

Git For Analysis

ATLAS Software Tutorials October 2019

Adam Parker

Based on slides from K. Suruliz

Introduction



In the previous talk, you learned about git, what it's used for, its basic features and commands (git clone, git add, git commit)

- The focus was mainly on offline release development
- The workflow for user analysis code is a bit different (simpler!)
- The main ATLAS git documentation can be found [HERE](#)

In this talk:

- Focus on git for analysis software
- An example of how you would version control your personal projects
- You will be able to try things out yourself in the hands-on session
- The focus was mainly on offline release development
- Detailed documentation (+ hands-on [SoftwareTutorialAnalysisInGitReleases](#))

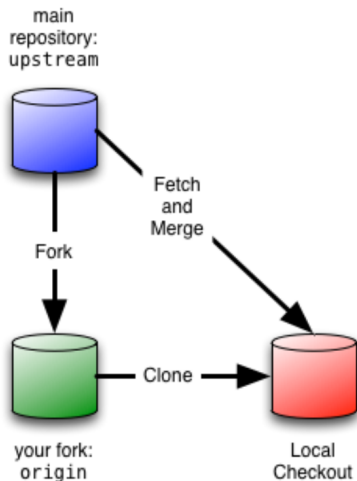
Development Workflow



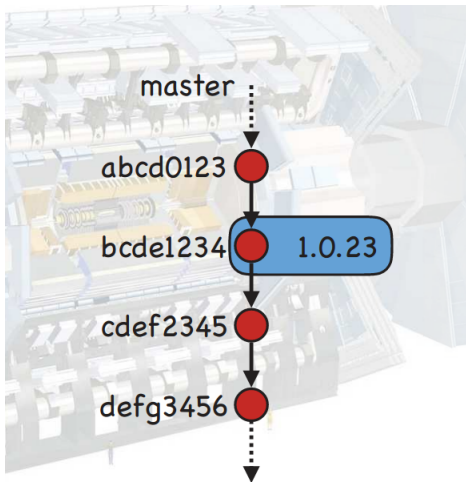
- For Athena development, we rely on forks a lot.

Each user has their fork (~copy) of the entire athena repository

- This is necessary when you have a large number of developers all contributing to one project



Managing a Private Project



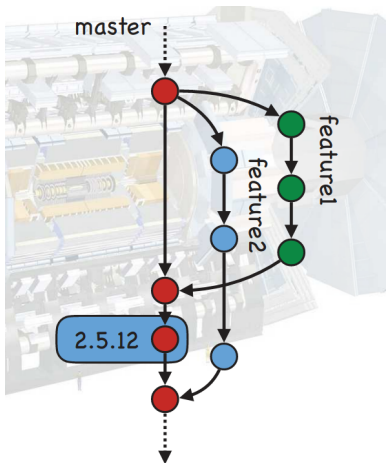
- If you're the only one managing a project, you don't even need to know about git branches too much

It is quite rare you will need multiple "production versions" of your own code

- You should just keep developing your code in the master branch of your repository

Possibly making some tags as you go along

Managing a Group Project



- If the group is small and the development is in an absolutely one track way you can still use the private repository workflow

Note, all developers require the correct access to the repository

- A better idea would be to have developers work in feature branches

This is not a separate fork but inside the *main* repository

Bottom line is, you should not use forks unless you have >50 developers

Groups and Projects



The screenshot shows the GitLab web interface for the 'MIA' group. The left sidebar contains navigation links: Overview, Details, Activity, Contribution Analytics, Issues (0), Merge Requests (1), and Members. The main content area displays the 'MIA' group details, including the group ID (7177) and a 'Leave group' link. Below this, there is a section for 'Subgroups and projects' with tabs for 'Shared projects' and 'Archived projects'. A search bar and a 'Last created' dropdown are also present. The list of projects includes:

Project Name	Description	Stars	Created
MIAPlotter	Base plotting classes, mainly python	0	8 months ago
HbbTrainMVA	Training for MVAs for MIA	1	8 months ago
HbbMVA	MVA access for MIA	0	8 months ago
MIASetup	Setup for MIA	3	8 months ago
MIA	MIA main package	5	1 month ago

Can control access on the level of groups or the level of projects contained within those groups

Permissions



You have seen already how to create a new project in GitLab. You need to set **permissions** correctly so others can access it.

The screenshot shows the GitLab interface for the 'MIA' group's 'Members' page. The page title is 'Members' and it indicates 'Existing 19' members. Below this, there is a section titled 'Members with access to MIA' with a search bar and a 'Sort by' dropdown set to 'Name, ascending'. The list of members is as follows:

Name	Username	Access Level	When given access
Adam Elliott Jasan	@ajasan	Developer	Given access 1 year ago
Adam Jackson Parker	@aparker	Developer	Given access 3 years ago
Alice Alfonsi	@aalfonsi	Developer	Given access 1 year ago
Andrew Mehta	@amehta	Developer	Given access 2 years ago
Antonio Manuel Mendes Jacques Da Costa	@amendesj	Developer	Given access 1 year ago
Carl Gwilliam	@gwilliam	Owner	Given access 3 years ago

Note the different levels of permissions. Roughly speaking, **reporters** have read-only access (should include the rest of ATLAS, generally), **developers** can push code/create merge requests, and **maintainers** can manage the repository

Commits, Tags and Hashes



- Each commit of a repository (athena or your project) is uniquely identified by a **hash**.

You can think of this as corresponding to a particular snapshot of the repository

- It is possible to checkout a package at a certain commit point using this hash.

Projects Groups More Search or jump to...

MIA > MIA > Commits

master MIA Filter by commit message

22 Oct, 2019 4 commits

- fixed SF for additional jets in VHcc Antonio Jacques Costa authored 1 day ago 54bbc752
- fix constructor problem Andrew Mehta authored 1 day ago e0bc05fa
- fixed some formatting issues Andrew Mehta authored 1 day ago 2c1da09d
- fixed issue with selection tool reverting to 60% wp. Disabled filling correct btagbin Andrew Mehta authored 1 day ago 01502e9a

21 Oct, 2019 1 commit

- add continuous option for FtagEffWweight and fix bug with default tagger assignment Andrew Mehta authored 2 days ago eaf40662

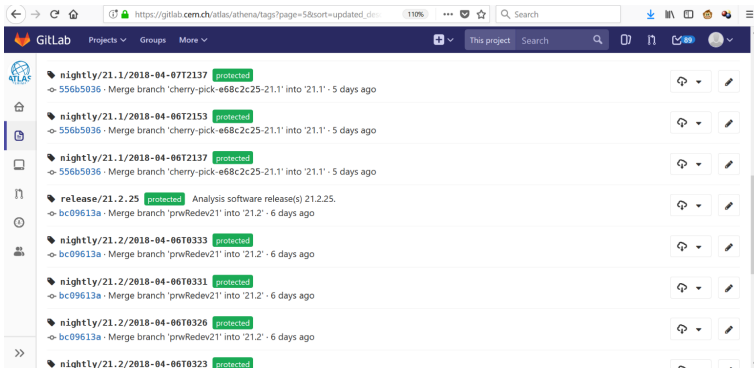
19 Oct, 2019 3 commits

- Added btagging systematics when running in ctagging mode Andrew Mehta authored 4 days ago f3849f7f

Commits, Tags and Hashes



Tags can be used to mark release points



Syntax for checking out a branch or particular tag:
git checkout <branch or tag name>, e.g.
git checkout release/21.2.93

Project Structure: SubModules



The basic philosophy is that projects should be entirely self-contained. There should be no need to tell users to manually check out additional packages needed for the software to work

This is where the concept of a **submodules** comes in.

Submodules are nested repositories inside a parent git repository at a specific tag or snapshot of time.

There are often situations where you need to use a:

- a package from another ATLAS member
- an updated R&D version of a CP package
- your own helper library or executables/macros.

This means packages can work out of the box and be completely self-contained.

You will see this in action in the hands-on

Project Structure: Example



S

susyanalysis

☆ Star 0 ƶ Fork 0 KRBS https://@gitlab.cern.ch:8443/i + Global

Files (24.6 MB) Commits (82) Branch (1) Tags (0) CI configuration Add Changelog Add License Add Contribution guide

passed 9c8bcace Fixing CMakeLists · a week ago by Kerim Suruliz

master susyanalysis /

Find file

Name	Last commit > 9c8bcace · a week ago · Fixing CMakeLists · History	Last Update
JetSimTools	cmake migration	2 months ago
JetSmearing @ 4b1fa011		
Mt2	Adding mt2 package	2 months ago
NNLOReweighter		2 months ago
StopPolarizationRewighting @ 79df0f72		
SusySusx	Fixing CMakeLists	a week ago
.gitlab-ci.yml	Fixing CMakeLists	a week ago
.gitmodules	Adding JetSmearing	3 weeks ago
CMakeLists.txt	Fixed RestFrames following Attila's recipe	a month ago
externals.cmake	Fixed RestFrames following Attila's recipe	a month ago

A package containing a library

A submodule

Packages containing the main analysis executable(s)

See Attila's talk this morning

Git is very powerful and has a lot of great features useful for analysis software development.

- However there is a lot to learn and the terminology is not always easy to understand forks, branches, tags, hashes, commits, origin, upstream, master...
- Many good online tutorials in addition to the ATLAS git tutorial, I suggest going through some of these
- Do spend a bit of time learning about git and acquiring good habits when you're starting

More on git/gitlab in Giordons talk later this afternoon



Backup