EDWARD MOYSE

(RE-USING SLIDES PREPARED BY CHRISTIAN GUMPERT, GRAEME STEWART ETC)

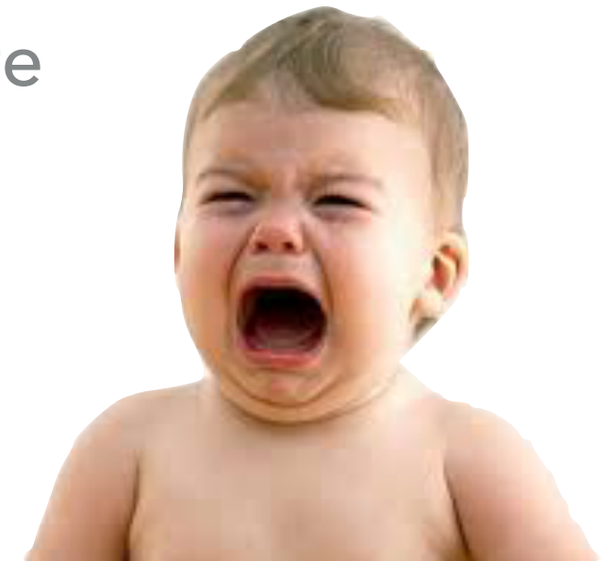
SOFTWARE VERSION CONTROL

THE ONE SLIDE SUMMARY

- The most important message of this tutorial is simple…

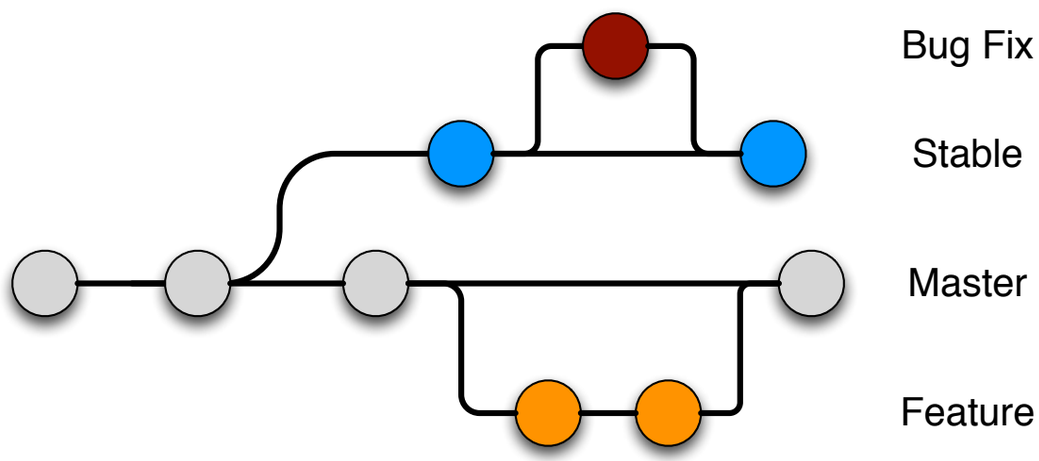
  Use a software revision control system for all of your code

- And that means ***now***…
  - …not tomorrow or next week
- Because if you wait until you need it, it will be too late
- And this will cause you pain :-(



REVISION/VERSION CONTROL SYSTEMS

- Software revision/version control is a piece of software that records different versions of code in a **persistent** and **recoverable** way.
- Many advantages:
  - **Recovery**: so you can recover a working version from the past
    - It's very easy, during software development, to break your code and not quite remember how to put it back together
  - **Analysis**: Compare different versions
    - Helps understand why your results might be different
  - **Branching**: keep different lines of development separate
    - Use one *branch* to fix a few urgent bugs
    - Use a different *branch* to develop a new feature
    - Merge them once the new feature is complete
    - *Hugely helps collaboration*



A BRIEF HISTORY OF REVISION CONTROL

- The first RCS were designed to be used on large systems where everyone logged into the same machine
  - They tracked code on the same filesystem where it lived (e.g., in a subdirectory)
  - **SCCS** and **RCS** are examples
- Then client-server systems were developed, so that developers could work on their own machines
  - Checking code into a central server to share and collaborate
  - **CVS** and **SVN** are examples (and ATLAS has used both)
- More recently **distributed** version control systems have arisen
  - These are *decentralised*, so everyone has a complete copy of the repository
  - Gives a lot of freedom to developers to share and merge as they like, so liked very much by the open source community
  - **git**, **mercurial** and **bit keeper** are examples

SOCIAL CODING

- Distributed version control systems are great, but they are made even better by using a social coding website
  - These sites allow developers
    - Browse code easily
    - Compare different versions
    - Take copies (a.k.a. fork)
    - Offer patches back to upstream repositories
      - And discuss and review these patches before acceptance
- The best known social coding site is GitHub, but there are others, e.g., BitBucket and GitLab

GitHub

Atlassian Bitbucket

GitLab

ONE MODERN SOLUTION

GitLab

- It's **definitely** better to pick a modern distributed version control system, backed up by a good social coding website
  - CERN no longer supports SVN + CVS.
- A **much** better choice is **git** + **gitlab**
  - **git** is the most popular open source VCS by quite some way
    - It overtook SVN in 2014, so most people know how to use it
    - It can host huge projects (Linux Kernel, CMSSW, athena (+ atlasexternals))
    - It scales very well and it's extremely fast and powerful
  - A **GitLab** service is offered by CERN
    - Has pretty much every feature that GitHub has
    - Integrated with SSO, easy to keep the repository private (except to your ATLAS colleagues)
    - Most ATLAS SW projects, including athena (https://gitlab.cern.ch/atlas/athena) , are hosted on gitlab
      - It IS okay to use github etc too, but you must be aware of the implications with regards to data/analysis code privacy - gitlab is probably safer & **ask** if you're not sure

GETTING STARTED ON GITLAB

gitlab.cern.ch

Computing an... Big Science ATLAS Software Agenda LHC Atlas Software Computing Tools and Reference Entertainment Misc Apple Yahoo! Google Maps YouTube

GitLab Projects Groups Activity Milestones Snippets Search or jump to...

You pushed to `master-fix-MuonSegmentMomentum` at Edward Moyse / athena 6 minutes ago

Create merge request

Projects

New project

Your projects 1,000+ Starred projects 12 Explore projects

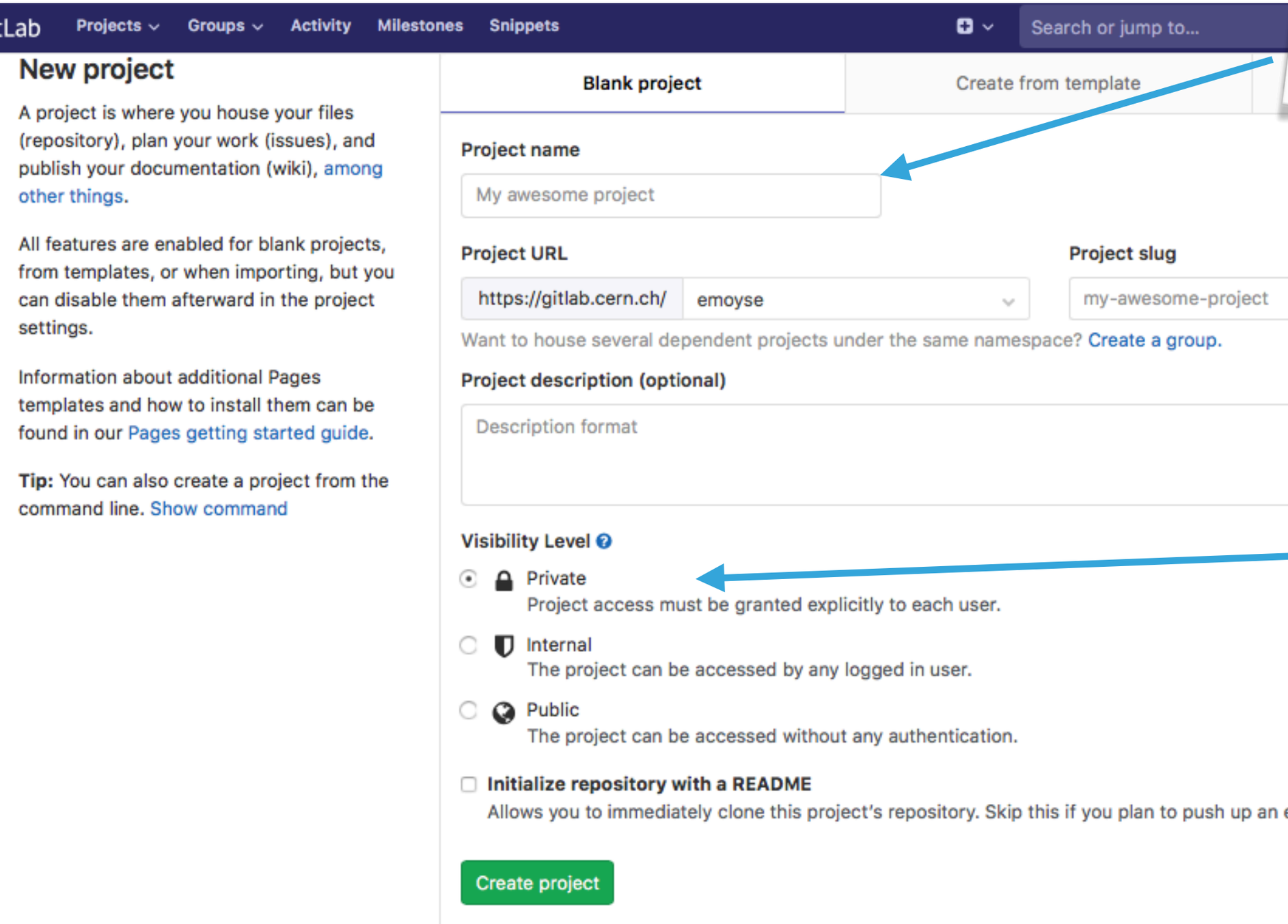
Filter by name... Last updated

All Personal

A	atlas-sw-git / **atlassoftwaredocs** Owner ATLAS Offline Software Documentation		22	13	6	2	
ATLAS	atlas-phys-susy-wg / RPVLL / **athena** Reporter The ATLAS Experiment's main offline software repository		0	0	0	0	
ATLAS	Edward Moyse / **athena** Maintainer		0	0	0	0	...utes ago
H	HZZ / HZZSoftware / **HZZAnalRun2Code** Reporter		4	1	0	2	Updated 14 minutes ago
A	atlas-physics-office / HIGG / ANA-HIGG-2018-43 / **ANA-HIGG-2018-43-INT1** Reporter		0	0	0	0	Updated 24 minutes ago
ATLAS	atlas / **athena** Owner The ATLAS Experiment's main offline software repository		71	814	162	0	Updated 6 minutes ago

Create a new project

CREATING A NEW REPOSITORY/PROJECT

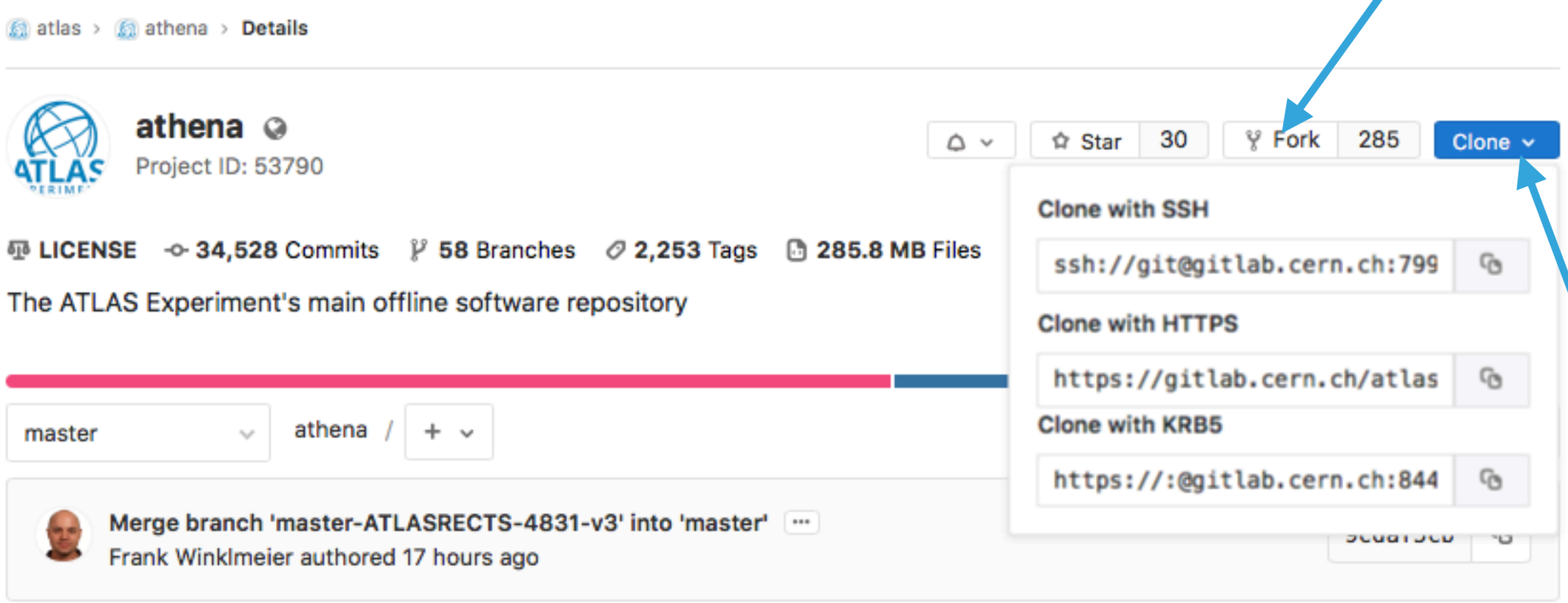


Make the name **descriptive**

Remember to abide by ATLAS Information Protection!

EXISTING REPO

Forking a project gives you a personal copy of it, hosted on gitlab



- You can **clone** an existing repo (copy it locally) or **fork** it (make a new copy on gitlab and then clone that fork)
- In ATLAS we typically (but not always) recommend forking a repository and then making a 'merge request' back to the original
- The latest version of Athena has been forked >800 times

Clone (copy locally) this repository

Different protocols have different URLs (all work, I personally prefer ssh)

CLONE AND CHECKOUT

- Once you have your shiny new repo (either by forking, or starting from scratch) you'll want to do something with it
- Mostly this requires taking a copy of it locally
  - `git clone`

```
teal:~$ git clone https://:@gitlab.cern.ch:8443/graemes/atlasgit.git
Cloning into 'atlasgit'...
remote: Counting objects: 488, done.
remote: Compressing objects: 100% (235/235), done.
remote: Total 488 (delta 269), reused 418 (delta 228)
Receiving objects: 100% (488/488), 129.29 KiB | 0 bytes/s, done.
Resolving deltas: 100% (269/269), done.
Checking connectivity... done.
teal:~$ ls atlasgit/
README.md                       cmaketags.py
asvn2git.py                     cmttags.py
atlasoffline-exceptions.txt     glogger.py
atutils.py                      license.txt
branchbuilder.py                tjson2txt.py
casefilter.sh
teal:~$
```

There is a subtlety here:

- ***clone*** is making a local copy of the repo;
- ***checkout*** is creating a working version from your local copy

However, the default '**clone**' folds in a checkout as well

WHERE AM I? WHAT'S GOING ON?

- git has useful commands to help

```
teal:~/atlasgit$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
teal:~/atlasgit$
```

```
teal:~/atlasgit$ git log -n 1
commit df5d44280bcfa5df382764b284985314e263dbd6
Author: Graeme A Stewart <graeme.andrew.stewart@cern.ch>
Date:   Wed Sep 14 09:04:28 2016 +0200

    Updated documentation and start support for patch branches
```

```
teal:~/atlasgit$ git diff origin/master
diff --git a/README.md b/README.md
index 2018bfc..9b3c34d 100644
--- a/README.md
+++ b/README.md
@@ -190,3 +190,6 @@ Final Notes

 You should not attempt to push to the upstream reopository
 during the Zombie Apocolypse.
+
+If you encounter fast zombies (see 28 Days Later) commit
+directly and do not use the stage area.
teal:~/atlasgit$
```

MAKE SOME CHANGES...

- Note that git tells you what to do to accept these changes (`git add`) or to roll them back (`git checkout`)
- It's well worth reading the messages carefully - git tries to help

```
teal:~/atlasgit$ emacs README.md
teal:~/atlasgit$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
teal:~/atlasgit$
```

STAGE AND COMMIT

- Git has a staging area, where files go when they are added
- When you commit, the files that are staged (and only those) are stored in the repository
- Remember to write a good commit message
  - Single line summarising the change
  - Separate paragraphs for details

```
teal:~/atlasgit$ git add README.md
teal:~/atlasgit$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   README.md

teal:~/atlasgit$ git commit
[master c5abba4] Add warning for Zombie Apocolypse
 1 file changed, 6 insertions(+)
teal:~/atlasgit$
```

PUSH YOUR CHANGES

- After a commit your changes are in your copy of the repository only
  - Which is now one commit ahead of the remote copy on GitLab
  - So if you want other people to see the changes you need to move them to GitLab
  - In git this is called a *push*

```
teal:~/atlasgit$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
nothing to commit, working directory clean
teal:~/atlasgit$ git push origin
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 434 bytes | 0 bytes/s, done.
Total 3 (delta 2), reused 0 (delta 0)
To https://:@gitlab.cern.ch:8443/graemes/atlasgit.git
   df5d442..c5abba4  master -> master
teal:~/atlasgit$
```

EXAMINE YOUR CHANGES ON GITLAB

Graeme Stewart / atlasgit

This project Search

Project Activity Repository Pipelines Graphs Issues Merge Requests 0

Authored by **Graeme A Stewart** 12 minutes ago

Browse Files Options

Commit c5abba469f0e6fdfe8f089471d2d408cd8a764c6 1 parent df5d4428 master

Add warning for Zombie Apocolypse

README.md now covers this case

Showing **1 changed file** with **6 additions** and **0 deletions**

Hide whitespace changes Inline Side-by-side

README.md View file @c5abba4

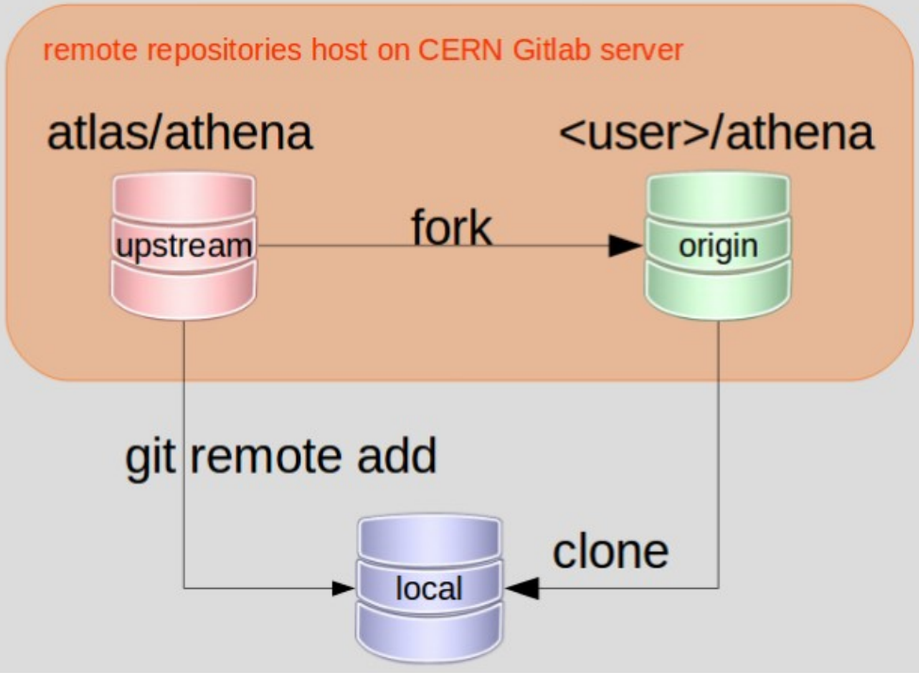
```
@@ -184,3 +184,9 @@ or

 Note that in the last case (`--tags`) make sure you _delete_ all tags in `import/`,
 as these are not needed post-import and they substantially degrade performance.
+
+Final Notes
+---
+
+You should not attempt to push to the upstream reopository
+during the Zombie Apocolypse.
```

RECAP

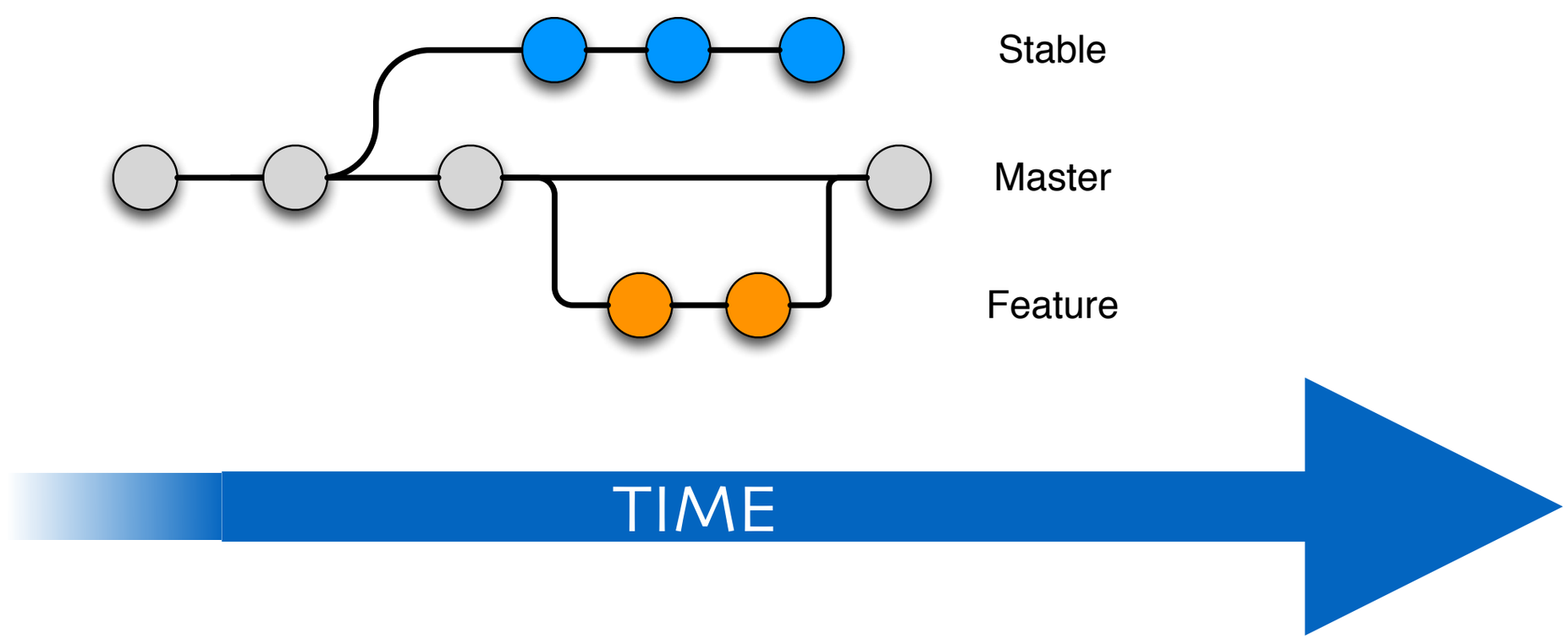
- Your **working copy** holds changes you just made
- The **stage area** holds file versions that will go to the next commit
- The **local repo** holds all the commits you made locally
- The **remote repo** (usually called **origin**) can have a different state from your local repo
  - push to move your **local repo** changes to **origin**
  - pull to take changes from **origin** to your local repo
- To get your changes into the release (**upstream**) means making a **merge request** (which is not covered today) [merge request tutorial]

- **atlas/athena (aka upstream)**
  The official ATLAS code repository from which all nightlies and releases are built
- **<user>/athena (aka origin)**
  Your private copy of Athena. Mainly used as *container* to publish your changes and create **merge requests** such that your changes get into the official repository.
- **local working copy**
  A fully functional, local git repository to work with.



BRANCHES

- git can branch code incredibly quickly and efficiently
  - Which is wonderful for insulating different changes (fixes and features)
- It's strongly advised to use a branch to work on all changes
  - Then merge back into the 'main' branch only when ready



USING BRANCHES

- git branch -l
  - List all branches
- git checkout [-b] BRANCH
  - Switch to branch
  - -b create it if needed
- git merge BRANCH
  - Merge changes *from* BRANCH to the currently checkedout branch

```
teal:~/atlasgit$ git checkout -b fast_zombie_fix
Switched to a new branch 'fast_zombie_fix'
teal:~/atlasgit$ git status
On branch fast_zombie_fix
nothing to commit, working directory clean
teal:~/atlasgit$ emacs README.md
teal:~/atlasgit$ git add README.md
teal:~/atlasgit$ git commit -m "Fast zombie notes"
[fast_zombie_fix 556f4b2] Fast zombie notes
 1 file changed, 3 insertions(+)
teal:~/atlasgit$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.
teal:~/atlasgit$ git merge fast_zombie_fix
Updating c5abba4..556f4b2
Fast-forward
 README.md | 3 +++
 1 file changed, 3 insertions(+)
teal:~/atlasgit$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
nothing to commit, working directory clean
teal:~/atlasgit$
```

Can look at branches in gitlab

atlas › athena › Repository › **Branches**

Overview Active Stale All

Filter by branch name | Delete merged branches | New branch

Protected branches can be managed in project settings.

Active branches

21.2 protected
1da78a99 · Merge branch '21.2-IsolationCorrectionsDDNewDefaultPath' into '21.2' · 6 minutes ago
999+ 999+ | Merge request | Compare

master default protected
dfcc4b8b · Merge branch 'master' into 'master' · 1 hour ago

pufittrack
4377aa23 · adding the pufittrack algorithm · 1 hour ago
999+ 999+ | Merge request | Compare

21.0-TrigMC protected
31882be2 · Merge branch 'cherry-pick-6067798f-21.0-TrigMC' into '21.0-TrigMC' · 11 hours ago
999+ 999+ | Merge request | Compare

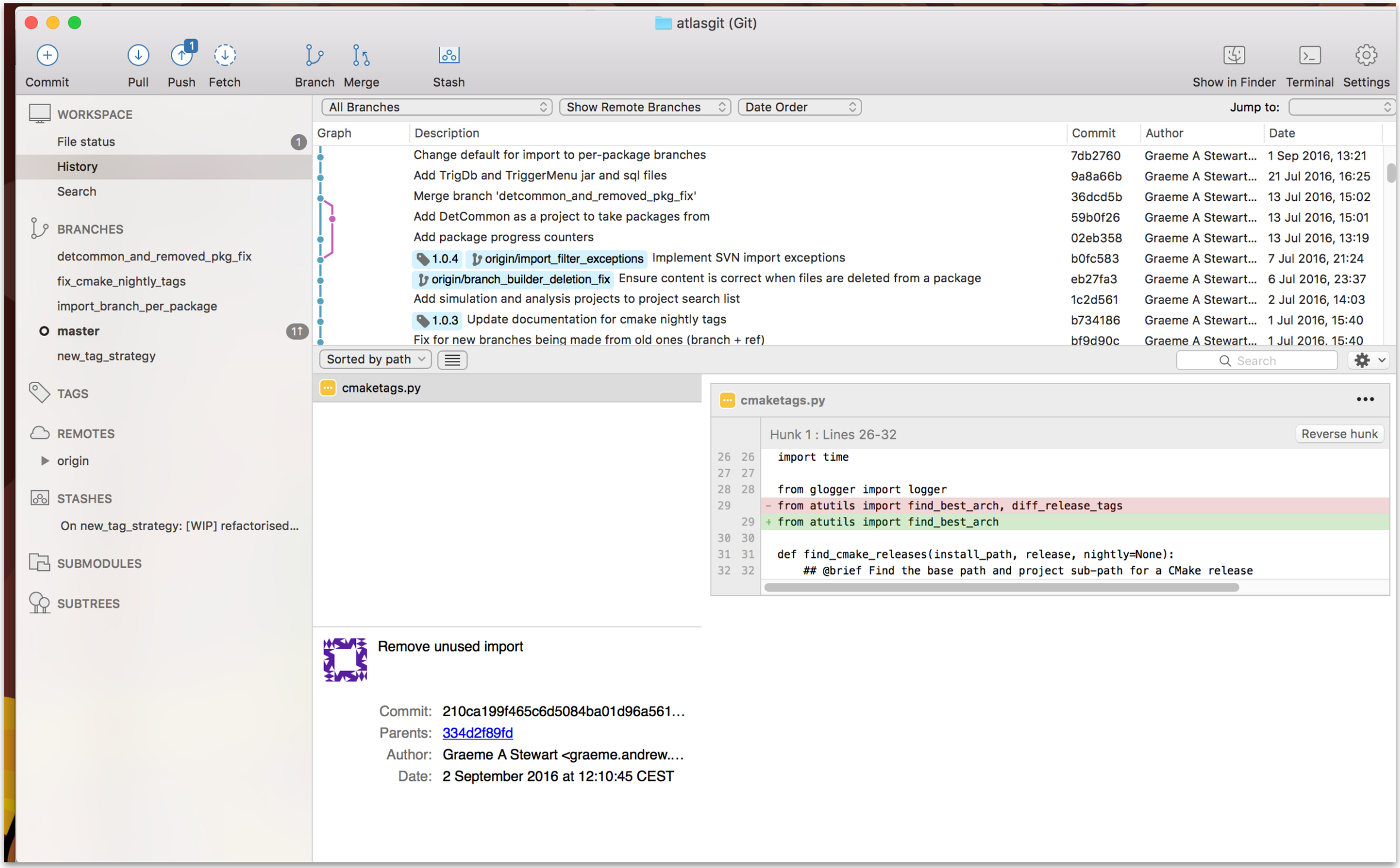
21.1 protected
31b94de2 · Merge branch 'TM-16-10-18' into '21.1' · 13 hours ago
999+ 999+ | Merge request | Compare

Show more active branches

GIT GUIS

- **git is a brilliant tool**
- It is incredibly powerful … but as a consequence it can sometimes be intimidating (especially if you started out with SVN)
  - (and also the naming of its command is sometimes a bit odd)
- As a popular open source tool there are a ton of great GUIs you can use
  - GitHub Desktop
  - SourceTree
  - GitKraken
- These are highly recommended as they can make branching, merging, diffing, etc. all a lot clearer
  - (However, to really understand git it's still worth getting your hands dirty!)

SOURCE TREE


atlasgit (Git)
Commit
Pull
Push
Fetch
Branch
Merge
Stash
Show in Finder
Terminal
Settings
WORKSPACE
File status
History
Search
BRANCHES
detcommon_and_removed_pkg_fix
fix_cmake_nightly_tags
import_branch_per_package
master
new_tag_strategy
TAGS
REMOTES
origin
STASHES
On new_tag_strategy: [WIP] refactorised...
SUBMODULES
SUBTREES
All Branches
Show Remote Branches
Date Order
Jump to:
Graph
Description
Commit
Author
Date
Change default for import to per-package branches 7db2760 Graeme A Stewart... 1 Sep 2016, 13:21
Add TrigDb and TriggerMenu jar and sql files 9a8a66b Graeme A Stewart... 21 Jul 2016, 16:25
Merge branch 'detcommon_and_removed_pkg_fix' 36dcd5b Graeme A Stewart... 13 Jul 2016, 15:02
Add DetCommon as a project to take packages from 59b0f26 Graeme A Stewart... 13 Jul 2016, 15:01
Add package progress counters 02eb358 Graeme A Stewart... 13 Jul 2016, 13:19
1.0.4 origin/import_filter_exceptions Implement SVN import exceptions b0fc583 Graeme A Stewart... 7 Jul 2016, 21:24
origin/branch_builder_deletion_fix Ensure content is correct when files are deleted from a package eb27fa3 Graeme A Stewart... 6 Jul 2016, 23:37
Add simulation and analysis projects to project search list 1c2d561 Graeme A Stewart... 2 Jul 2016, 14:03
1.0.3 Update documentation for cmake nightly tags b734186 Graeme A Stewart... 1 Jul 2016, 15:40
Fix for new branches being made from old ones (branch + ref) bf9d90c Graeme A Stewart... 1 Jul 2016, 15:40
Sorted by path
Search
cmaketags.py
cmaketags.py
Hunk 1 : Lines 26-32
Reverse hunk
import time
from glogger import logger
- from atutils import find_best_arch, diff_release_tags
+ from atutils import find_best_arch
def find_cmake_releases(install_path, release, nightly=None):
@brief Find the base path and project sub-path for a CMake release
Remove unused import
Commit: 210ca199f465c6d5084ba01d96a561...
Parents: 334d2f89fd
Author: Graeme A Stewart <graeme.andrew....
Date: 2 September 2016 at 12:10:45 CEST

- We made a **cheat sheet!**

git ATLAS Cheat Sheet

Getting started

Cloning your fork of the **entire** repository to a local disk (not on AFS, please!):
`git clone https://:@gitlab.cern.ch:8443/YOUR_USER_NAME/athena.git`

Add the main repository as upstream:
`git remote add upstream https://:@gitlab.cern.ch:8443/atlas/athena.git`

Alternative: Set up a **sparse**-checkout workdir:
`git atlas init-workdir https://@gitlab.cern.ch:8443/YOUR_USER_NAME/athena.git`

Add / List / Remove packages in your dir:
`git atlas addpkg <package> / git atlas lspkg / git atlas rmpkg <package>`

Start a new mini-project

Bring you local clone up-to-date with the main repository
`git fetch upstream`

Create a branch for your changes based on another branch (like 21.0 or master)
`git checkout –b MY_BRANCH_NAME upstream/parent_branch --no-track`

Or switch to an existing branch:
`git checkout MY_BRANCH_NAME`

Inspection:

List remotes:
`git remote –v show`

List changed files in the work-dir:
`git status`

Changes of tracked files not yet staged
`git diff`

Show (last) commit
`git show [commit-id]`

Show branches (verbose, all)
`git branch [-v] [-av]`

List tags:
`git tag -l`

Get previous commit messages:
`git log`

Local changes:

Add files for later commits
`git add [file]`

Undo the above:
`git reset[file]`

Store a snapshot added files:
`git commit`

Edit previous commit (before push):
`git commit --ammend`

Reverting a commit:
`git revert [commit-id]`

Discard local (uncommitted) changes
`git checkout [path]`

Merging:

Merge changes from *Branch_Name* into current branch
`git merge BRANCH_NAME`

Uploading changes:

Pushing your local work to your fork
`git push origin –u MY_BRANCH_NAME`

Notation

upstream: The official athena repository on gitlab the used to build nightlies and releases

origin: Your private fork of the official athena repository on gitlab **branch**: Either a release-series (like 21.0.X) or your feature branch

master: The development branch

commit id: a hash uniquely identifying the state of a repo

HEAD: The most recent commit id of the current branch

tag: A human-readable alias of a commit id

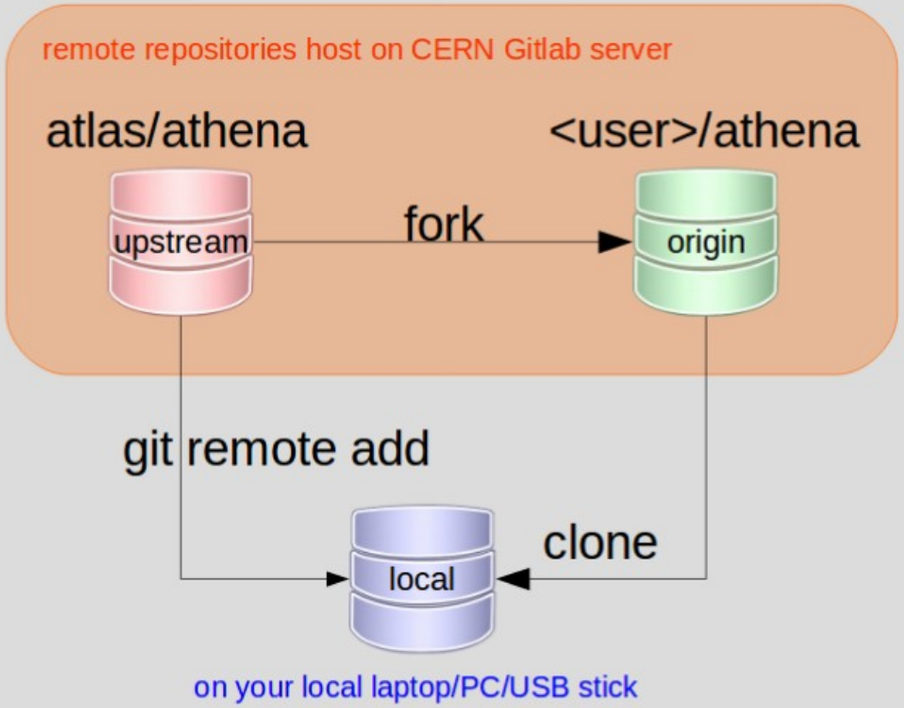
https://atlassoftwaredocs.web.cern.ch/gittutorial/GitCheatSheet.pdf

SUMMARY

- **Use git for all of your code**
- In ATLAS, we have tried to use git 'out of the box' as much as possible
  - Means that the many many many git tutorials out there on the web are directly applicable
    - e.g. https://www.atlassian.com/git/tutorials
  - Also means that the skills you are learning now are transferable
- For a more detailed tutorial on how to interact with athena (which is well worth a look, even if you aren't an athena developer!) have a look here:
  - https://atlassoftwaredocs.web.cern.ch/gittutorial/
- We also have tutorials for specific tasks (e.g. athena merge review shifts) and specific roles (e.g. developers, release coordinators)
  - Feel free to have a look, even if it doesn't directly concern you (yet!) - you will learn how it fits together
    - These pages are under active development so check back often

Software development for atlas/athena

- **atlas/athena (aka upstream)**
  The official ATLAS code repository from which all nightlies and releases are built
- **<user>/athena (aka origin)**
  Your private copy of Athena. Mainly used as *container* to publish your changes and create **merge requests** such that your changes get into the official repository.
- **local working copy**
  A fully functional, local git repository to work with.



Getting your changes into athena

- general idea:

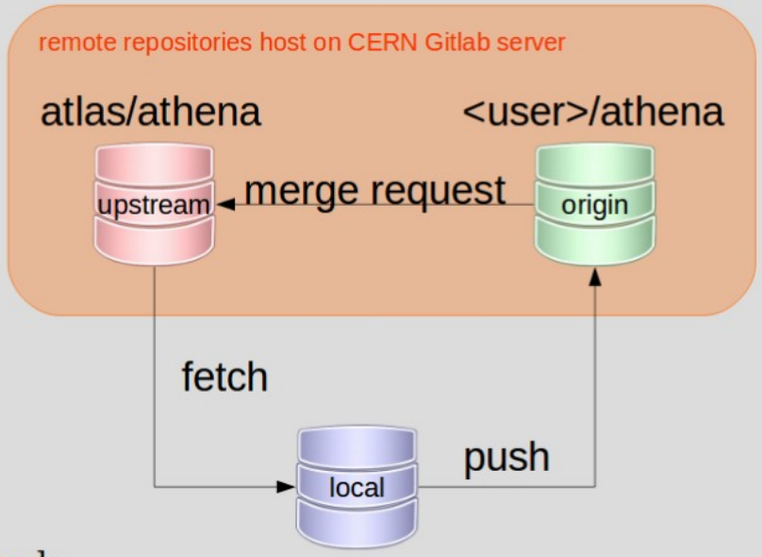
1) fetching updates from upstream
   Gives you the latest state of the official repository.

2) create a new local development branch
   Allows you to implement changes, build, test, commit…

3) push your changes to your fork
   Publish your work and make it visible to others.

4) open merge request in Gitlab
   Request to include your changes into the official ATLAS code repository such that future nightlies/releases benefit from your work.

- we support sparse checkouts using git atlas and partial builds using the AthenaWorkDir project



more details in the tutorial
atlassoftwaredocs.web.cern.ch/gittutorial/