suite with longer applications is run

It may happen that a new error appears in this phase, the release coordinator does a `git revert` to remove the offending MR and requests the developer to fix the issue

This process is similar to *rejecting* a tag

Conclusions

ATLAS system can be adapted (after simplifications) to Geant4

It similar to our current system and thus should ease transition

The role of developers is increased: they are responsible to checking the *continuous results* (if not green the shifter does not even see the MR)

The role of continuous integration is fundamental and we should review the list of our tests: move some of the faster examples to continuous?

Thank you

S. Farrell, V. Tsulaia, C. Leggett @ Berkeley Lab for their time to discuss this issues