GIT Versioning Tool

Content Overview

Content Overview

01 History and fundamental concepts behind source control

Introduction to source control

History and fundamental concepts behind source control

- A way to manage files and directories
- Track changes over time
- Recall previous versions
- ‘Source control’ is a subset of a VCS.

Introduction to source control

History and fundamental concepts behind source control

- (1972) Source Code Control System (SCCS)
  - closed source, part of UNIX

- (1982) Revision Control System(RCS)
  - - open source

- (1986) Concurrent Versions System (CVS)
  - open source

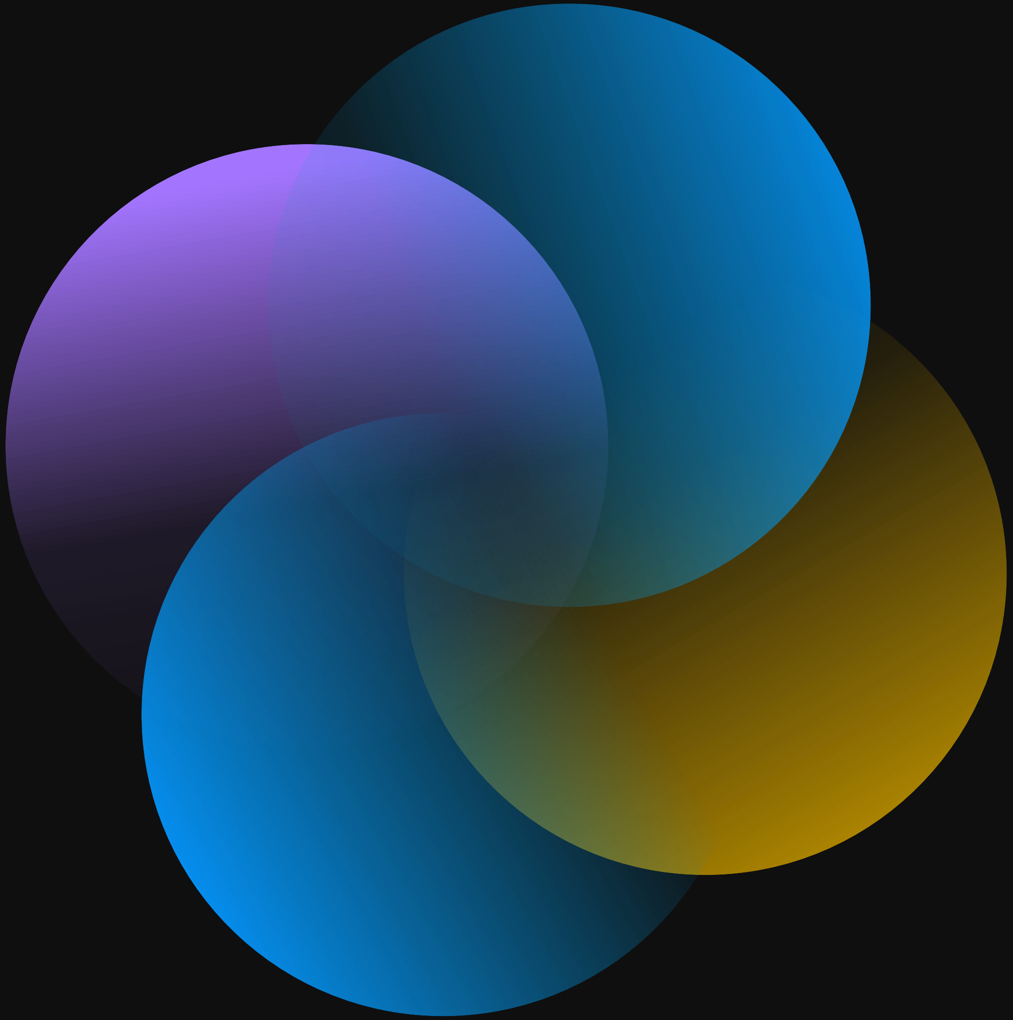
- (2000) Apache Subversion (SVN)
  - - open source

Introduction to source control

History and fundamental concepts behind source control

- (2000) BitKeeper SCM
  - closed source, proprietary, used with source code management of Linux kernel
  - free until 2005
  - distributed version control

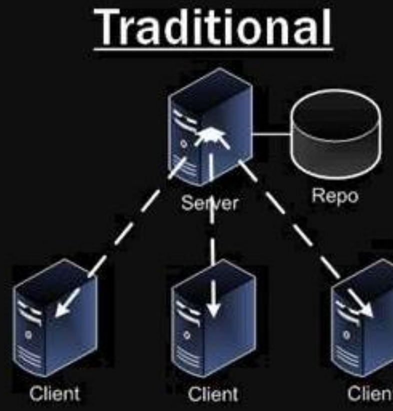
02 Centralized vs. distributed version control

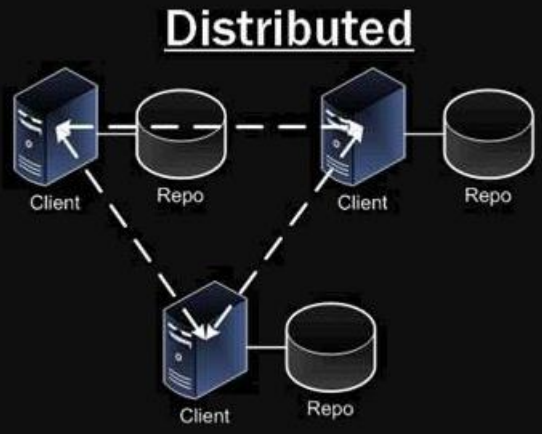


Introduction to source control

Centralized Version Control vs Distribute Version Control

Sr. No.	Key	Centralized Version Control	Distributed Version Control
1	Working	In CVS, a client need to get local copy of source from server, do the changes and commit those changes to centeral source on server.	In DVS, each client can have a local branch as well and have a complete history on it. Client need to push the changes to branch which will then be pushed to server repository.
2	Learning Curve	CVS systems are easy to learn and set up.	DVS systems are difficult for beginners. Multiple commands needs to be remembered.
3	Branche s	Working on branches in difficult in CVS. Developer often faces merge conflicts.	Working on branches in easier in DVS. Developer faces lesser conflicts.
4	Offline Access	CVS system do not provide offline access.	DVD systems are workable offline as a client copies the entire repository on their local machine.
5	Speed	CVS is slower as every command need to communicate with server.	DVS is faster as mostly user deals with local copy without hitting server everytime.
6	Backup	If CVS Server is down, developers cannot work.	If DVS server is down, developer can work using their local copies.



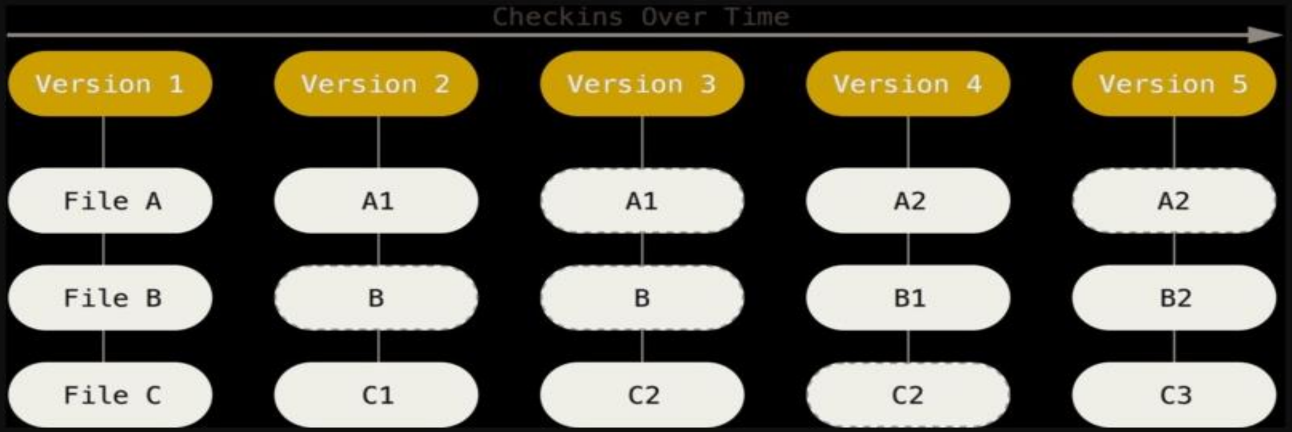


04 What is Git?

Introduction to Git

What is Git? Basic Git concepts and architecture

- Created by Linus Torvalds, April 2005
- Replacement for BitKeeper to manage Linux kernel changes
- A command line version control program
- Uses checksums to ensure data integrity
- Distributed version control (like BitKeeper)
- Cross-platform (including Windows!)
- Storing data as snapshots of the project over time
- Open source, free



Introduction to Git

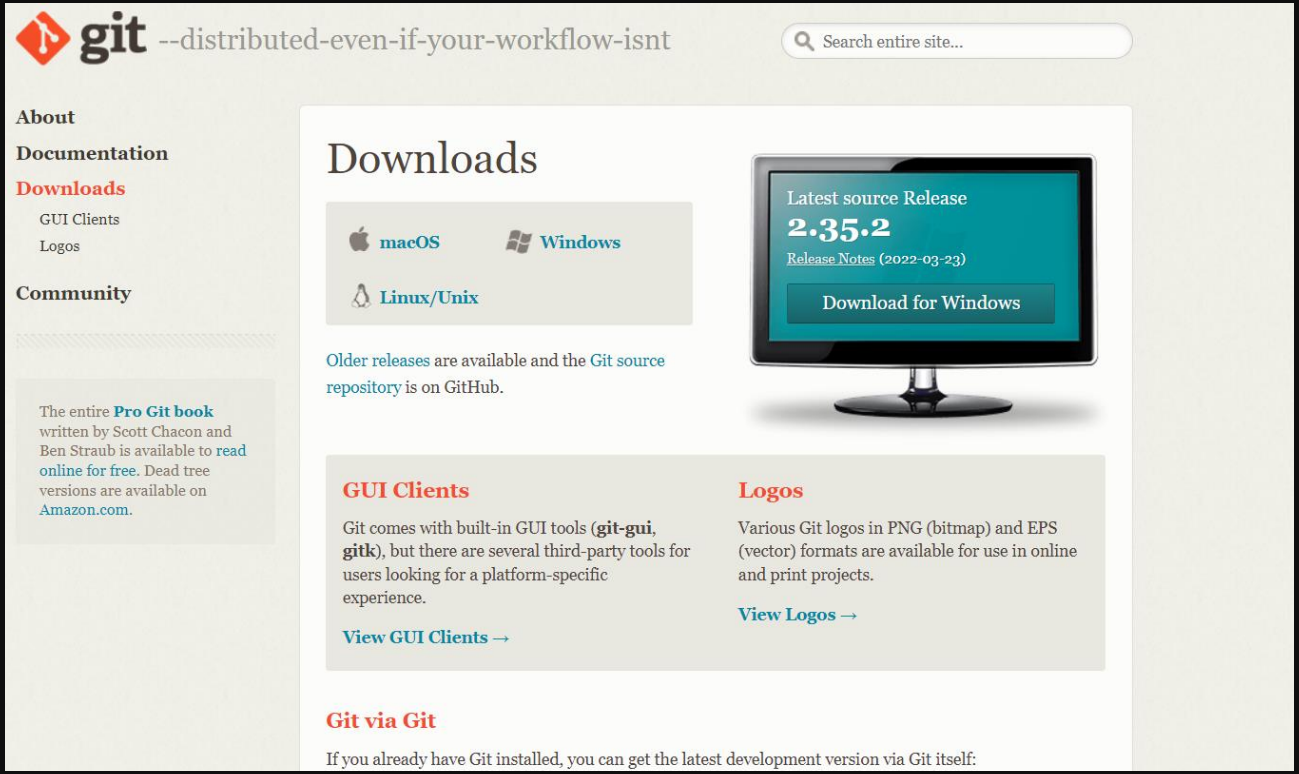
Git: Distributed Version Control System

- "If you're not distributed, you're not worth using." – Linus Torvalds
- No need to connect to central server
- Can work without internet connection
- No single failure point
- Developers can work independently and merge their work later
- Every copy of a Git repository can serve either as the server or as a client (and has complete history!)
- Git tracks changes, not versions
- Bunch of little change sets floating around

Introduction to Git

How to install Git?

- https://git-scm.com/downloads
- Steps For Installing Git for Windows.pdf



Downloads

Older releases are available and the Git source repository is on GitHub.

The entire Pro Git book written by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

GUI Clients

Git comes with built-in GUI tools (**git-gui**, **gitk**), but there are several third-party tools for users looking for a platform-specific experience.

View GUI Clients →

Logos

Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.

View Logos →

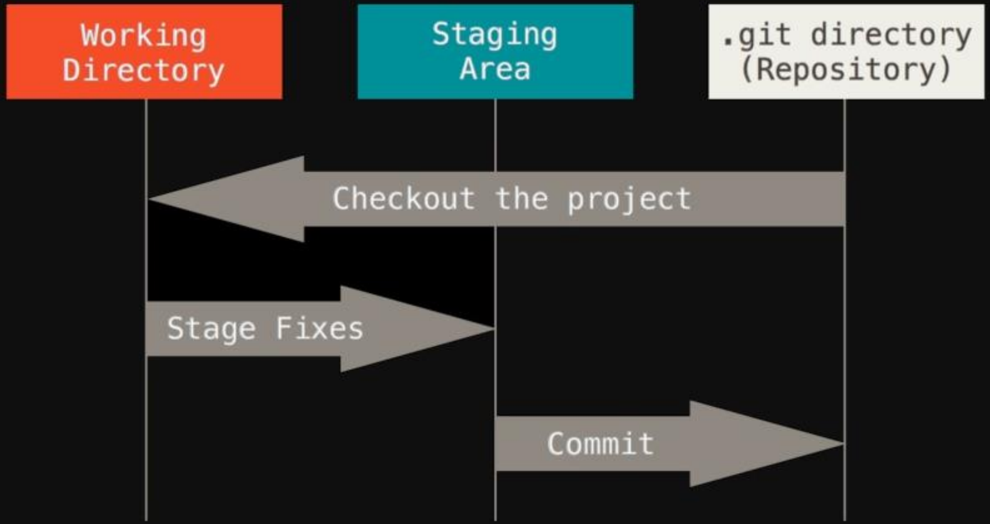
Git via Git

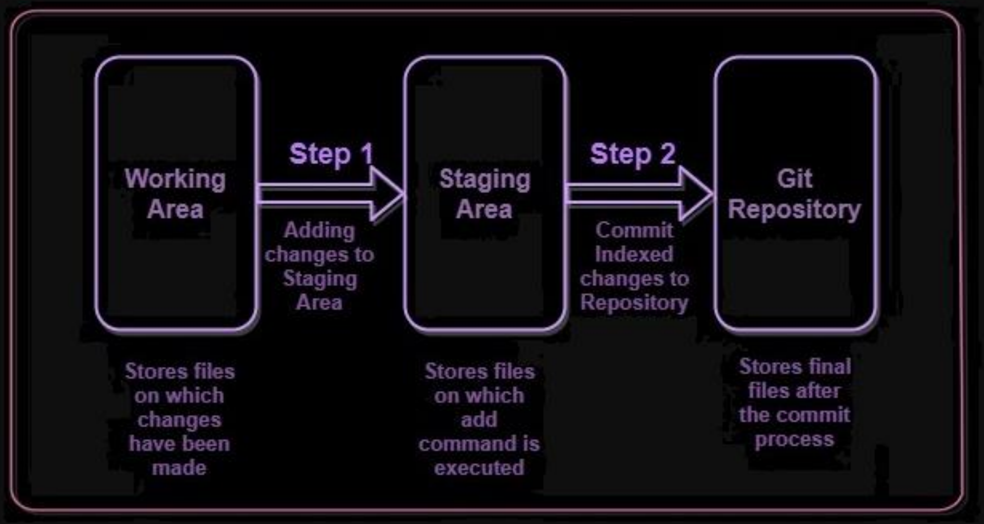
If you already have Git installed, you can get the latest development version via Git itself:

Introduction to Git

What is repository?

- “repo” = repository; usually used to organize a single project
- repos can contain folders and files, images, videos, spreadsheets, and data sets – anything your project needs
- A Git repository tracks and saves the history of all changes made to the files in a Git project.
- It saves this data in a directory called .git, also known as the repository folder.

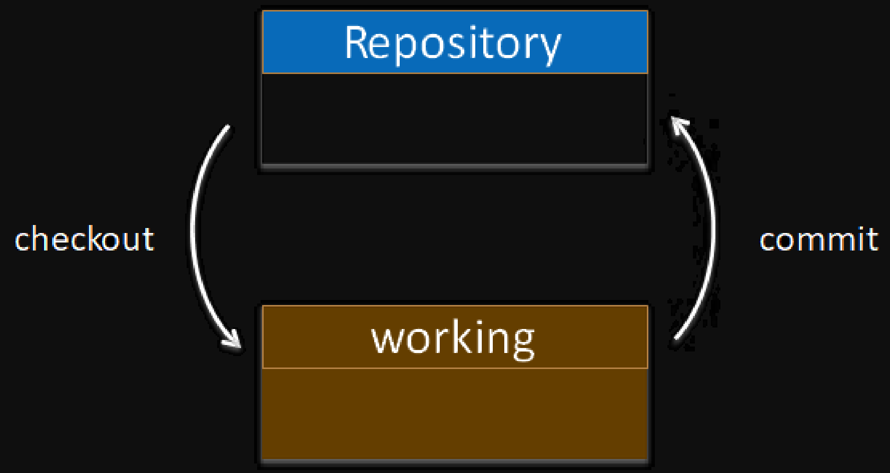



Working Area
Step 1
Adding changes to Staging Area
Staging Area
Step 2
Commit Indexed changes to Repository
Git Repository
Stores files on which changes have been made
Stores files on which add command is executed
Stores final files after the commit process

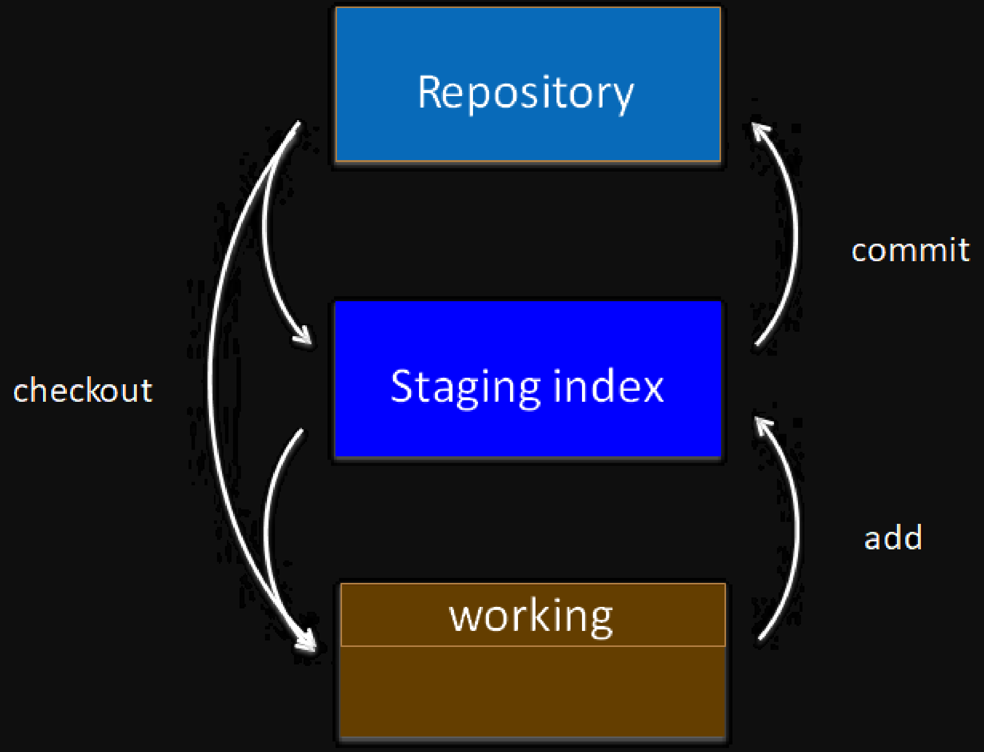
Introduction to Git

Other vs Git Architecture

- Other VCS uses Two tree architecture



- Git uses three tree architecture



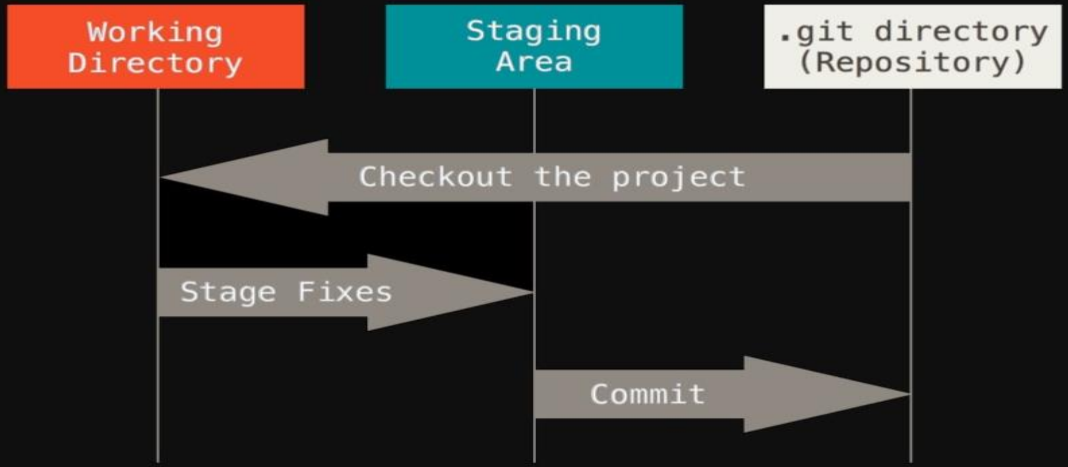
04 Git workflows

Atos

Introduction to Git

Git workflows: Creating a new repo (adding, committing code)

- The basic Git workflow goes something like this:
- You modify files in your working tree.
- You selectively stage just those changes you want to be part of your next commit, which adds only those changes to the staging area.
- You do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.



Introduction to Git

Git workflows: Creating a new repo (adding, committing code)

- Initialize a new project in a directory:

```
git init
```

- Add a file using a text editor to the directory:

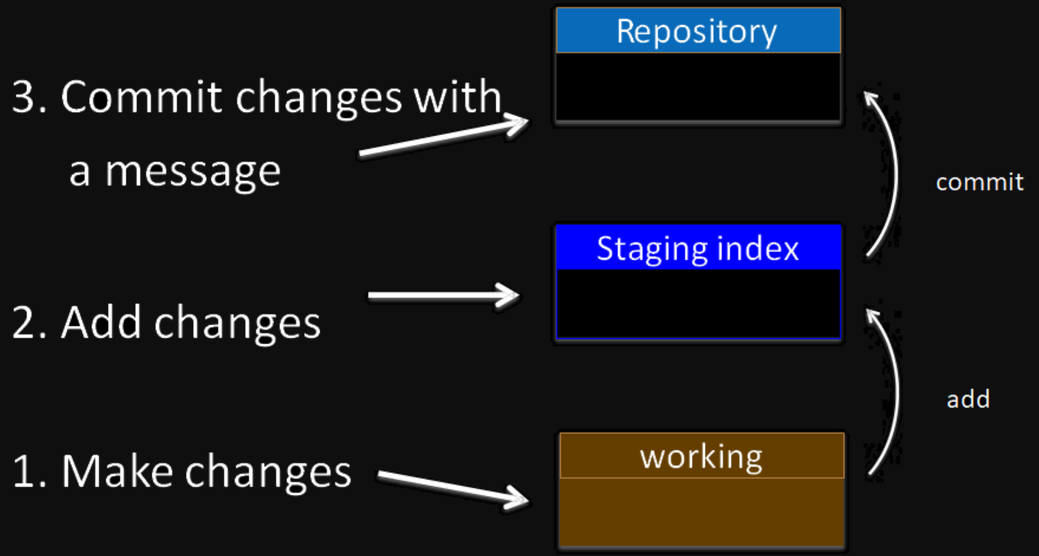
```
git add filename
```

- Add every change that has been made to the directory:

```
git add .
```

- Commit the change to the repo:

```
git commit –m "important message here"
```

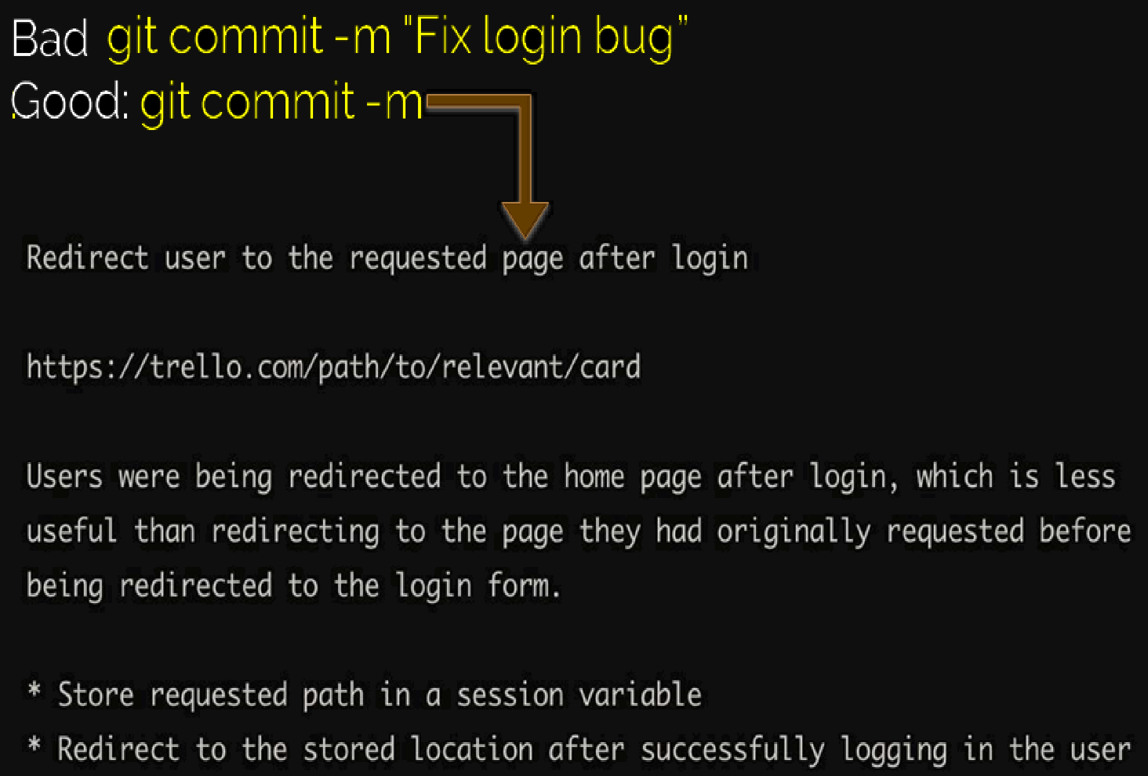


Atos

Introduction to Git

Commit

- Tell what it does (present tense)
- Single line summary followed by blank space followed by more complete description
- Keep lines to <= 72 characters
- Ticket or bug number helps
- Bad: "Typo fix"
- Good: "Add missing / in CSS section"
- Bad: "Updates the table. We'll discuss next Monday for updating."

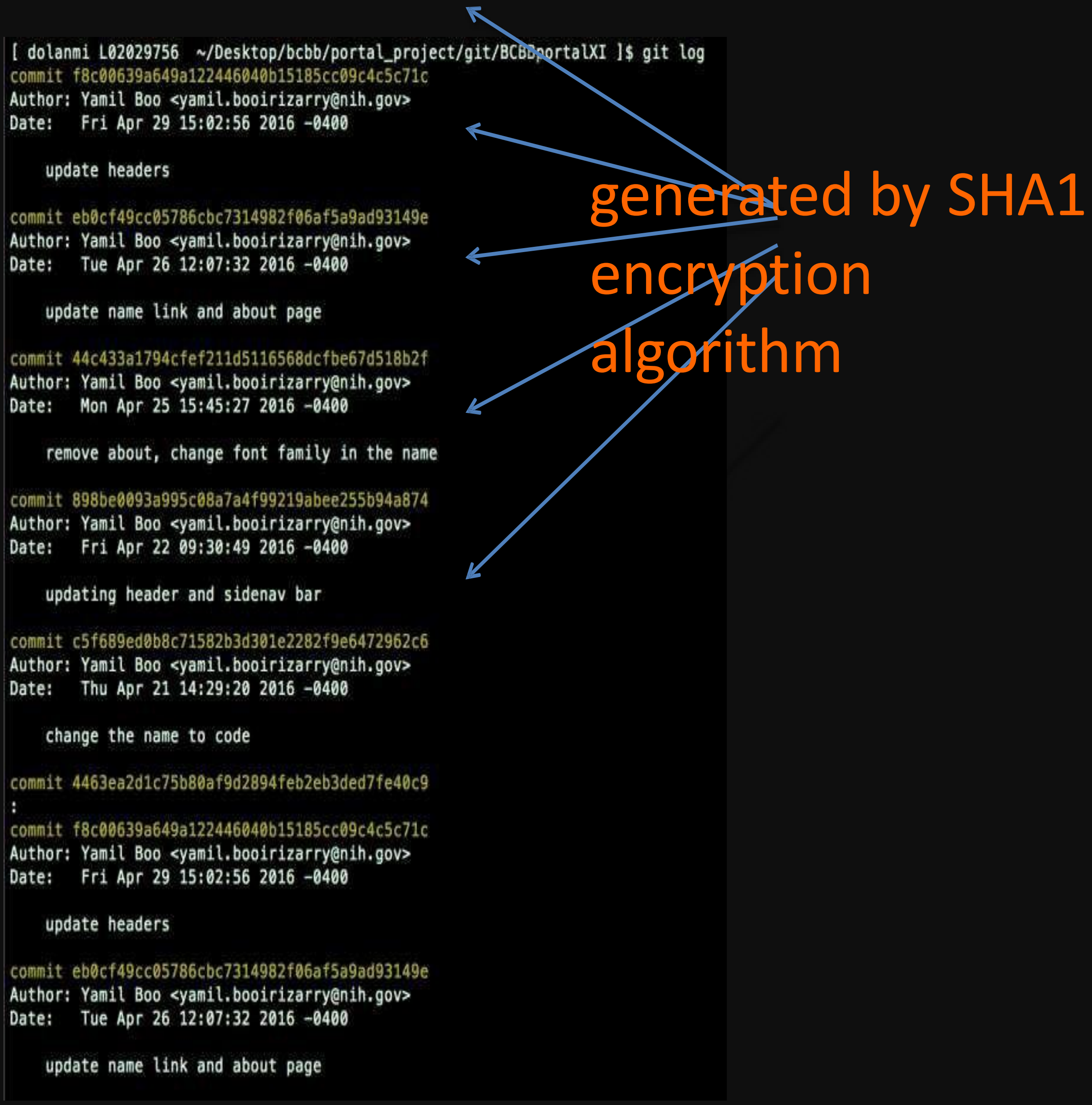


Bad git commit -m "Fix login bug"
Good: git commit -m

```
Redirect user to the requested page after login

https://trello.com/path/to/relevant/card

Users were being redirected to the home page after login, which is less
useful than redirecting to the page they had originally requested before
being redirected to the login form.

* Store requested path in a session variable
* Redirect to the stored location after successfully logging in the user
```

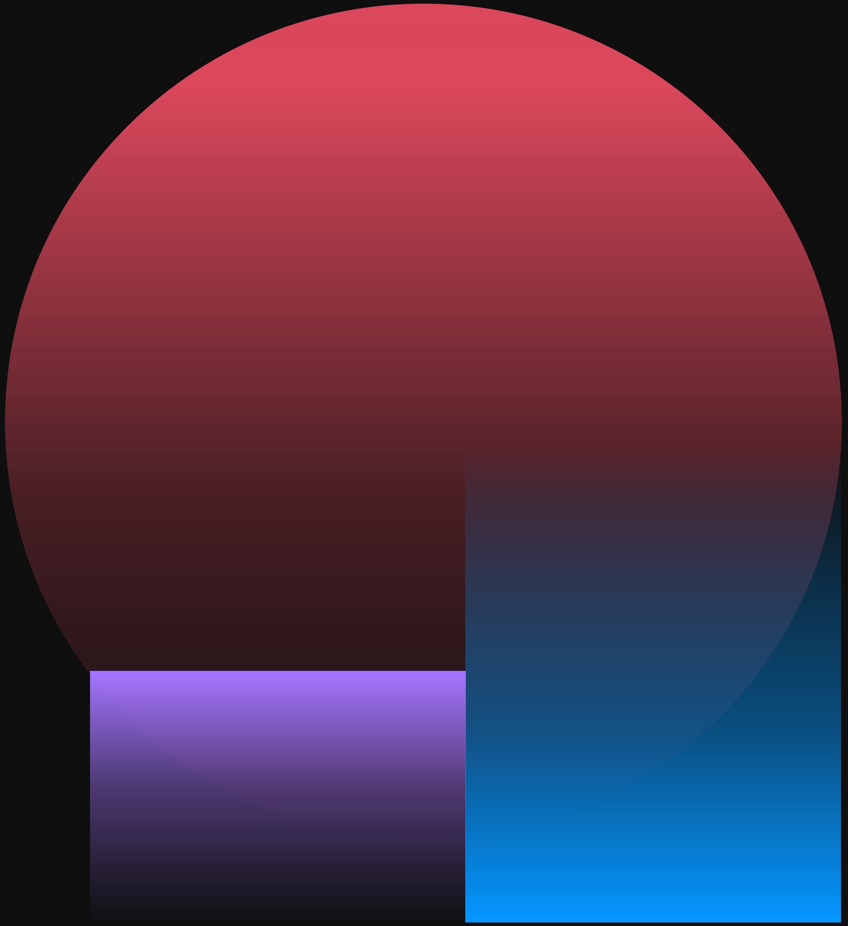
Introduction to Git
log

- Git log is a utility tool to review and read a history of everything that happens to a repository.
- Multiple options can be used with a git log to make history more specific.
- Generally, the git log is a record of commits.



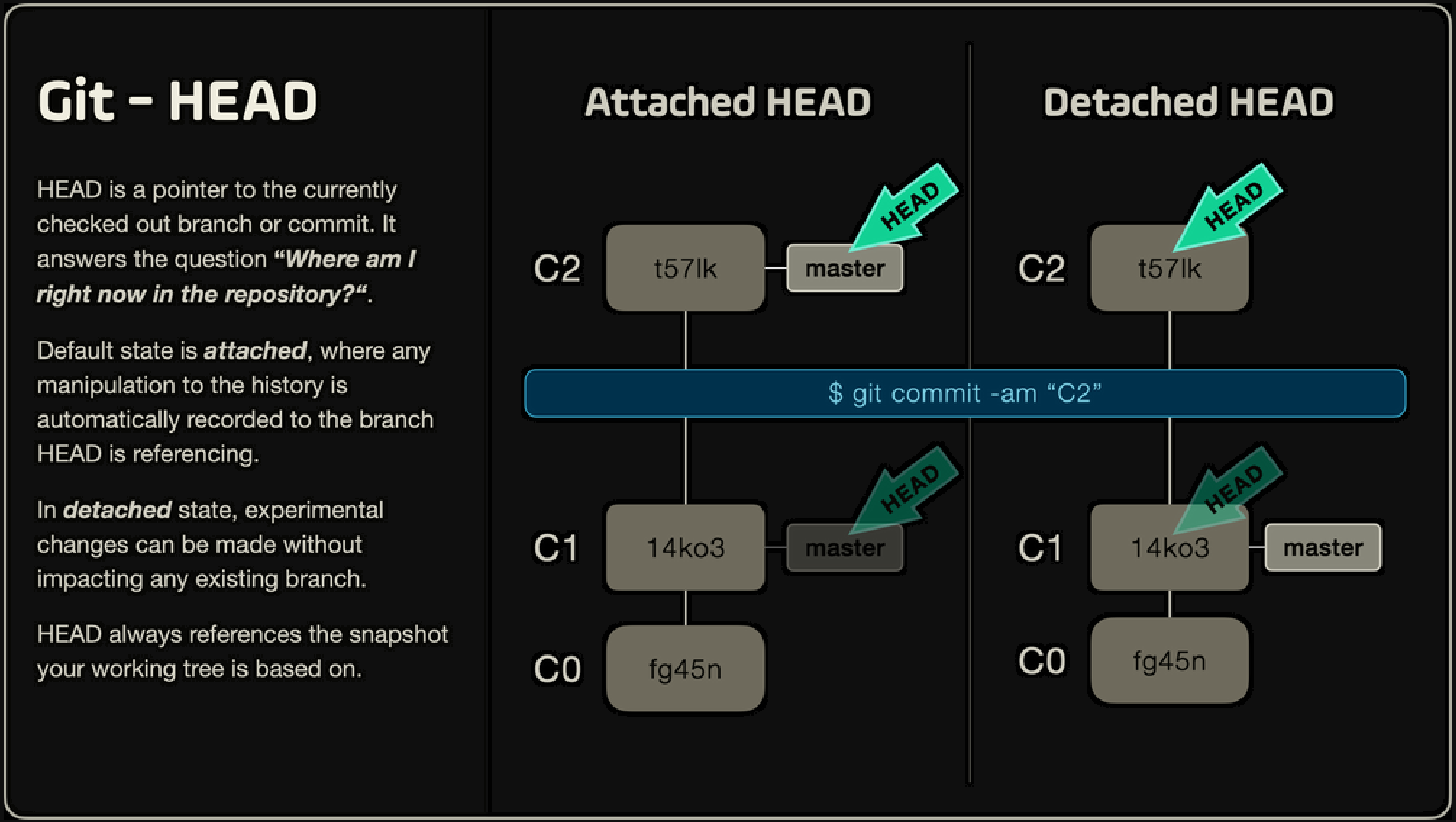
```
[ dolanmi L02029756  ~/Desktop/bcbb/portal_project/git/BCBBportalXI ]$ git log
commit f8c00639a649a122446040b15185cc09c4c5c71c
Author: Yamil Boo <yamil.booirizarry@nih.gov>
Date:   Fri Apr 29 15:02:56 2016 -0400

    update headers

commit eb0cf49cc05786cbc7314982f06af5a9ad93149e
Author: Yamil Boo <yamil.booirizarry@nih.gov>
Date:   Tue Apr 26 12:07:32 2016 -0400

    update name link and about page

commit 44c433a1794cfef211d5116568dcfbe67d518b2f
Author: Yamil Boo <yamil.booirizarry@nih.gov>
Date:   Mon Apr 25 15:45:27 2016 -0400

    remove about, change font family in the name

commit 898be0093a995c08a7a4f99219abee255b94a874
Author: Yamil Boo <yamil.booirizarry@nih.gov>
Date:   Fri Apr 22 09:30:49 2016 -0400

    updating header and sidenav bar

commit c5f689ed0b8c71582b3d301e2282f9e6472962c6
Author: Yamil Boo <yamil.booirizarry@nih.gov>
Date:   Thu Apr 21 14:29:20 2016 -0400

    change the name to code

commit 4463ea2d1c75b80af9d2894feb2eb3ded7fe40c9
:
commit f8c00639a649a122446040b15185cc09c4c5c71c
Author: Yamil Boo <yamil.booirizarry@nih.gov>
Date:   Fri Apr 29 15:02:56 2016 -0400

    update headers

commit eb0cf49cc05786cbc7314982f06af5a9ad93149e
Author: Yamil Boo <yamil.booirizarry@nih.gov>
Date:   Tue Apr 26 12:07:32 2016 -0400

    update name link and about page
```

05 HEAD



Introduction to Git
HEAD Pointer

- Points to a specific commit in repo
- As new commits are made, the pointer changes
- HEAD always points to the "tip" of the currently checked-out branch in the repo (not the working directory or staging index)
- Last state of repo (what was checked out initially)
- HEAD points to parent of next commit (where writing the next commit takes place)



Git - HEAD

HEAD is a pointer to the currently checked out branch or commit. It answers the question ***"Where am I right now in the repository?"***.

Default state is ***attached***, where any manipulation to the history is automatically recorded to the branch HEAD is referencing.

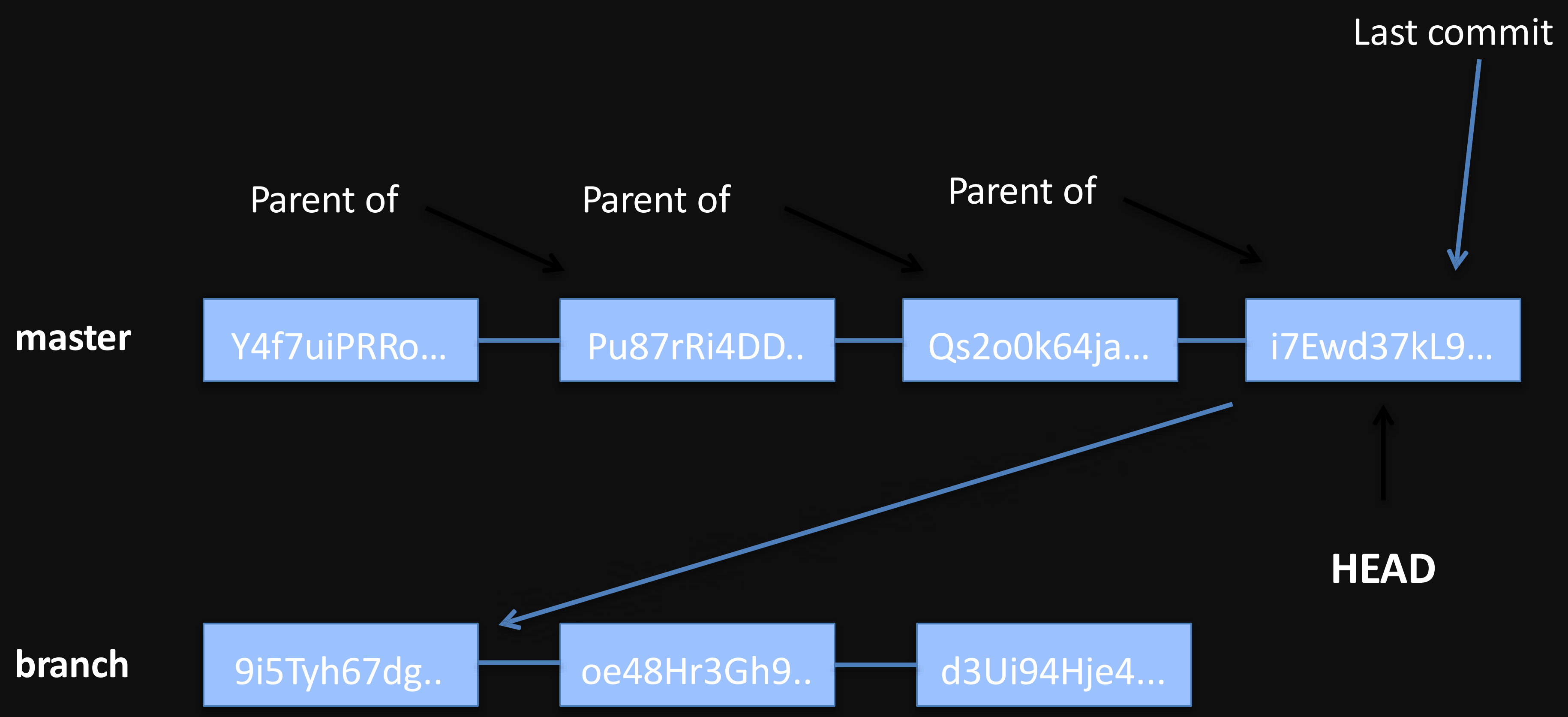
In ***detached*** state, experimental changes can be made without impacting any existing branch.

HEAD always references the snapshot your working tree is based on.

Atos

Introduction to Git

HEAD

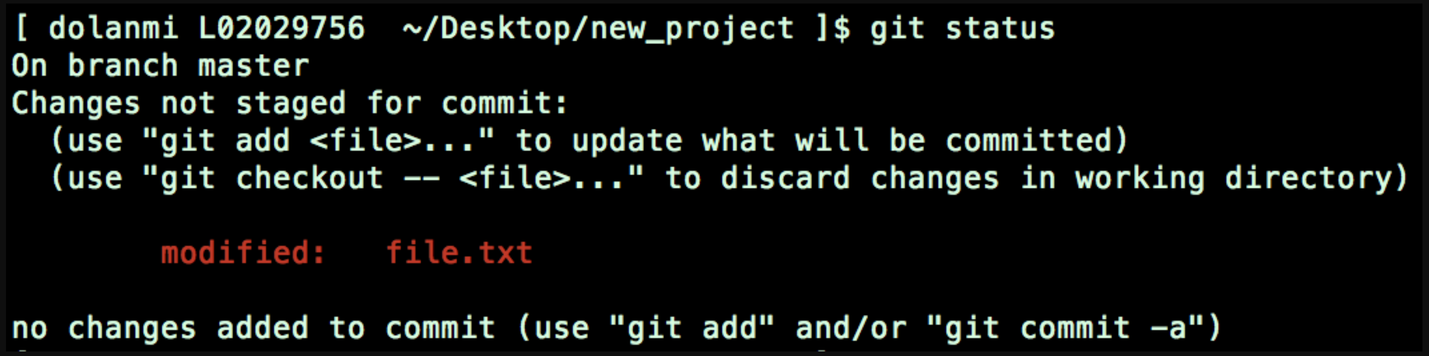


06 Git commands

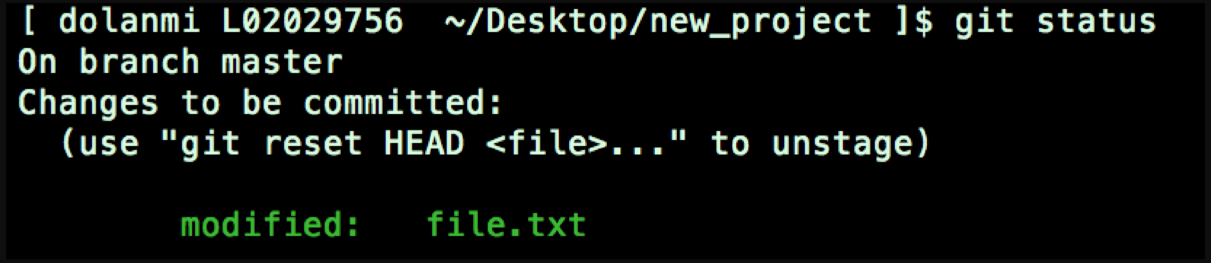
Introduction to Git

Which files were changed and where do they sit in the three tree?

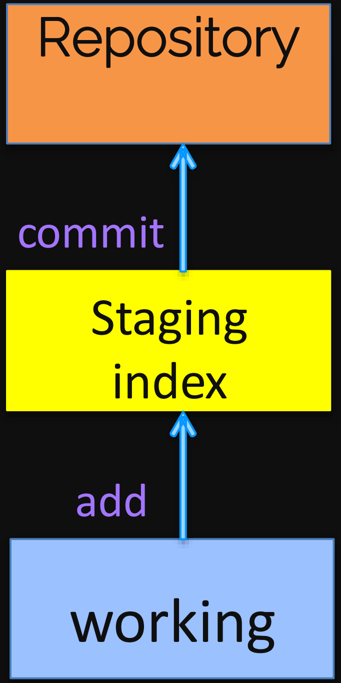
- git status – allows one to see where files are in the three-tree scheme



```
[ dolanmi L02029756  ~/Desktop/new_project ]$ git status
On branch 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:   file.txt

no changes added to commit (use "git add" and/or "git commit -a")
```



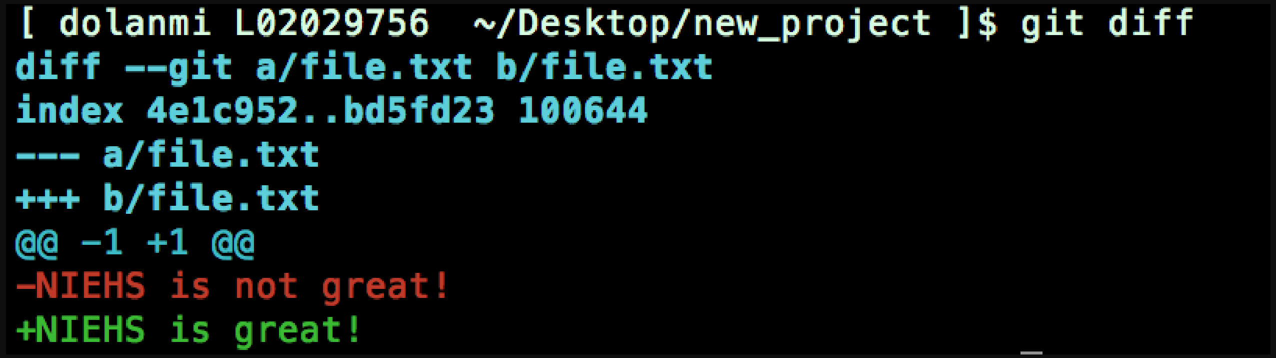
```
[ dolanmi L02029756  ~/Desktop/new_project ]$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   file.txt
```



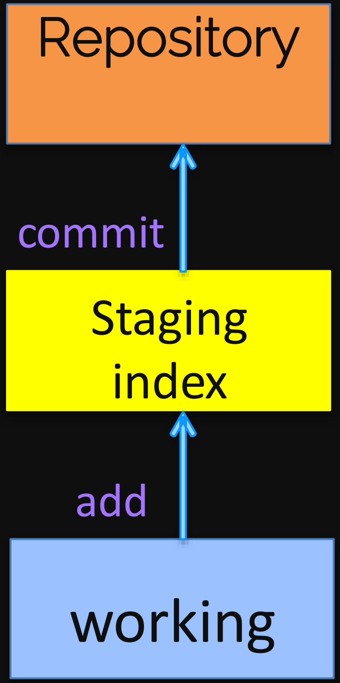
Introduction to Git

What changed in working directory?

- git diff – compares changes to files between repo and working directory
- *Note*: git diff --staged - compares staging index to repo
- *Note*: git diff filename can be used as well

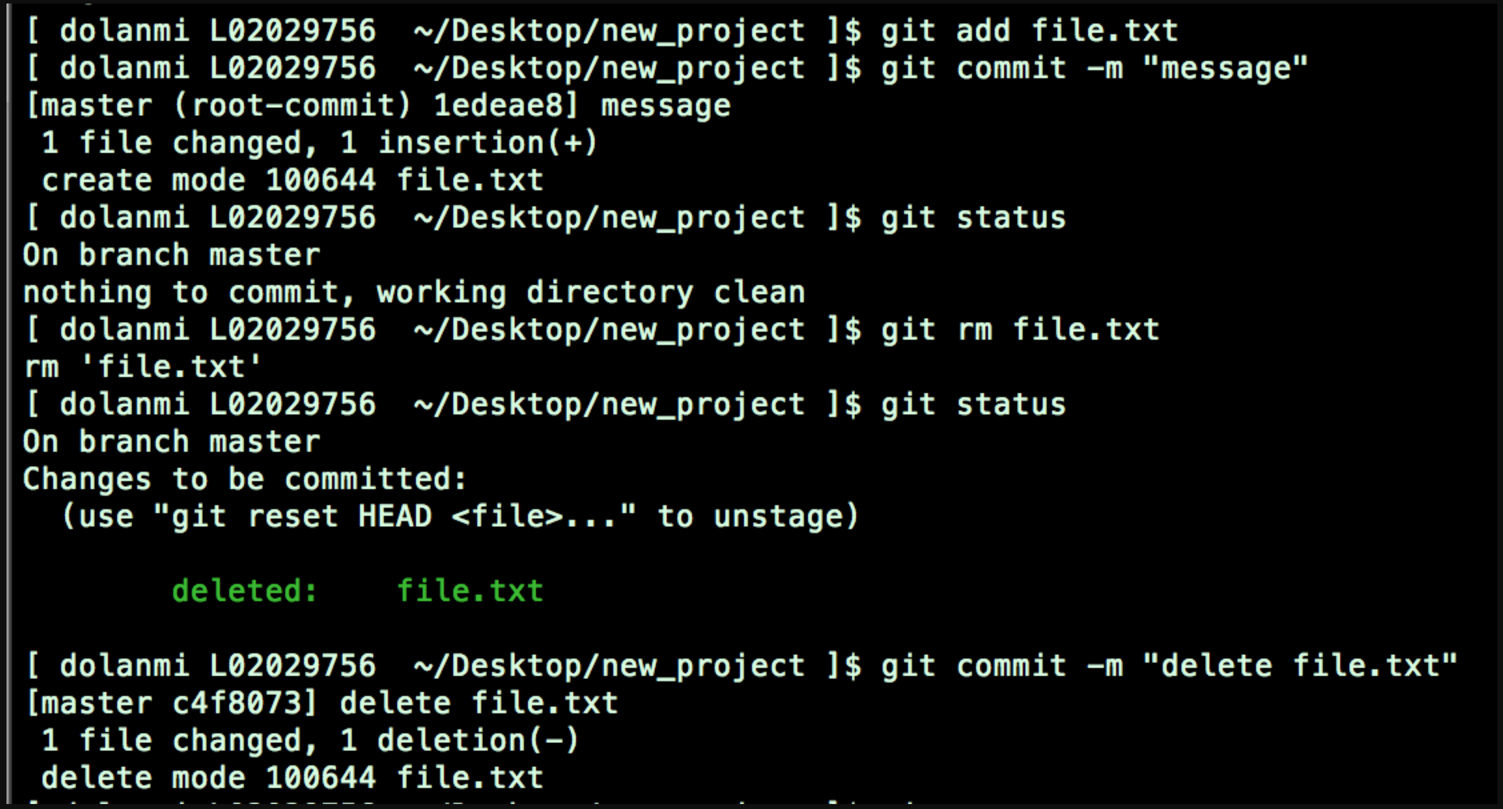


```
[ dolanmi L02029756  ~/Desktop/new_project ]$ git diff
diff --git a/file.txt b/file.txt
index 4e1c952..bd5fd23 100644
--- a/file.txt
+++ b/file.txt
@@ -1 +1 @@
-NIEHS is not great!
+NIEHS is great!
```

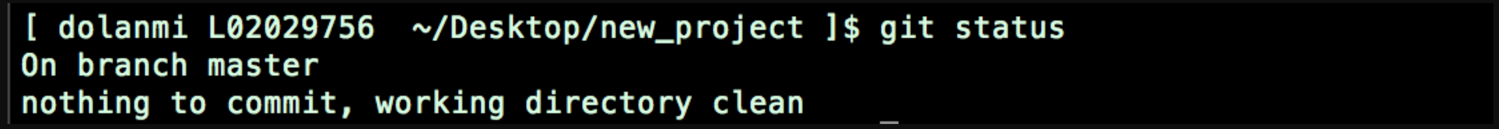


Introduction to Git

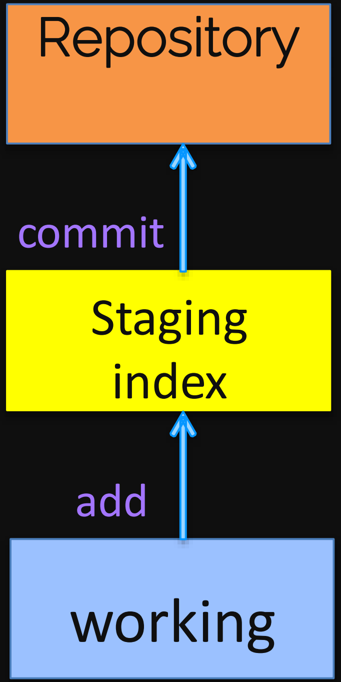
Deleting files from the repo



```
[ dolanmi L02029756  ~/Desktop/new_project ]$ git add file.txt
[ dolanmi L02029756  ~/Desktop/new_project ]$ git commit -m "message"
[master (root-commit) 1edeae8] message
 1 file changed, 1 insertion(+)
 create mode 100644 file.txt
[ dolanmi L02029756  ~/Desktop/new_project ]$ git status
On branch master
nothing to commit, working directory clean
[ dolanmi L02029756  ~/Desktop/new_project ]$ git rm file.txt
rm 'file.txt'
[ dolanmi L02029756  ~/Desktop/new_project ]$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        deleted:    file.txt

[ dolanmi L02029756  ~/Desktop/new_project ]$ git commit -m "delete file.txt"
[master c4f8073] delete file.txt
 1 file changed, 1 deletion(-)
 delete mode 100644 file.txt
```



```
[ dolanmi L02029756  ~/Desktop/new_project ]$ git status
On branch master
nothing to commit, working directory clean
```



Introduction to Git
Commands

```
git init
git status
git log
git add
git commit
git diff
git rm
git mv
```

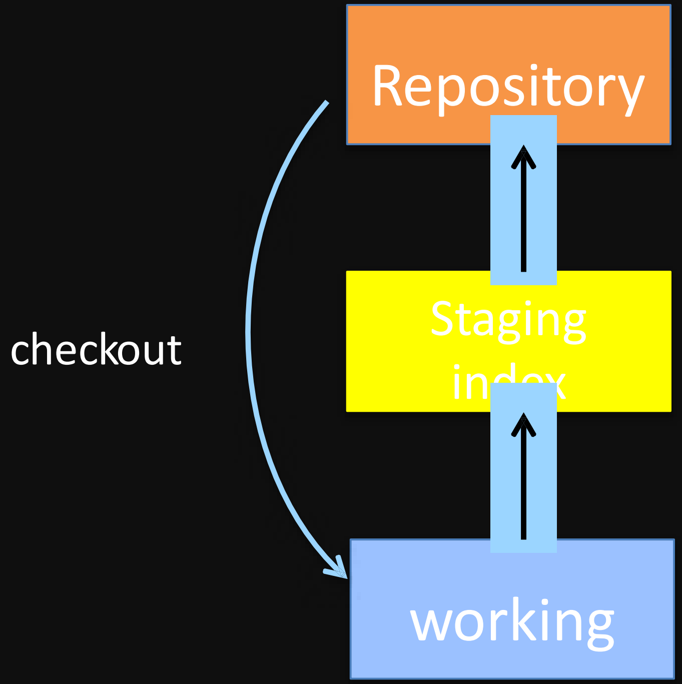
75% of the time you'll be using only these commands

Atos

Introduction to Git

What if I want to undo changes made to working directory?

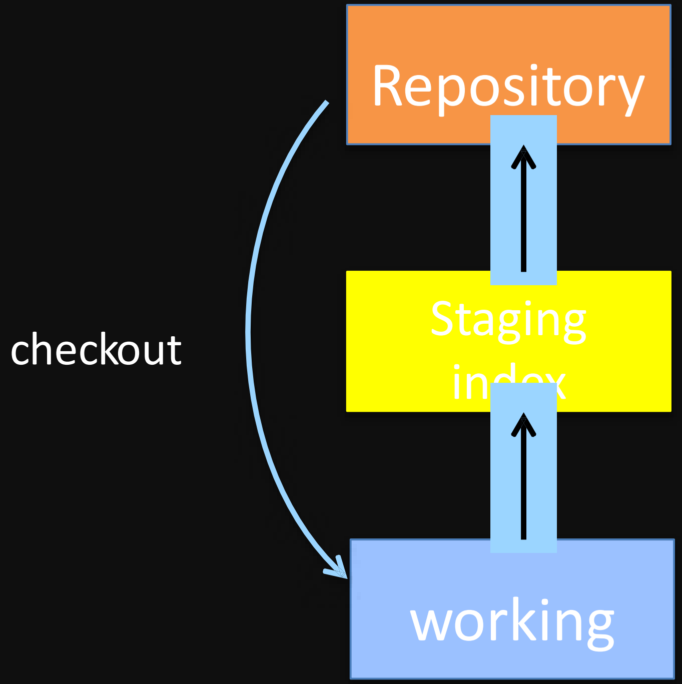
- git checkout *something*
- (where “something” is a file or an entire branch)
- git checkout will grab the file from the repo
- *Example:* git checkout -- file1.txt
- (“checkout file ‘file1.txt’ from the current branch”)



Introduction to Git

What if I want to undo changes made to working directory?

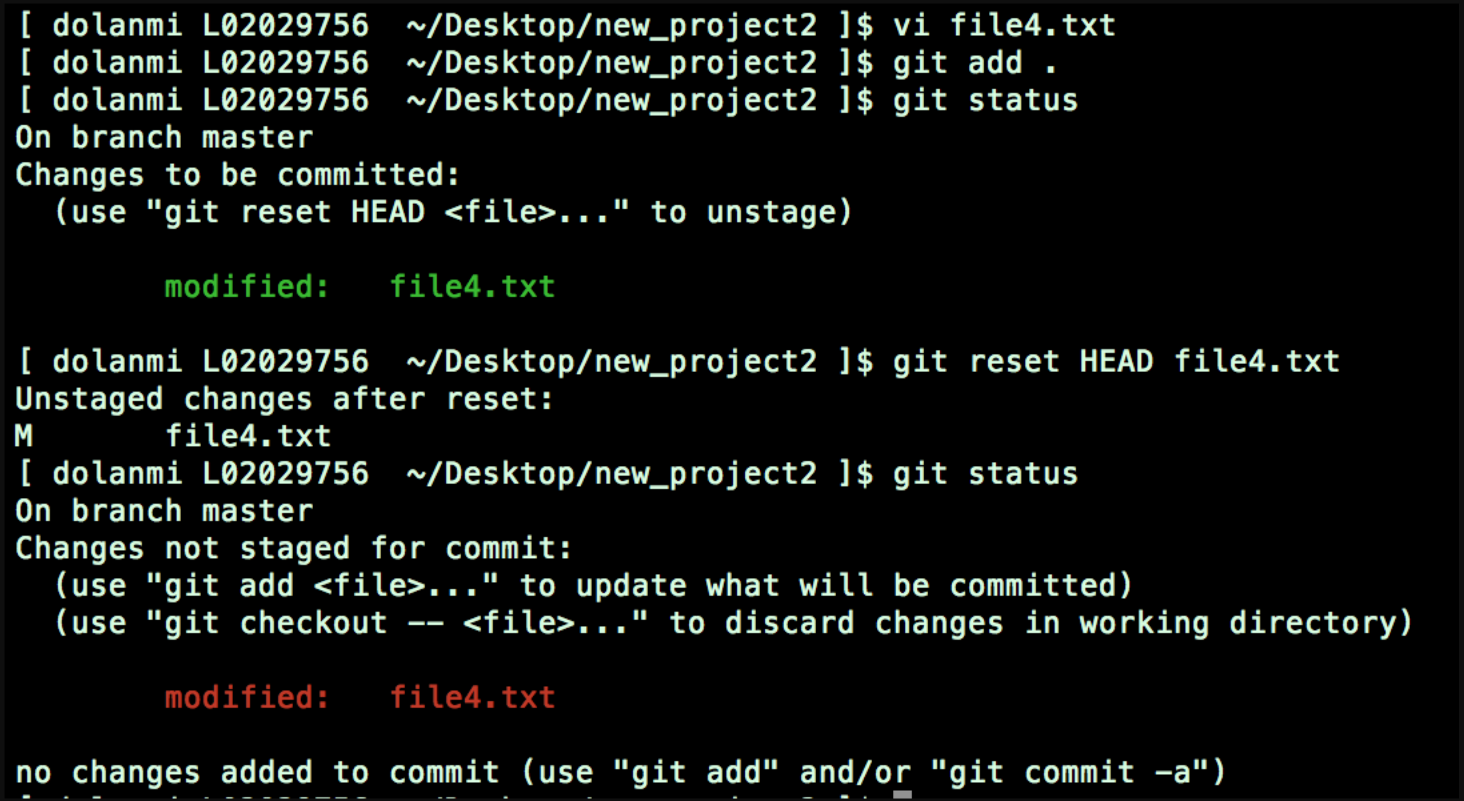
- git checkout *something*
- (where "something" is a file or an entire branch)
- git checkout will grab the file from the repo
- *Example:* git checkout -- file1.txt
- ("checkout file 'file1.txt' from the current branch")



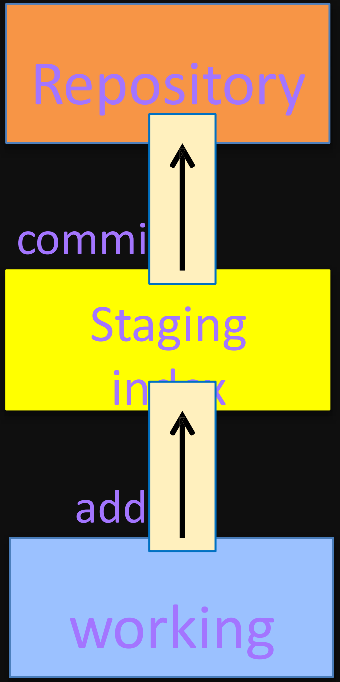
Introduction to Git

What if I want to undo changes added to staging area?

- git reset HEAD filename.txt



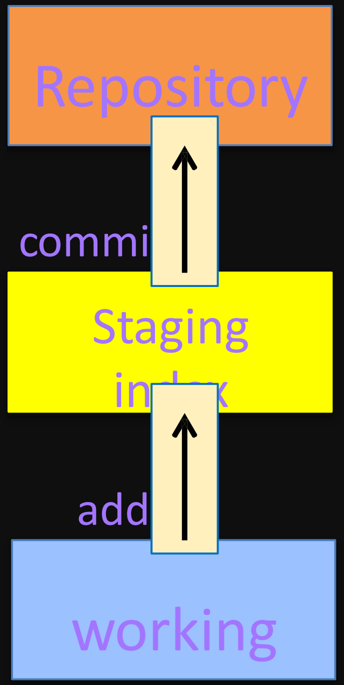
```
[ dolanmi L02029756  ~/Desktop/new_project2 ]$ vi file4.txt
[ dolanmi L02029756  ~/Desktop/new_project2 ]$ git add .
[ dolanmi L02029756  ~/Desktop/new_project2 ]$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   file4.txt

[ dolanmi L02029756  ~/Desktop/new_project2 ]$ git reset HEAD file4.txt
Unstaged changes after reset:
M       file4.txt
[ dolanmi L02029756  ~/Desktop/new_project2 ]$ git status
On branch 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:   file4.txt

no changes added to commit (use "git add" and/or "git commit -a")
```



Introduction to Git

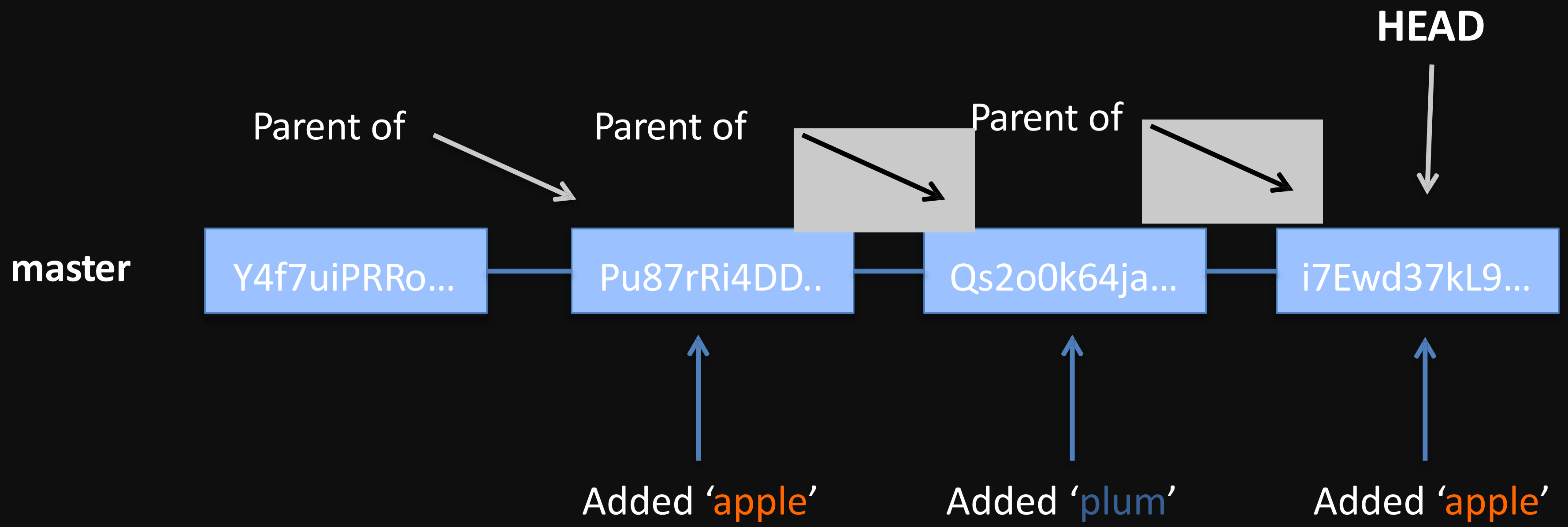
What if I want to undo changes committed to the repo?

- git commit --amend -m “message”
- allows one to amend a change to the last commit
- anything in staging area will be amended to the last commit



Introduction to Git

To undo changes to older commits, make a new commit

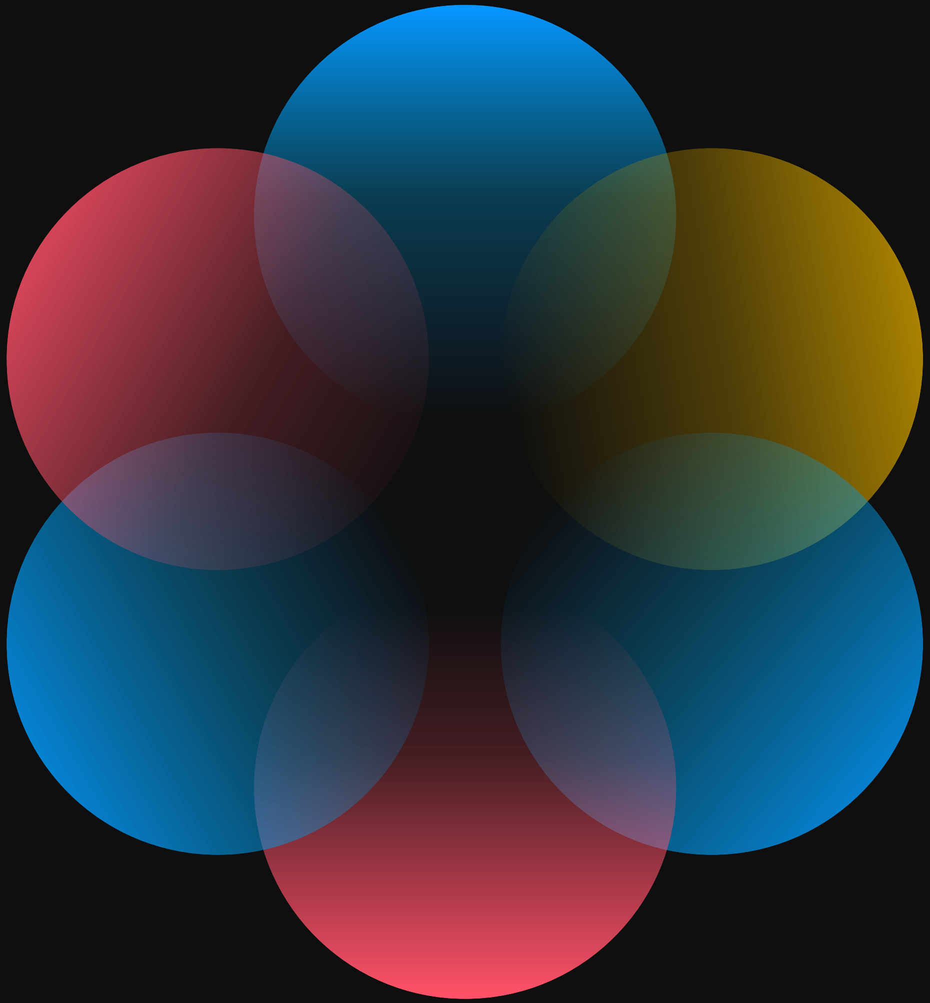


Introduction to Git
Obtain older versions

- git checkout 6e073c640928b -- filename.txt
- Note: Checking out older commit's places them into the staging area
- git checkout 6e073c640928b -- filename.txt

- git ls-tree tree-ish
- **tree-ish** – a way to reference a repo
  - full SHA, part SHA, HEAD, others

06 Master vs branch concept

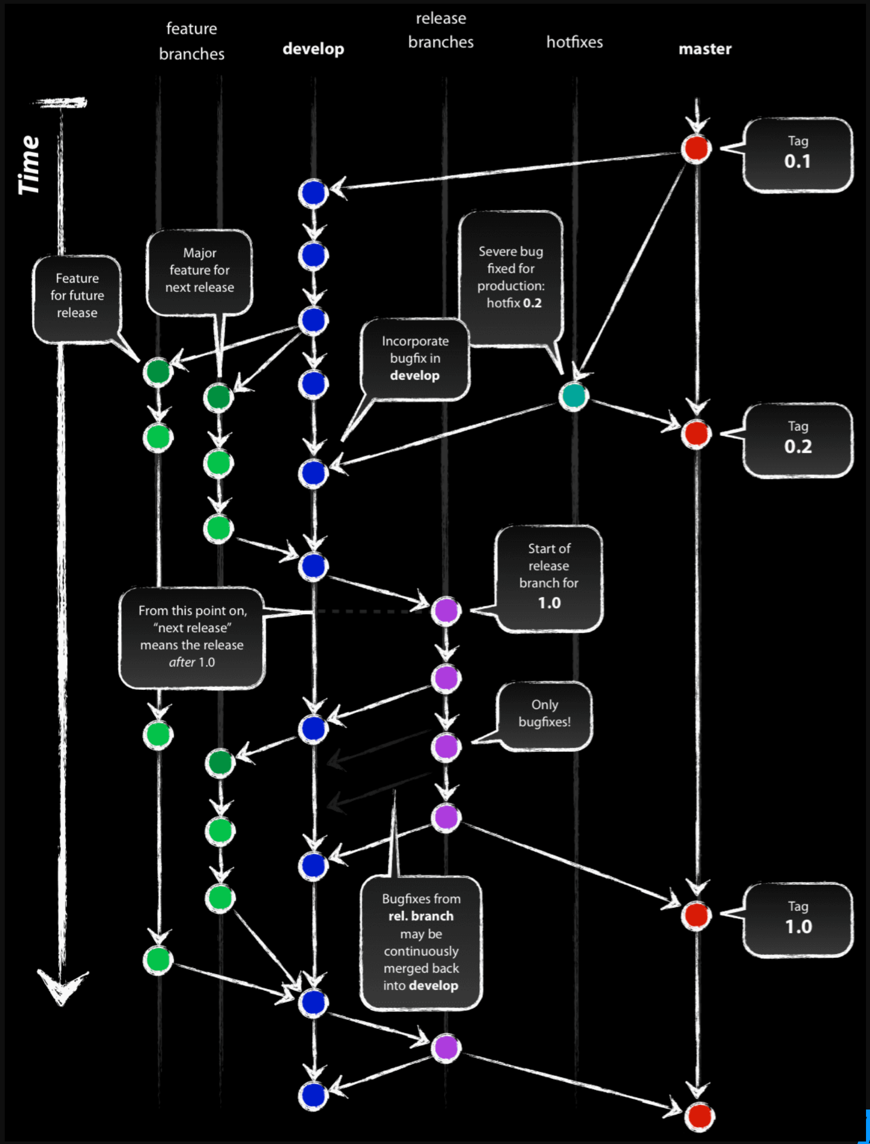


Atos

Git Master vs Branch

Branching

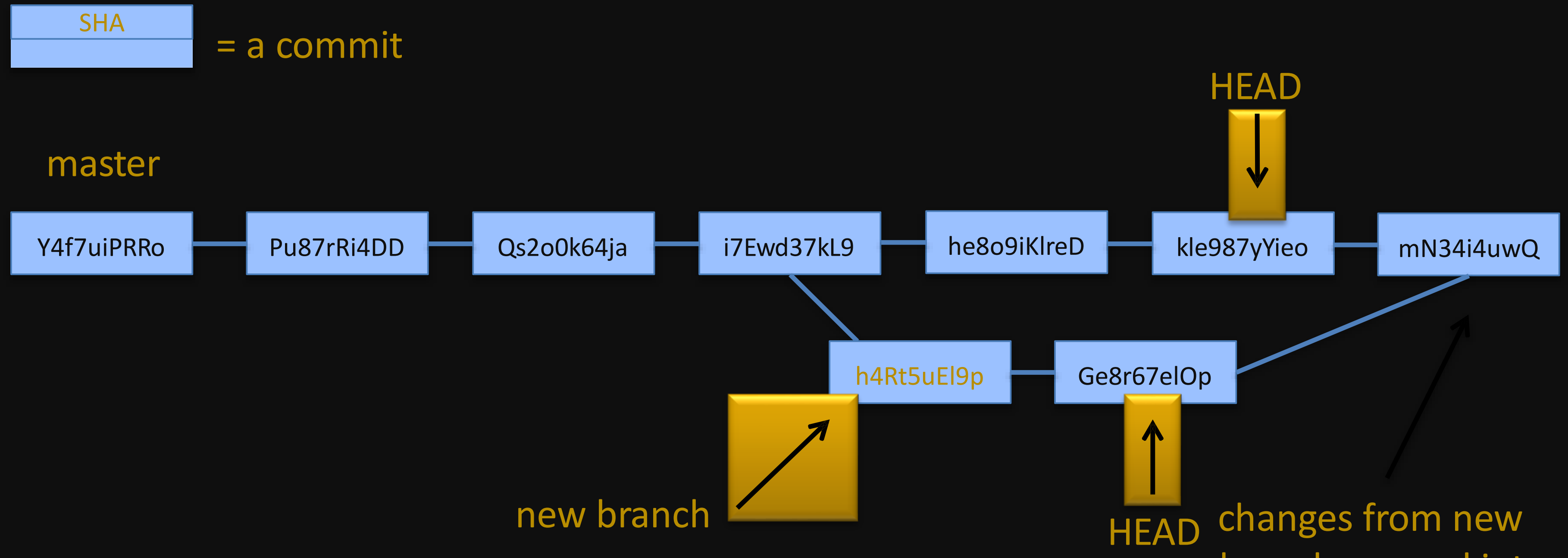
- Allows one to try new ideas
- If an idea doesn’t work, throw away the branch.
- Don’t have to undo many changes to master branch
- If it does work, merge ideas into master branch.
- There is only one working directory



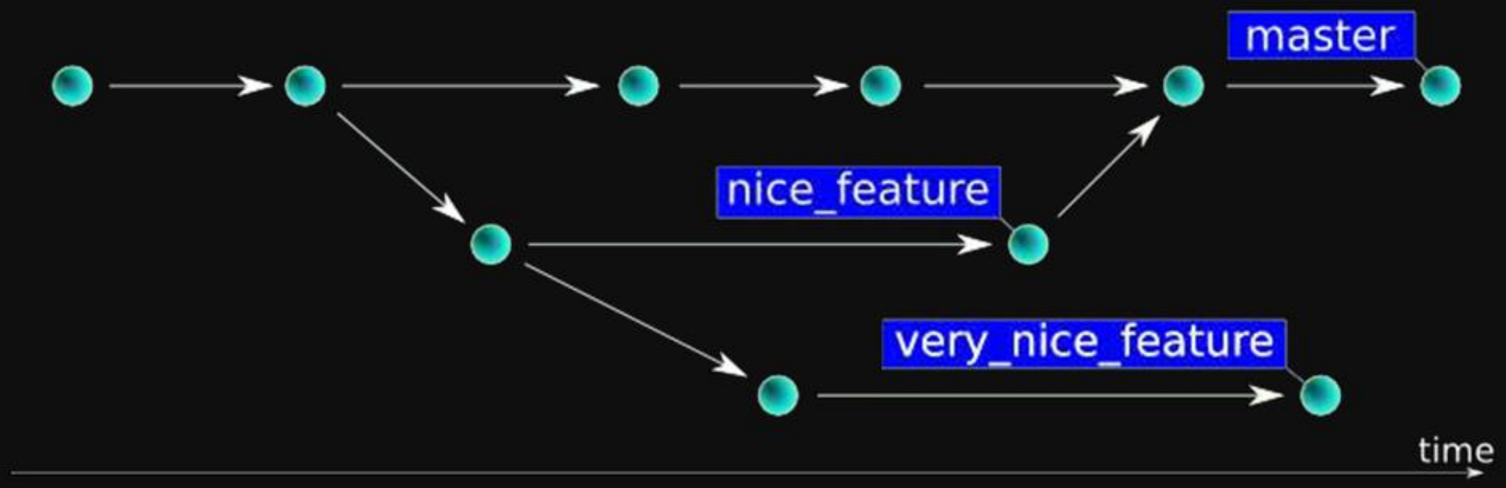
feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

Git Master vs Branch

Branching and merging example



changes from new branch merged into master



07 Creating a branch/switching between branches

Git Master vs Branch

Branch commands

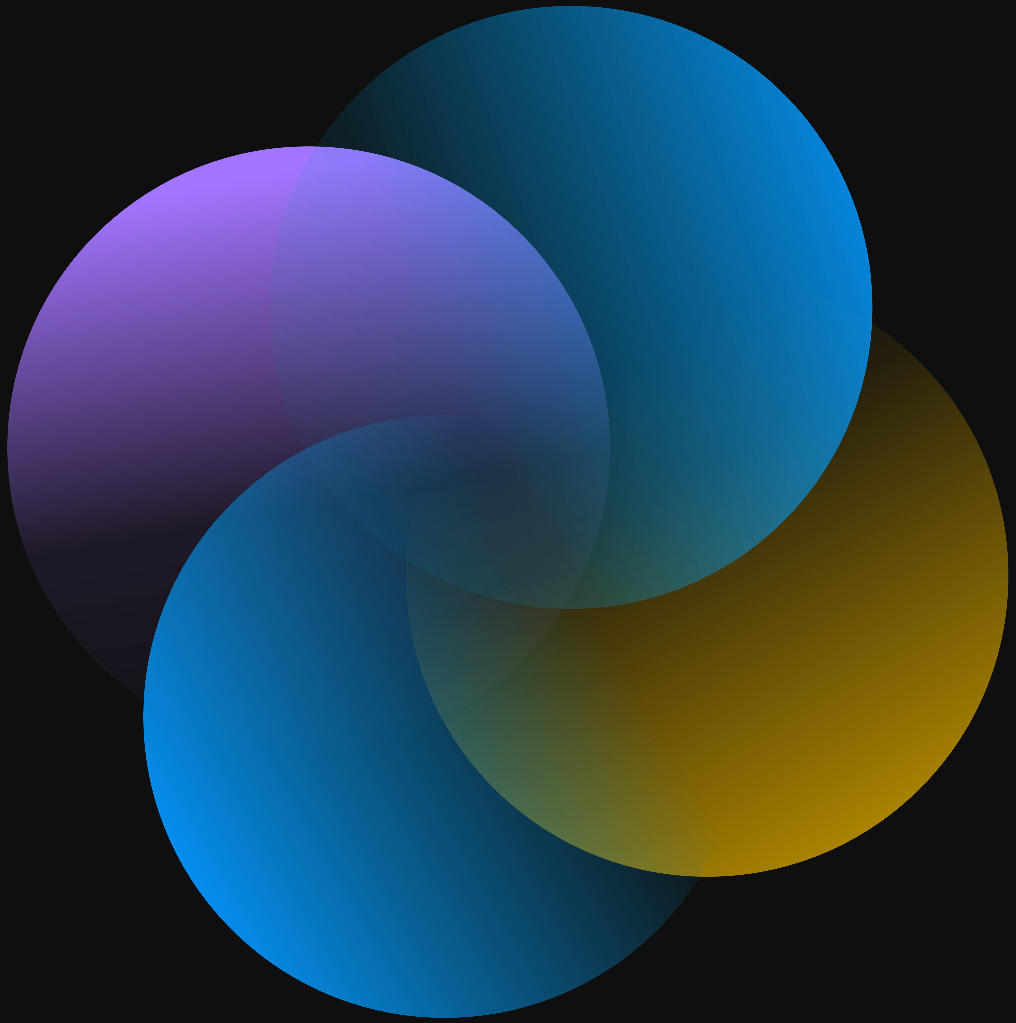
- git branch
- Create a new branch: git branch branchname
- Creating a local branch and switching to it:git checkout -b <new-branch-name>
- git checkout -b <new-branch-name> <from-branch-name>

- Pushing a local branch to remote:git push -u origin <branch-name>

- whenever your team members need to reach your branch, they can run the git fetch command as follows:
- git fetch
- git checkout <branch-name>

- git checkout new_branch_name
- git diff first_branch..second_branch

08 Merging branches and resolving conflicts



Atos

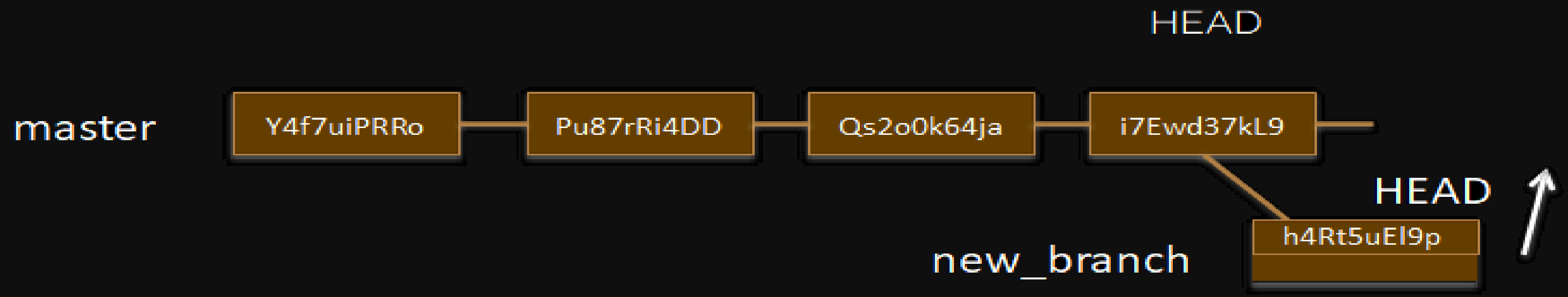
Git Master vs Branch

Merge

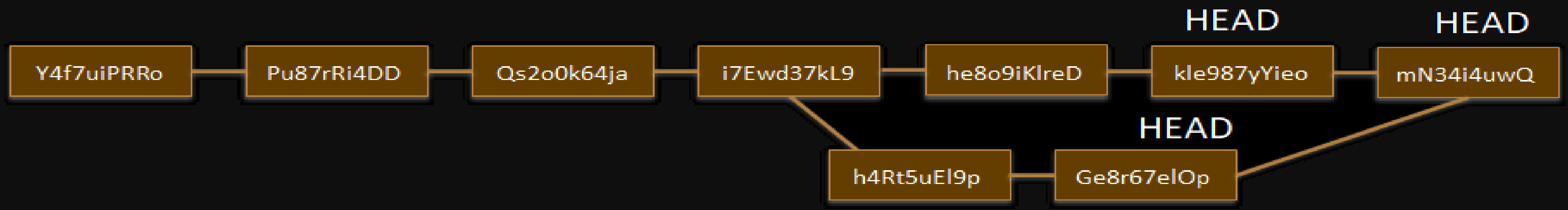
- From the branch into which you want to merge another branch
- git merge branch_to_merge

"**fast-forward**" merge occurs when HEAD of master

branch is seen when looking back



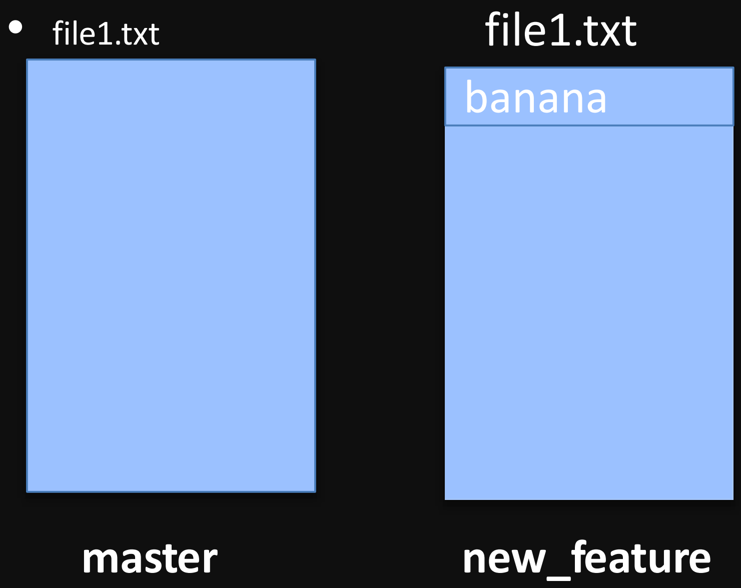
"**recursive**" merge occurs by looking back and combining ancestors to resolve merge

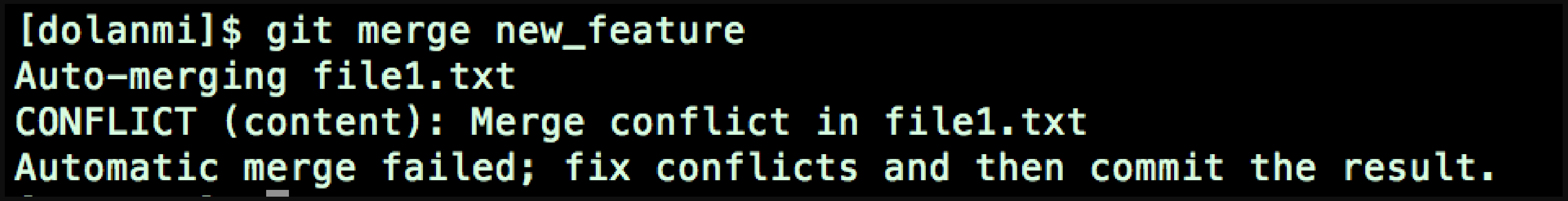


Git Merge

Merge Conflicts

- What if there are two changes to same line in two
- different commits?



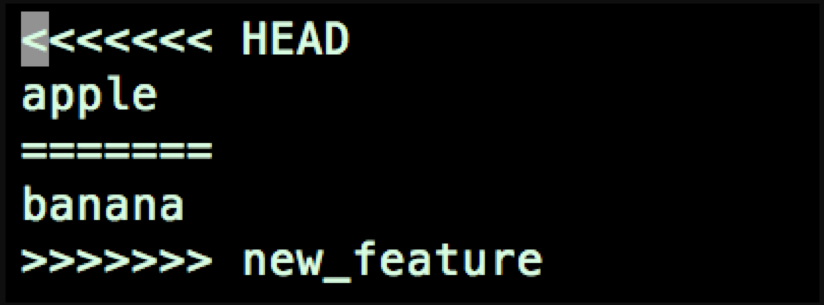


```
[dolanmi]$ git merge new_feature
Auto-merging file1.txt
CONFLICT (content): Merge conflict in file1.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Git Merge

Resolving merge conflicts

- Git will notate the conflict in the files!



```
<<<<<<< HEAD
apple
=======
banana
>>>>>>> new_feature
```

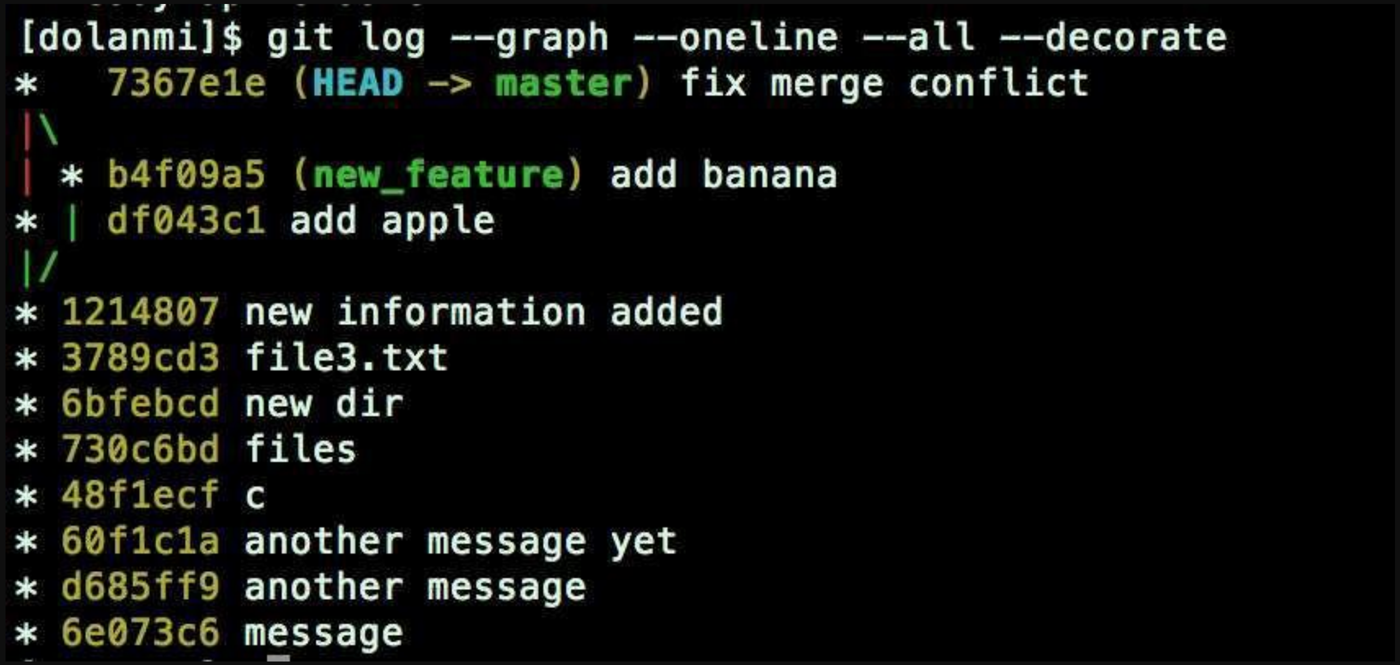
Solutions:

1. Abort the merge using git merge –abort
2. Manually fix the conflict
3. Use a merge tool (there are many out there)

Git merge

Graphing merge history

- git log --graph --oneline --all –decorate
- merge often, keep commits small/focused
- bring changes occurring to master into your branch frequently (“tracking”)



```
[dolanmi]$ git log --graph --oneline --all --decorate
*   7367e1e (HEAD -> master) fix merge conflict
|\
| * b4f09a5 (new_feature) add banana
* | df043c1 add apple
|/
* 1214807 new information added
* 3789cd3 file3.txt
* 6bfebcd new dir
* 730c6bd files
* 48f1ecf c
* 60f1c1a another message yet
* d685ff9 another message
* 6e073c6 message
```

09 What is GitHub?

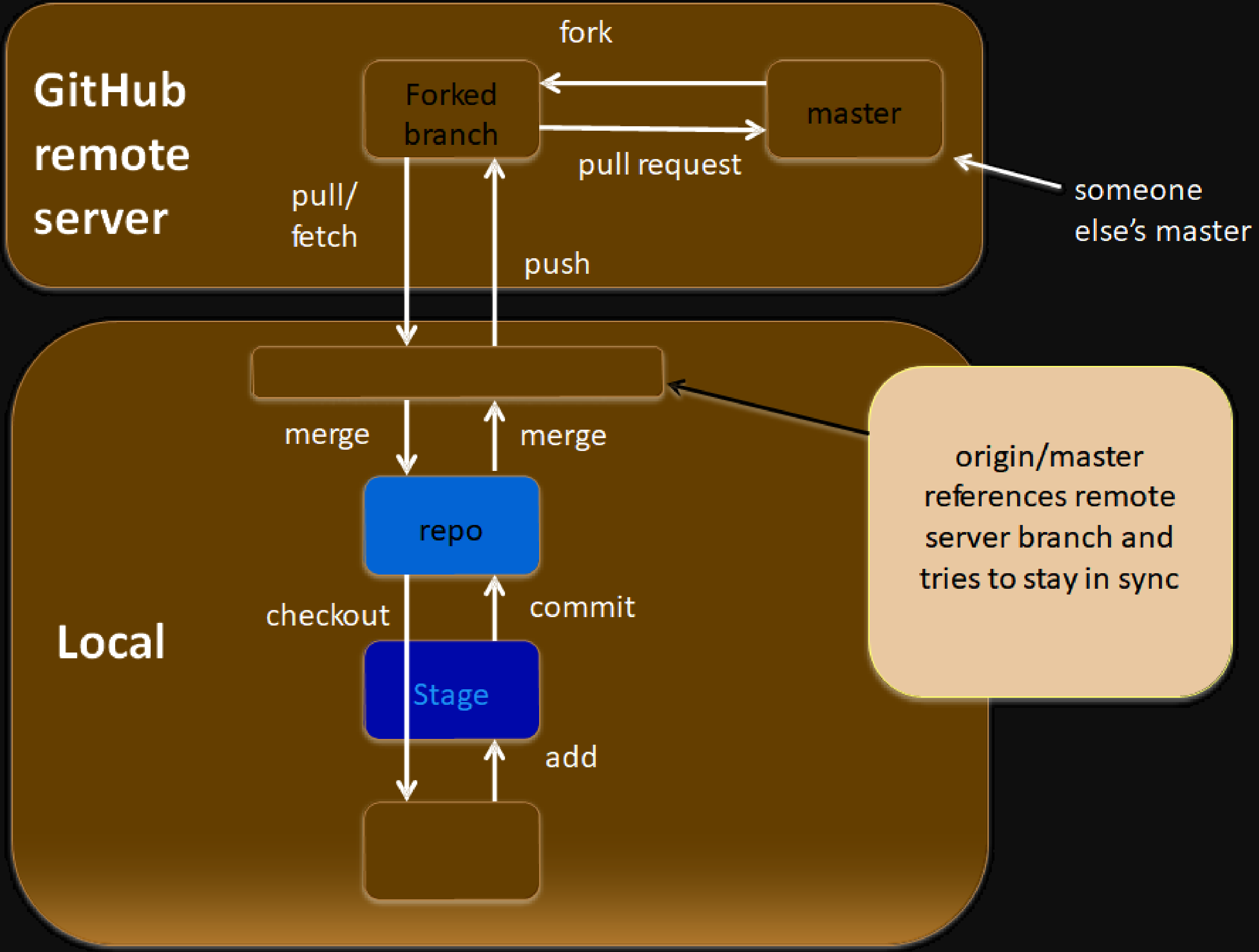
Introduction to GitHub

What is GitHub? Basic GitHub concepts

- A platform to host git code repositories
- http://github.com
- Launched in 2008
- Most popular Git host
- Allows users to collaborate on projects from anywhere
- GitHub makes git social!
- Free to start

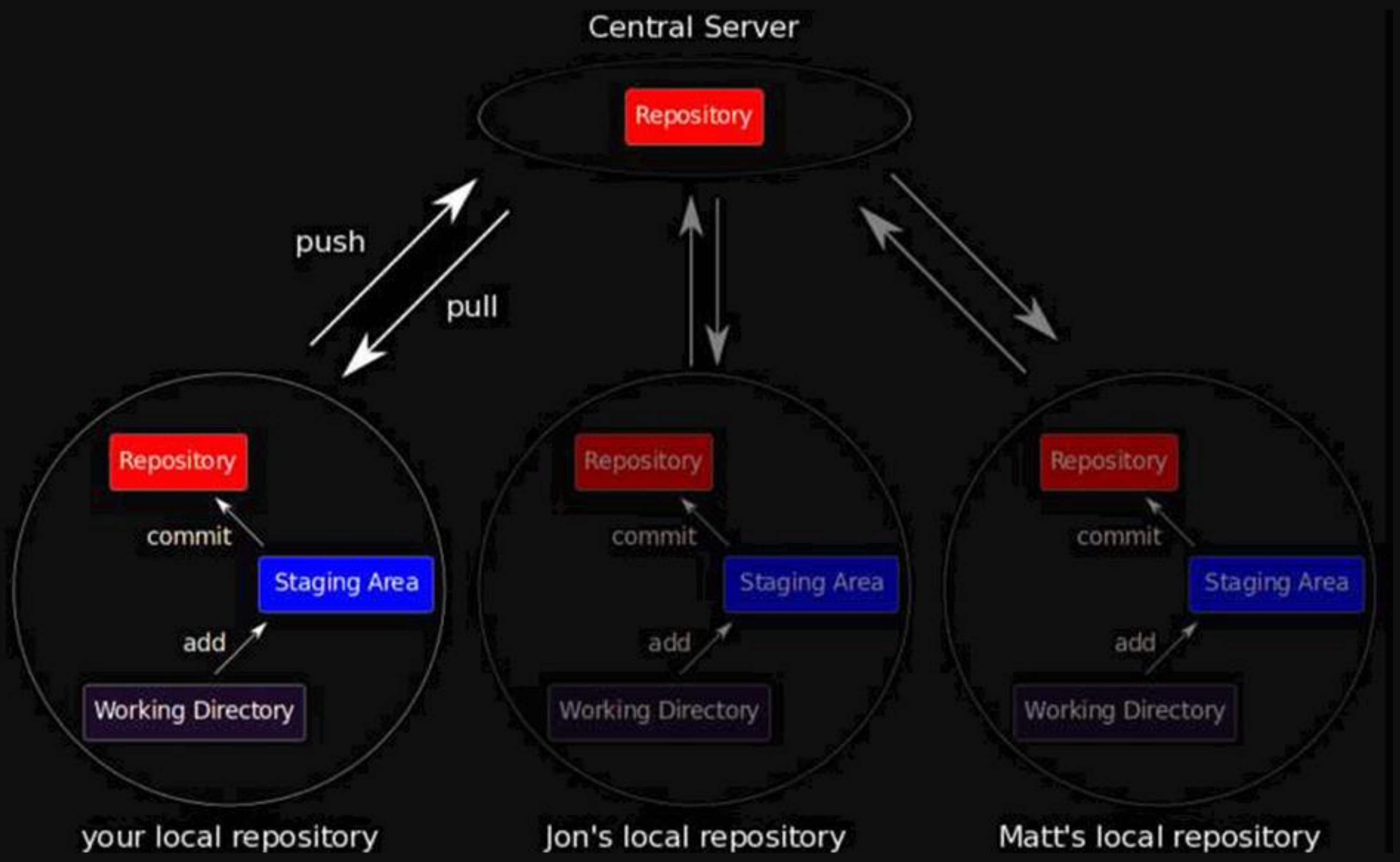
Introduction to GitHub

What is GitHub? Basic GitHub concepts



Introduction to GitHub

What is GitHub? Basic GitHub concepts

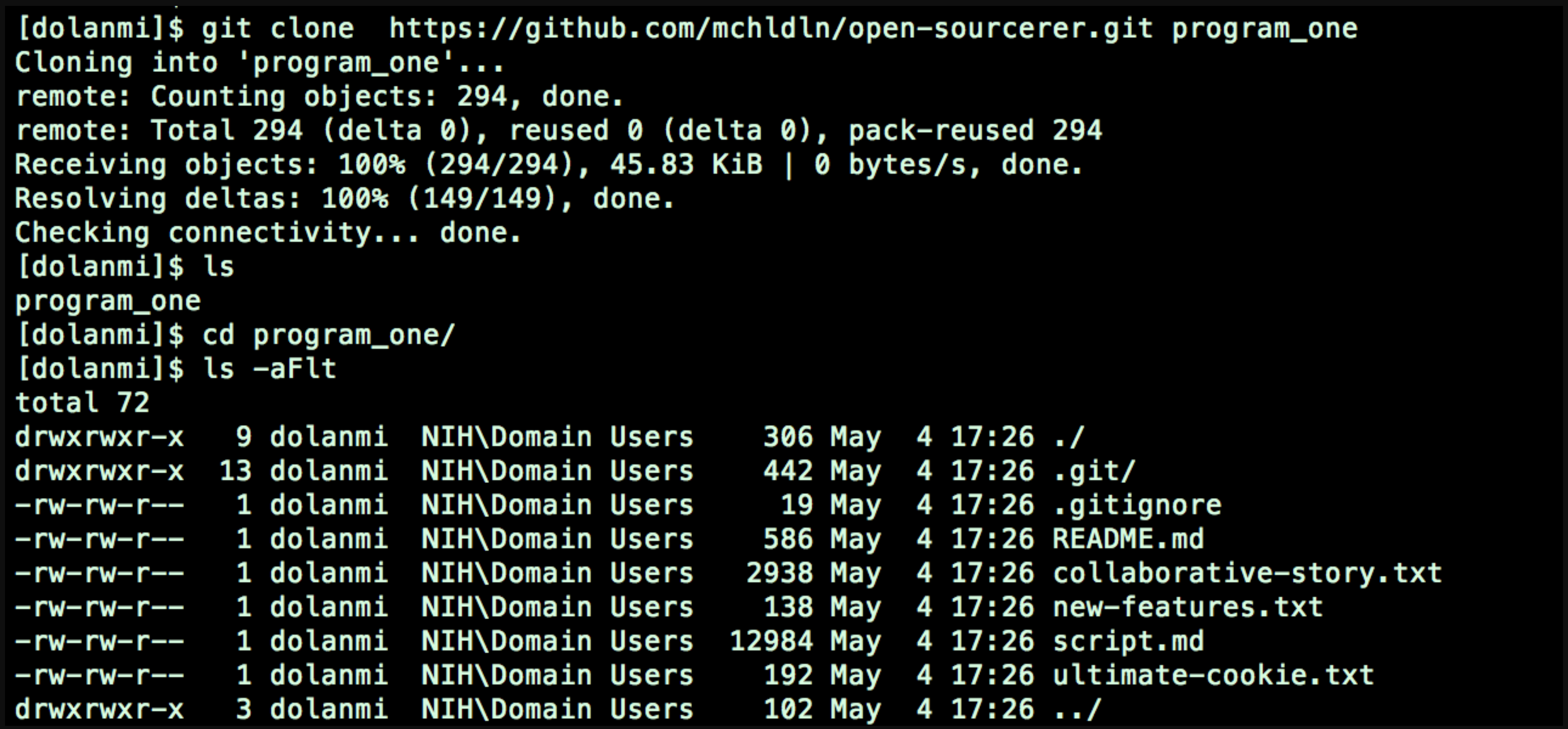


10 Cloning a remote repo

Atos

GitHub
Cloning a remote repo

- git clone URL <new_dir_name>

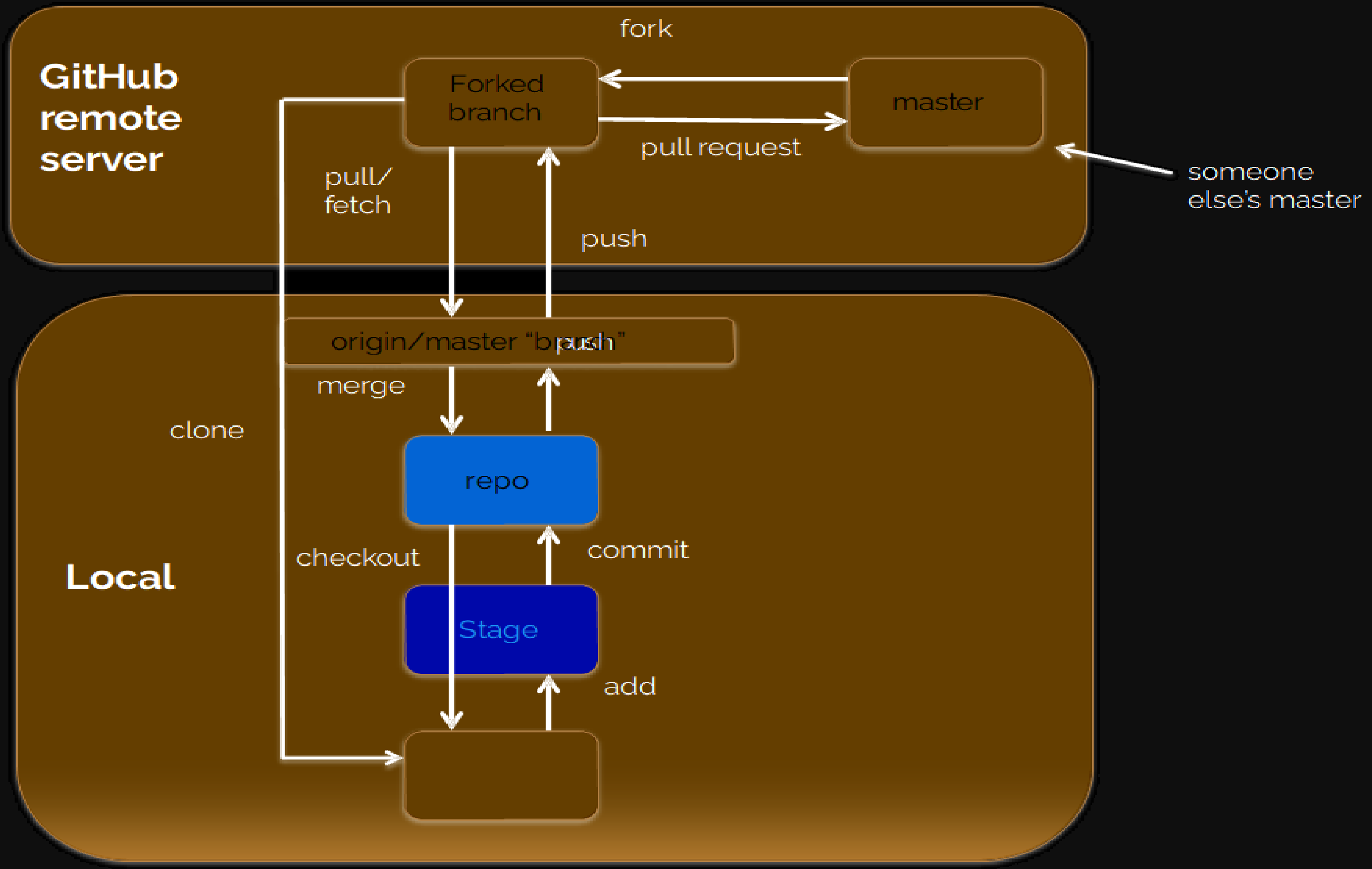


```
[dolanmi]$ git clone  https://github.com/mchldln/open-sourcerer.git program_one
Cloning into 'program_one'...
remote: Counting objects: 294, done.
remote: Total 294 (delta 0), reused 0 (delta 0), pack-reused 294
Receiving objects: 100% (294/294), 45.83 KiB | 0 bytes/s, done.
Resolving deltas: 100% (149/149), done.
Checking connectivity... done.
[dolanmi]$ ls
program_one
[dolanmi]$ cd program_one/
[dolanmi]$ ls -aFlt
total 72
drwxrwxr-x   9 dolanmi  NIH\Domain Users    306 May  4 17:26 ./
drwxrwxr-x  13 dolanmi  NIH\Domain Users    442 May  4 17:26 .git/
-rw-rw-r--   1 dolanmi  NIH\Domain Users     19 May  4 17:26 .gitignore
-rw-rw-r--   1 dolanmi  NIH\Domain Users    586 May  4 17:26 README.md
-rw-rw-r--   1 dolanmi  NIH\Domain Users   2938 May  4 17:26 collaborative-story.txt
-rw-rw-r--   1 dolanmi  NIH\Domain Users    138 May  4 17:26 new-features.txt
-rw-rw-r--   1 dolanmi  NIH\Domain Users  12984 May  4 17:26 script.md
-rw-rw-r--   1 dolanmi  NIH\Domain Users    192 May  4 17:26 ultimate-cookie.txt
drwxrwxr-x   3 dolanmi  NIH\Domain Users    102 May  4 17:26 ../
```

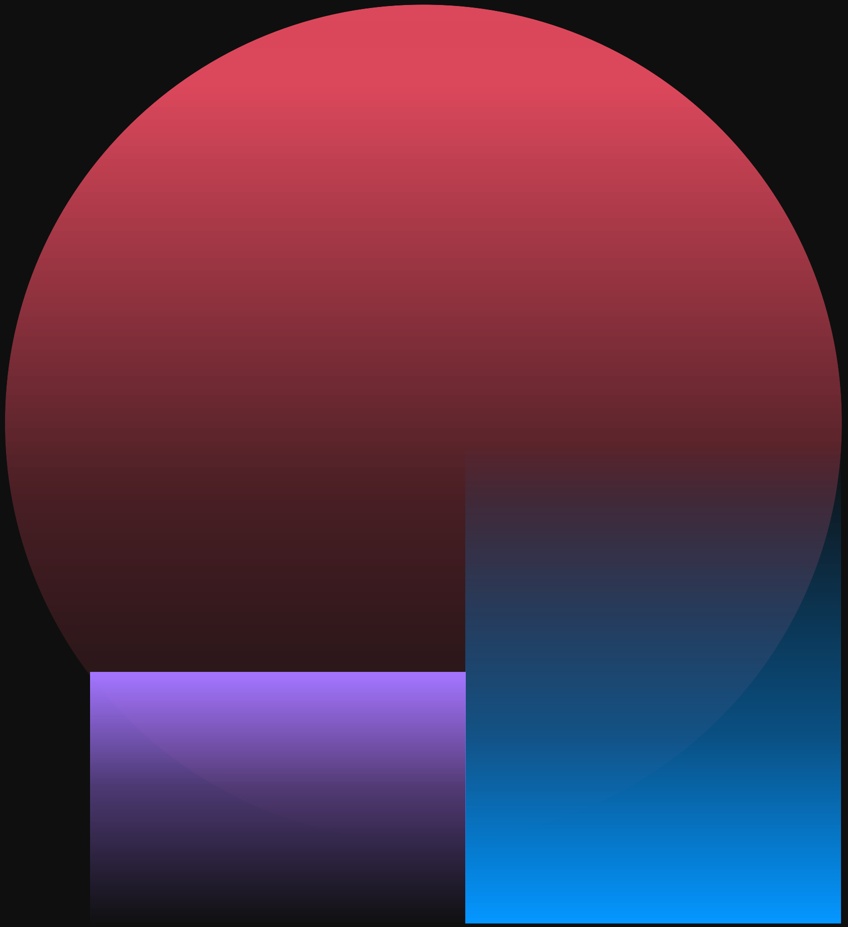
Atos

GitHub

Cloning



11 Fetching/Pushing to a remote repo

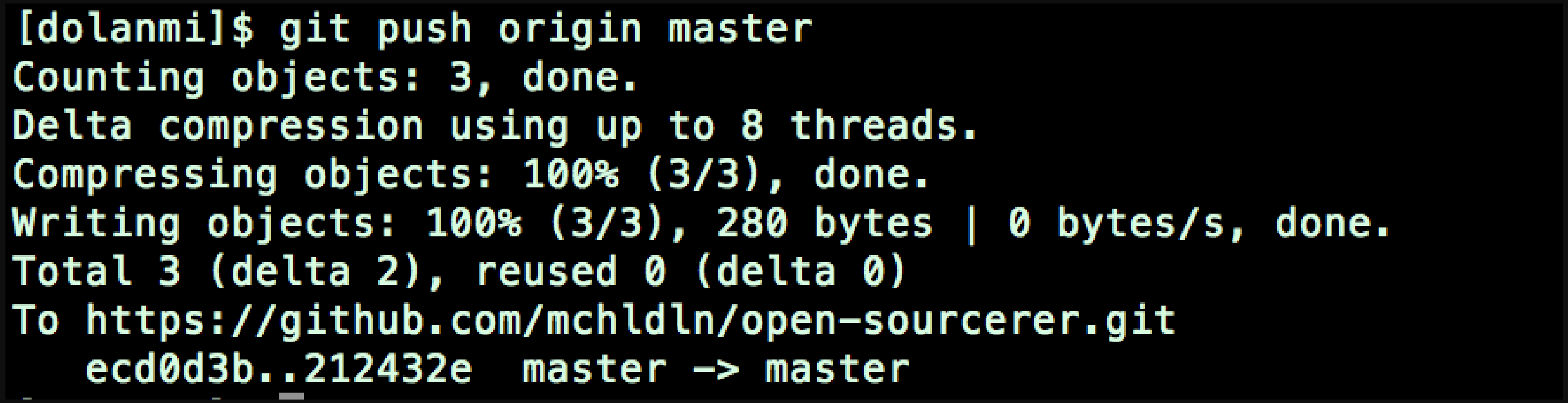


Atos

GitHub

How do I link my local repo to a remote repo?

- git push local_branch_alias branch_name



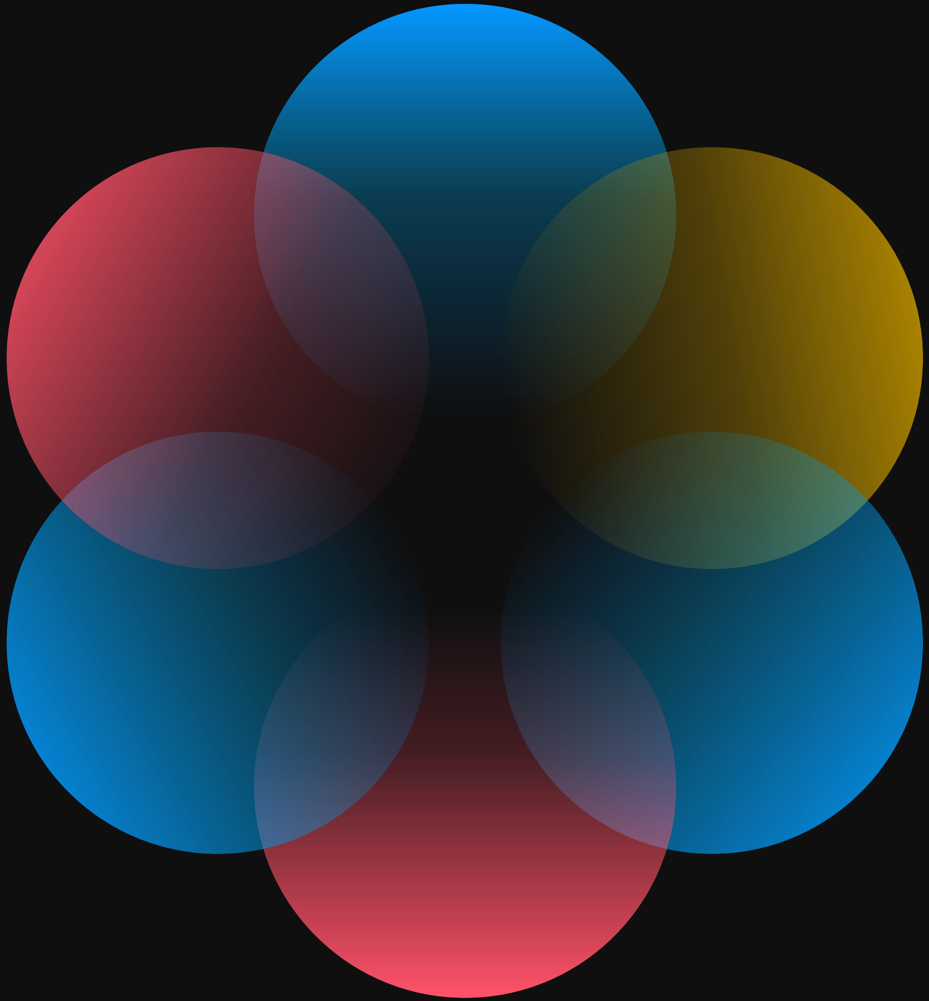
```
[dolanmi]$ git push origin master
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 280 bytes | 0 bytes/s, done.
Total 3 (delta 2), reused 0 (delta 0)
To https://github.com/mchldln/open-sourcerer.git
   ecd0d3b..212432e  master -> master
```

GitHub

Fetching from remote repo

- git fetch remote_repo_name
- **Fetch in no way changes a your working dir or any**
- **commits that you've made.**
- Fetch before you work
- Fetch before you push
- Fetch often
- *git merge* must be done to merge fetched changes into local branch

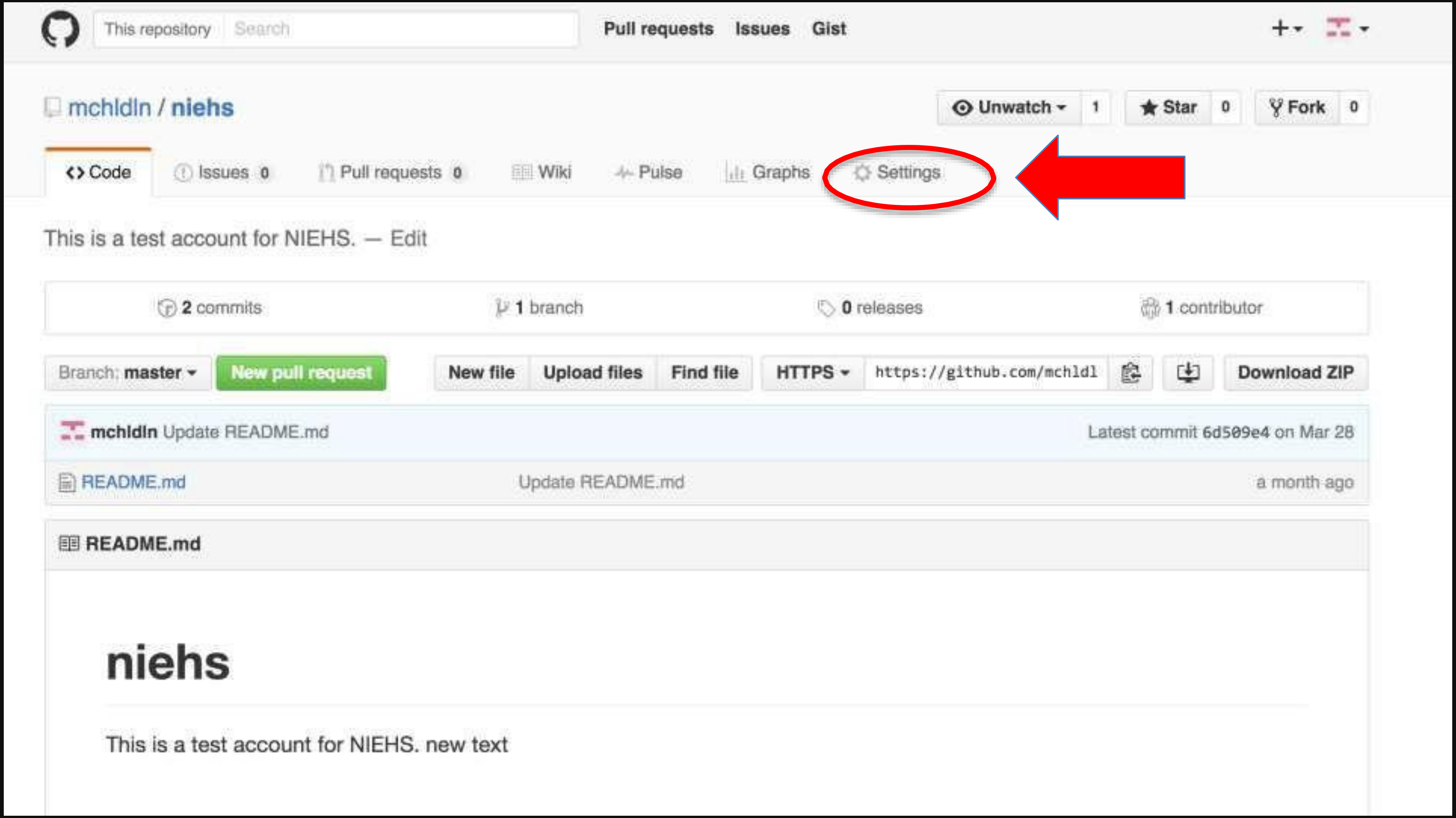
12 Collaborating using Git and GitHub



Atos

GitHub

Collaborating with Git



This is a test account for NIEHS. — Edit

2 commits | 1 branch | 0 releases | 1 contributor

mchldln Update README.md — Latest commit 6d509e4 on Mar 28

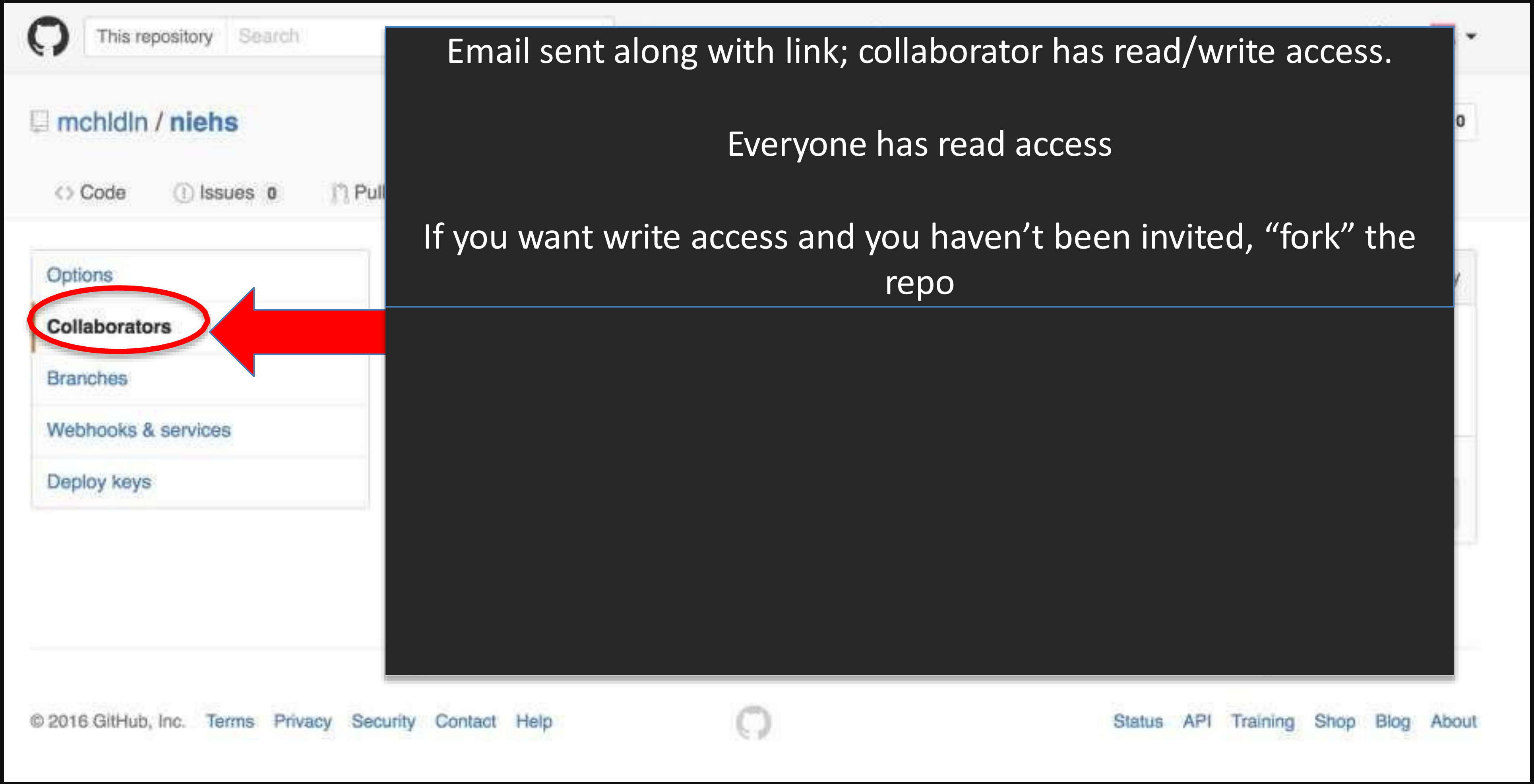
README.md — Update README.md — a month ago

README.md

niehs

This is a test account for NIEHS. new text

GitHub
Collaborating with Git



Email sent along with link; collaborator has read/write access.

Everyone has read access

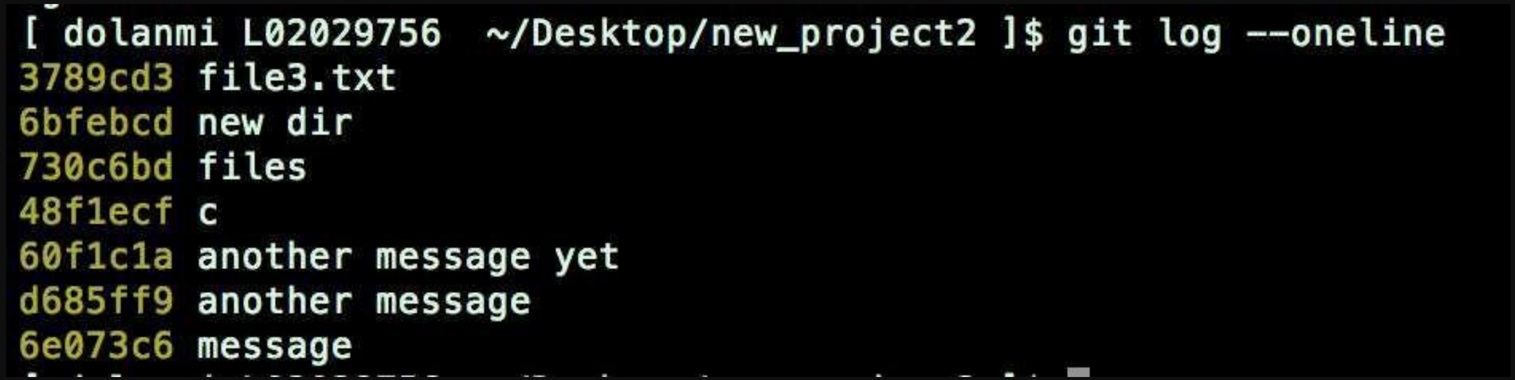
If you want write access and you haven't been invited, "fork" the repo

git commit -a

- Allows one to add to staging index and commit *at the same time*
- Grabs everything in working direcotry
- Files not tracked or being deleted are not included

git log --oneline

- gets first line and checksum of all commits in current branch



```
[ dolanmi L02029756  ~/Desktop/new_project2 ]$ git log --oneline
3789cd3 file3.txt
6bfebcd new dir
730c6bd files
48f1ecf c
60f1c1a another message yet
d685ff9 another message
6e073c6 message
```

git diff g5iU0oPe7x

When using checksum of older commit, will show you all changes compared to those in your working directory

Renaming and deleting branches

git branch –m/--move old_name new_name

git branch –d branch_name

Note: Must not be in branch_name

Note: Must not have commits in branch_name unmerged in branch from which you are deleting

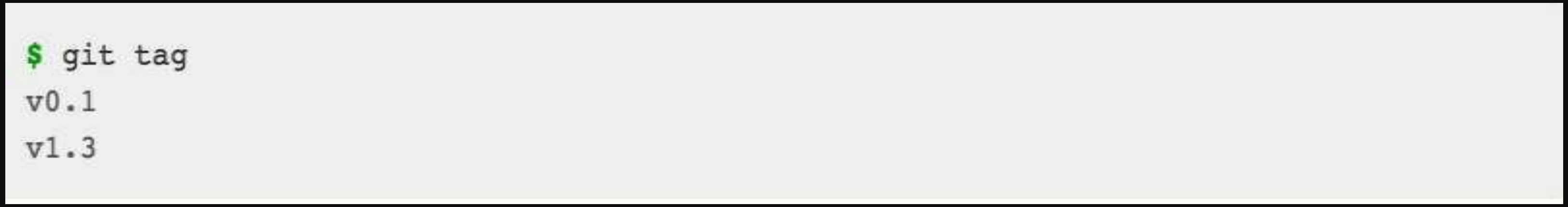
git branch –D branch_name

Note: If you are *really* sure that you want to delete branch with commits

Tagging

- Git has the ability to tag specific points in history as being important, such as releases versions (v.1.0, 2.0, ...)

git tag



```
$ git tag
v0.1
v1.3
```

Tagging

Two types of tags:

lightweight – a pointer to a specific comment – basically a SHA stored in a file

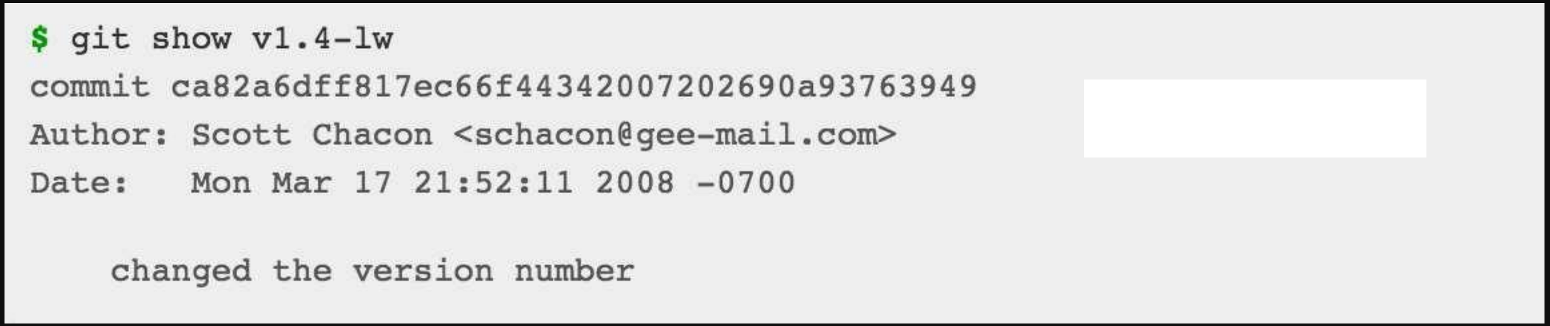
```
git tag tag_name
```

annotated – a full object stored in the Git database – SHA, tagger name, email, date, message and can be signed and verified with GNU Privacy Guard (GPG)

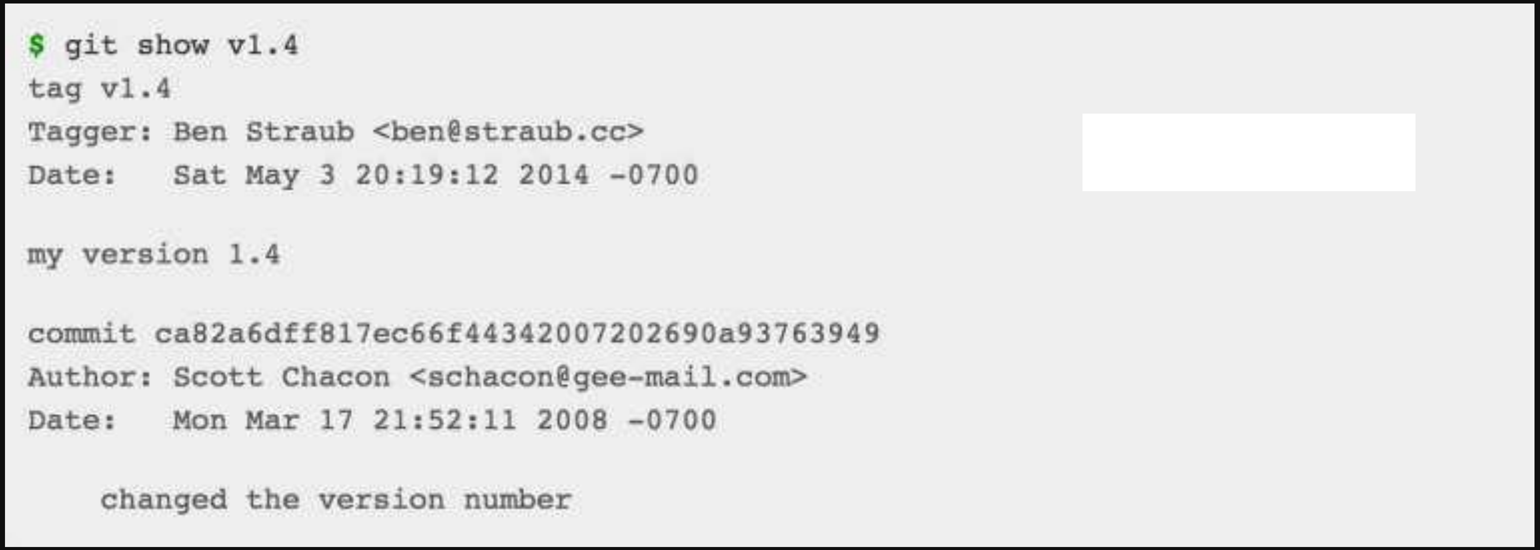
```
git tag –a tag_name –m “message”
```

How do I see tags?

git show tag_name



```
$ git show v1.4-lw
commit ca82a6dff817ec66f44342007202690a93763949
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Mon Mar 17 21:52:11 2008 -0700

    changed the version number
```



```
$ git show v1.4
tag v1.4
Tagger: Ben Straub <ben@straub.cc>
Date:   Sat May 3 20:19:12 2014 -0700

my version 1.4

commit ca82a6dff817ec66f44342007202690a93763949
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Mon Mar 17 21:52:11 2008 -0700

    changed the version number
```