

# Lecture 03

## version control with git



Course: Practical Bioinformatics (BIOL 4220)  
Instructor: Michael Landis  
Email: [michael.landis@wustl.edu](mailto:michael.landis@wustl.edu)



# Lecture 03 outline

Last time: modify filesystem

This time: version control

## git basics

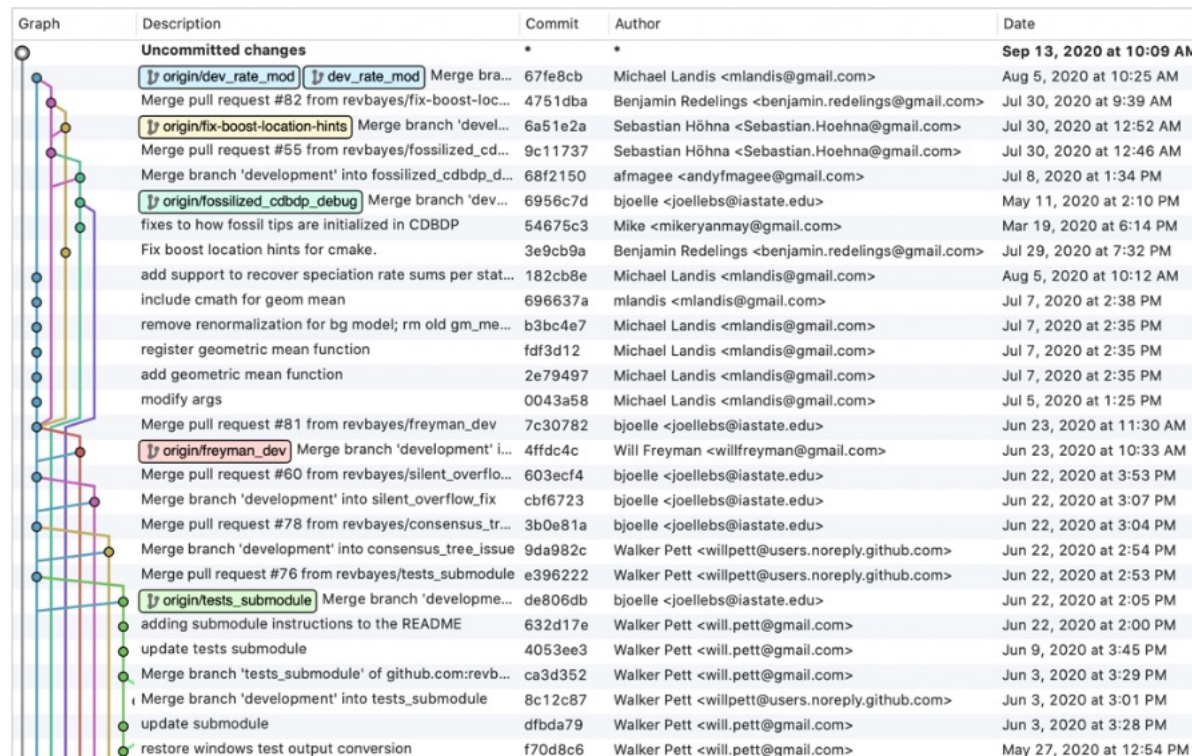
- repository anatomy
- stage (add) and commit
- branch and merge
- local and remote

# Version control by filename

```
Viburnum Biogeography - MD - Sep 4 2019.docx
Viburnum Biogeography - MJD - Aug 21 2019.docx
Viburnum Biogeography - MJD - PWS.docx
Viburnum Biogeography - MJD new.docx
Viburnum Biogeography - MJL - Aug 24 2019.docx
Viburnum Biogeography - MJL - Oct 17 2019.docx
Viburnum Biogeography - MJL - Oct 9 2019.docx
Viburnum Biogeography - MJL - Sep 25 2019.docx
Viburnum Biogeography - MJL edits - Aug 16 2019.docx
Viburnum Biogeography - MJL&MD edits 190731.docx
Viburnum Biogeography - MJL&MD edits 190812.docx
Viburnum Biogeography - MJL&MD edits 190812.orig.docx
Viburnum Biogeography - MJL&MD edits_ELS.docx
Viburnum Biogeography - manuscript - MJL edits 190731.docx
Viburnum Biogeography - supplement - MJL edits 190731.docx
ViburnumBiogeography_MJDnew_eje.docx .docx
Viburnum_phylogeny_manuscript_191018.docx
Viburnum_phylogeny_manuscript_191021.docx
Viburnum_phylogeny_manuscript_submitted_SystBiol_191020.pdf
Viburnum_phylogeny_supplement_191018.docx
Viburnum_phylogeny_supplement_191018_biorxiv.pdf
Viburnum_phylogeny_manuscript_191021_copy.docx
Viburnum_phylogeny_manuscript_191021.docx
Viburnum_phylogeny_manuscript_191027_MD_fresh.docx
Viburnum_phylogeny_manuscript_191027_MD_orig.docx
Viburnum_phylogeny_manuscript_200306_MJL_copy.docx
Viburnum_phylogeny_manuscript_200307_MJL.docx
Viburnum_phylogeny_manuscript_200310_MJL.docx
Viburnum_phylogeny_supplement_191018_copy.docx
Viburnum_phylogeny_supplement_191018.docx
Viburnum_phylogeny_supplement_191018_original.docx
Viburnum_phylogeny_supplement_191027_MD_orig.docx
Viburnum_phylogeny_supplement_200306_MJL.docx
Viburnum_phylogeny_supplement_200307_MD2 .docx
Viburnum_phylogeny_supplement_200307_MJL.docx
Viburnum_phylogeny_supplement_200310_MJL.docx
```

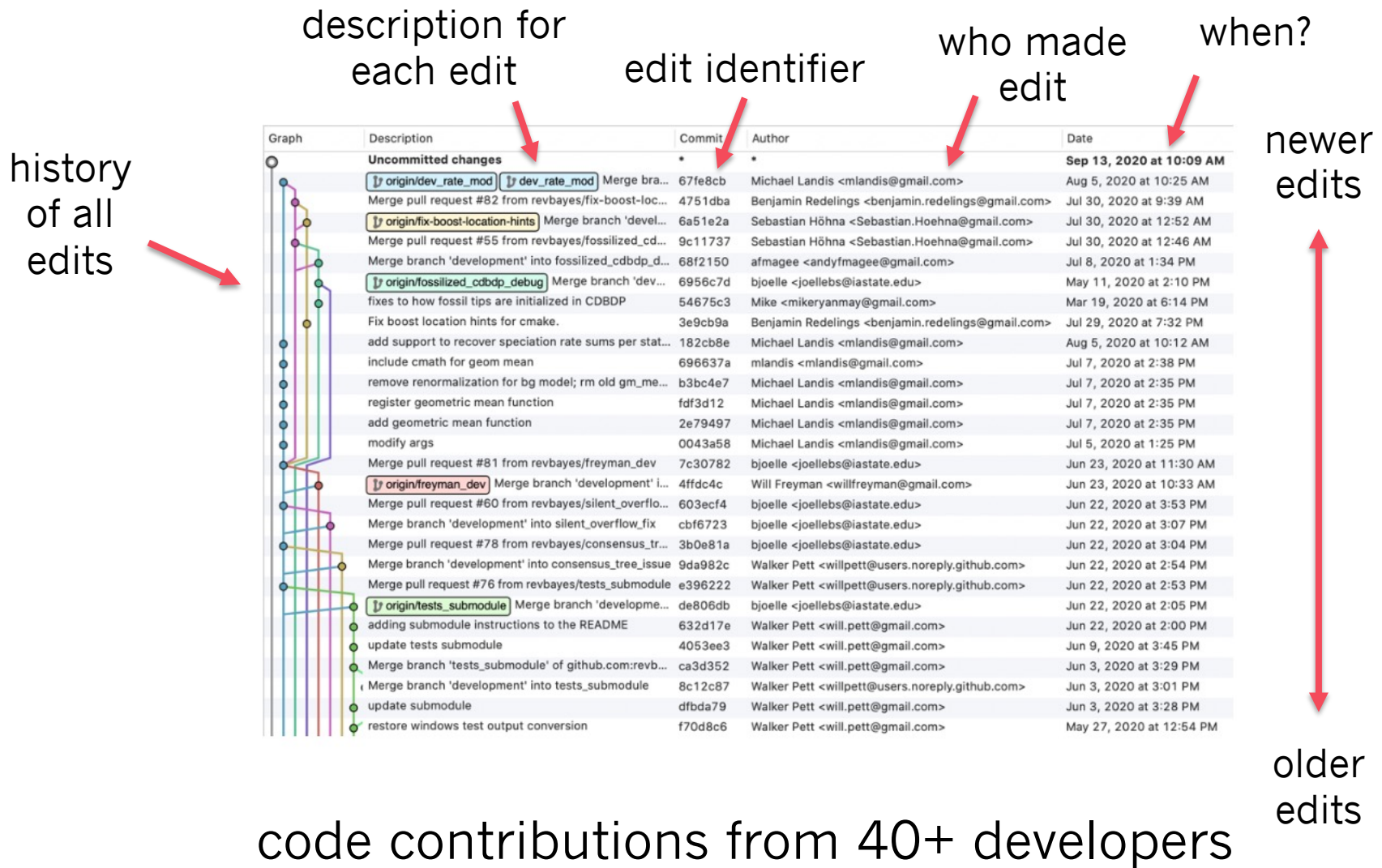
Word contributions from 5 co-authors

# Version control by software (git)



code contributions from 40+ developers

# Version control by software (git)



# Basics of git

git allows you to manage alternate histories and futures for a filesystem

- ***add*** files to monitor
- ***commit*** changed files to history
- ***branch*** to create alternate history
- ***merge*** to re-unify branched histories
- ***checkout*** commits/branches to recover past/alternate changes
- ***push*** histories to trusted collaborators
- ***pull*** histories from trusted collaborators

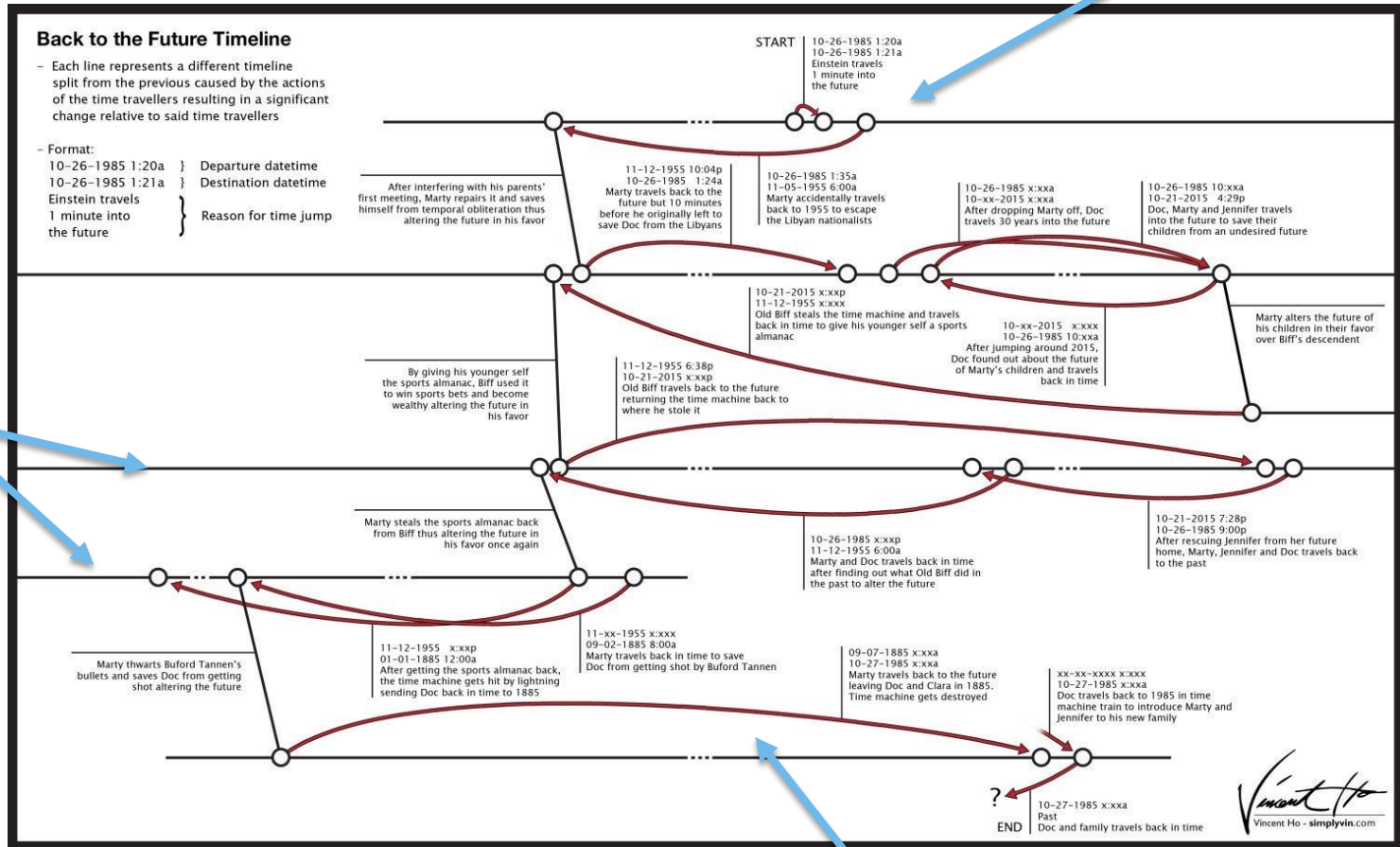




...using git

saved  
versions  
(**commit**)

alternate  
histories  
(**branch**,  
**merge**)



visit other  
history  
(**checkout**)

THIS IS GIT. IT TRACKS COLLABORATIVE WORK  
ON PROJECTS THROUGH A BEAUTIFUL  
DISTRIBUTED GRAPH THEORY TREE MODEL.

COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZE THESE SHELL  
COMMANDS AND TYPE THEM TO SYNC UP.  
IF YOU GET ERRORS, SAVE YOUR WORK  
ELSEWHERE, DELETE THE PROJECT,  
AND DOWNLOAD A FRESH COPY.





# 99% of git usage is saving local changes to your history

1. ***modify*** files in working directory
2. ***add*** modified files to staging area
3. ***commit*** staged files to repository history
4. 1% of the time, do something else
5. repeat

## Local repository

(e.g. on your VM)

### Working directory

*monitored files  
that you edit*



`git add`

### Staging area

*where files can be  
committed to history*



`git commit`

### Repository

*maintains  
commit history*




## Visualizing git spaces

- working directory
- staging area
- repository

Controls which files  
are saved, and how

# *git add*

moves file(s) from working directory into staging area

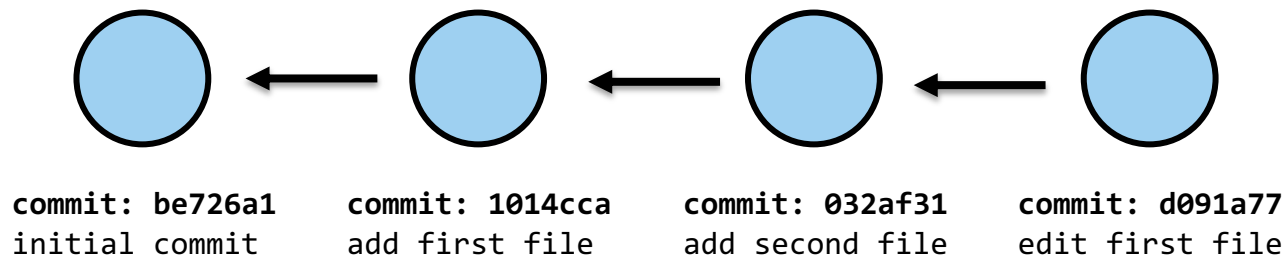
```
$ # status shows output.txt is untracked
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working
  directory)
        modified:   output.txt 
no changes added to commit (use "git add" and/or "git commit -a")
$ # instruct git to track output.txt
$ git add output.txt 
$ # status shows data.txt ready to be committed
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   output.txt 
$
```

# *git commit*

saves file(s) in staging area  
to the local repository

```
$ # status shows data.txt is staged to be committed
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   output.txt ←
$ # commit changes to repo with message (-m)
$ git commit -m 'edit output.txt' ←
[master 93c32e8] edit output.txt
 1 file changed, 1 insertion(+), 1 deletion(-)
$ # status shows all edits have been committed to repo
$ git status
On branch master
nothing to commit, working tree clean ←
```

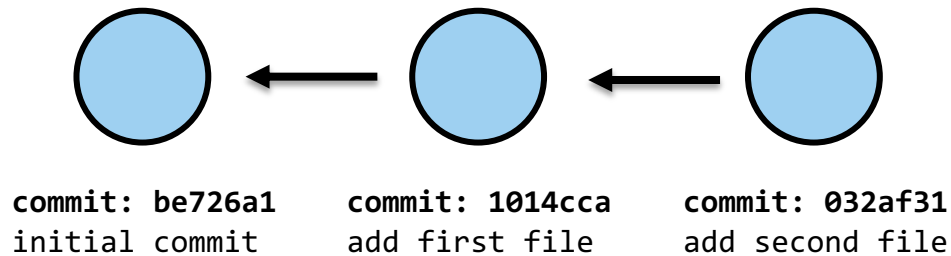
# Visualizing commit history as a graph



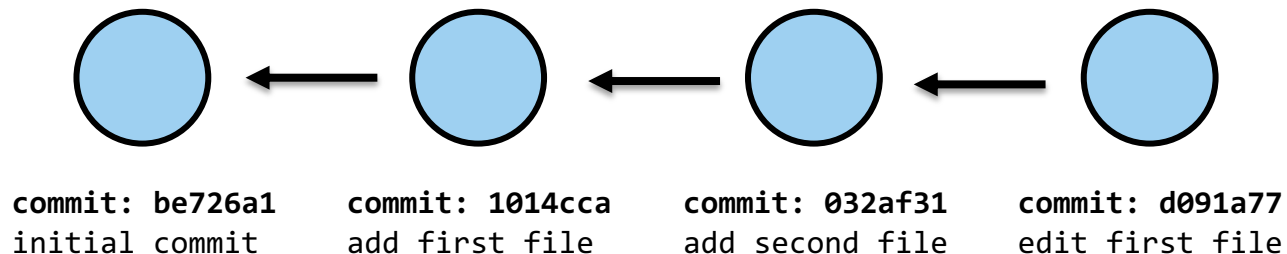
each **node** represents filesystem changes that have been committed to history

each **arrow** points toward the previous moment in history for that commit

# before *commit*



# after *commit*





# *git show*

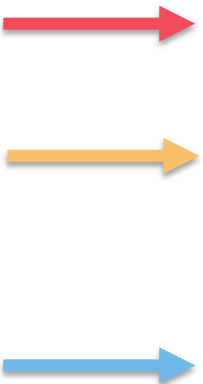
provides detailed info about commit/file  
(targets current commit, by default)

```
$ git show
commit 93c32e804c4ba667d61561ac8de49e6f19b6fcab (HEAD -> master)
Author: Michael Landis <mlandis@gmail.com>
Date:   Sat Sep 12 17:09:27 2020 -0500
    edit output.txt
diff --git a/output.txt b/output.txt
index af5626b..df5290f 100644
--- a/output.txt
+++ b/output.txt
@@ -1,1 @@
-Hello, world!
+Jello, warld!
```

shows differences in  
committed files

# *git status*

provides general info about commit  
and staged status for all files in repo



```
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   run.sh
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working
directory)
    modified:   output.txt
Untracked files:
  (use "git add <file>..." to include in what will be committed)
data.txt
```

run.sh is in staging  
area, ready to be  
committed

add tracked file  
*output.txt* to staging  
area for commit

add untracked file  
*data.txt* to monitor  
changes

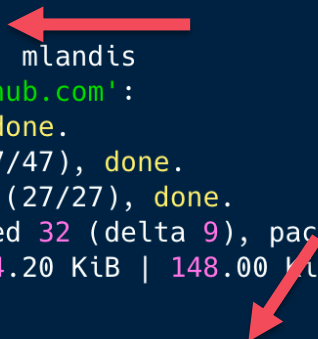
# 1% of git usage is managing and sharing commits

- **clone** repo from online server
- **checkout** other committed versions
- **revert** an unwanted commit
- **pull** commits from another repo
- **push** commits to another repo
- make a new **branch** of commits
- **merge** commits between two branches

# *git clone*


copies remote repo to local directory

```
# directory contents before clone
~/labs$ ls
lab-01a-mlandis  lab-01b-mlandis
# clone lab-02a-mlandis from github.com
~/labs$ git clone https://github.com/WUSTL-Biol4220/lab-02a-mlandis.git
Cloning into 'lab-02a-mlandis'...
Username for 'https://github.com': mlandis
Password for 'https://mlandis@github.com':
remote: Enumerating objects: 47, done.
remote: Counting objects: 100% (47/47), done.
remote: Compressing objects: 100% (27/27), done.
remote: Total 47 (delta 13), reused 32 (delta 9), pack-reused 0
Unpacking objects: 100% (47/47), 4.20 KiB | 148.00 KiB/s, done.
# directory contents after clone
~/labs$ ls
lab-01a-mlandis  lab-01b-mlandis  lab-02a-mlandis
# lab-02a-mlandis contains files from github.com repo
~/labs$ ls lab-02a-mlandis/
data  README.md
```


Two red arrows are present in the terminal output. The first arrow points from the right towards the text 'lab-02a-mlandis' in the 'Cloning into' line. The second arrow points from the right towards the 'lab-02a-mlandis' directory listing in the 'directory contents after clone' section.

# git checkout

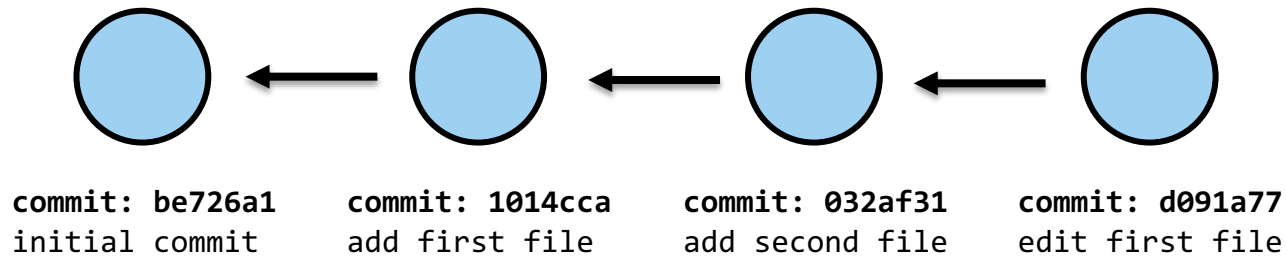
replace filesystem with files from  
previous or alternative histories;  
*extremely versatile*



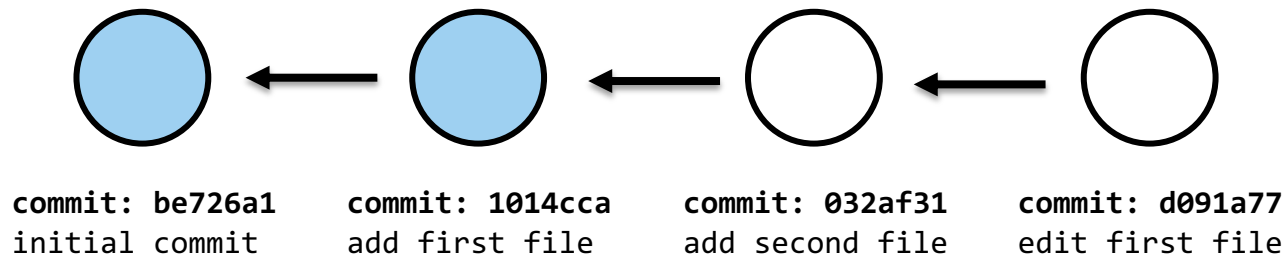
```
# edit README.md
~/labs/lab-02a-mlandis$ nano README.md
# add new version to staging area for commit
~/labs/lab-02a-mlandis$ git add README.md
# commit staged files
~/labs/lab-02a-mlandis$ git commit -m 'fix typo'
[master 71cfa39] fix typo
1 file changed, 1 insertion(+)
~/labs/lab-02a-mlandis$ git add README.md; git commit -m 'fix another'
[master a2c2c16] fix another typo
1 file changed, 1 insertion(+)
# suppose we wanted to _view_ how repo
# appeared in earlier commit 71cfa39
~/labs/lab-02a-mlandis$ git checkout 71cfa39
Note: switching to '71cfa39'.
You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in
this state without impacting any branches by switching back to a branch.
(lengthy warning message)
```



# before *checkout*



# after *checkout*



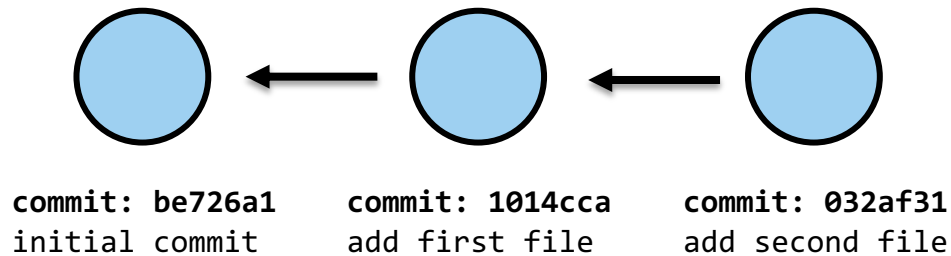


# *git revert*

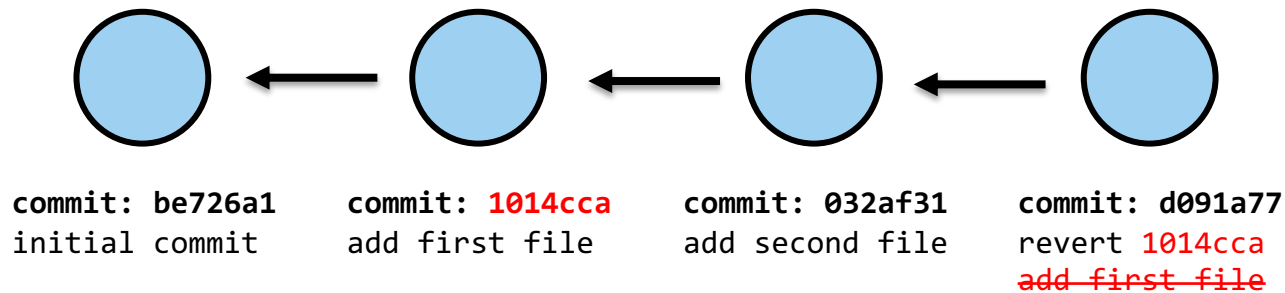
adds commit to history that  
undoes edits from previous commit

```
# intentionally add a typo to README.md
$ nano README.md
# commit the typo to history
$ git commit -am 'make typo'
→ [master 8bedaa9] make typo
   1 file changed, 2 insertions(+)
# undo the commit containing the typo by #
  adding a 'revert' commit to the history
$ git revert 8bedaa9 ←
[master 26abd8e] Revert "make typo"
   1 file changed, 2 deletions(-)
```

# before *revert*




# after *revert*



# *git pull*

retrieves committed changes from  
another repo (e.g. from GitHub)



```
# retrieve changes from github.com
~/labs/lab-02a-mlandis$ git pull
Username for 'https://github.com': mlandis
Password for 'https://mlandis@github.com':
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 681 bytes | 681.00 KiB/s, done.
From https://github.com/WUSTL-Biol4220/lab-02a-mlandis
    b34d17b..0245064  master    -> origin/master
Merge made by the 'recursive' strategy.
 README.md | 1 +
 1 file changed, 1 insertion(+)
```

## Local repository

(e.g. on your VM)

### Working directory

*monitored files  
that you edit*



`git add`

### Staging area

*where files can be  
committed to history*



`git commit`

### Repository

*maintains  
commit history*

*pull* commits  
from remote  
repository

`git  
pull`

## Remote repository

(e.g. on GitHub)

### Repository

*maintains  
commit history*



# *git push*

sends local committed changes to  
another repo (e.g. to GitHub)

```
# send local commits to github.com
~/labs/lab-02a-mlandis$ git push
Username for 'https://github.com': mlandis
Password for 'https://mlandis@github.com':
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 2 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (9/9), 1.08 KiB | 551.00 KiB/s, done.
Total 9 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), completed with 1 local
object.
To https://github.com/WUSTL-Biol4220/lab-02a-mlandis.git
  0245064..6105673  master -> master
```

## Local repository

(e.g. on your VM)

### Working directory

*monitored files  
that you edit*



`git add`

### Staging area

*where files can be  
committed to history*



`git commit`

### Repository

*maintains  
commit history*

`git  
push`



*push* commits  
to remote  
repository

## Remote repository

(e.g. on GitHub)

### Repository

*maintains  
commit history*



# *git branch*

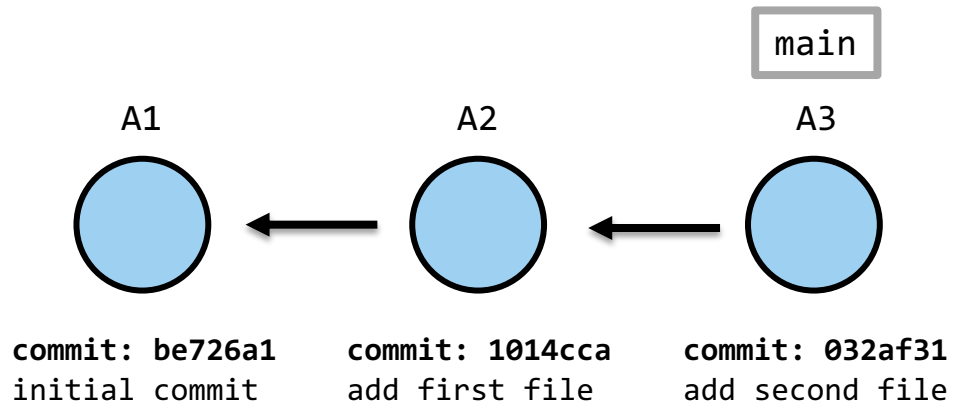
create a new branch of commit histories;  
... or switch between branches

```
# we are on the 'master' branch
~/labs/lab-02a-mlandis$ git branch
* master
# split off a new branch from 'master', called 'fix'
~/labs/lab-02a-mlandis$ git branch fix
# switch to the new 'fix' branch
mlandis@biol4220-mlandis:~/labs/lab-02a-mlandis$ git checkout
fix
Switched to branch 'fix'
# we are now on the 'fix branch'
mlandis@biol4220-mlandis:~/labs/lab-02a-mlandis$ git branch
* fix
master
```

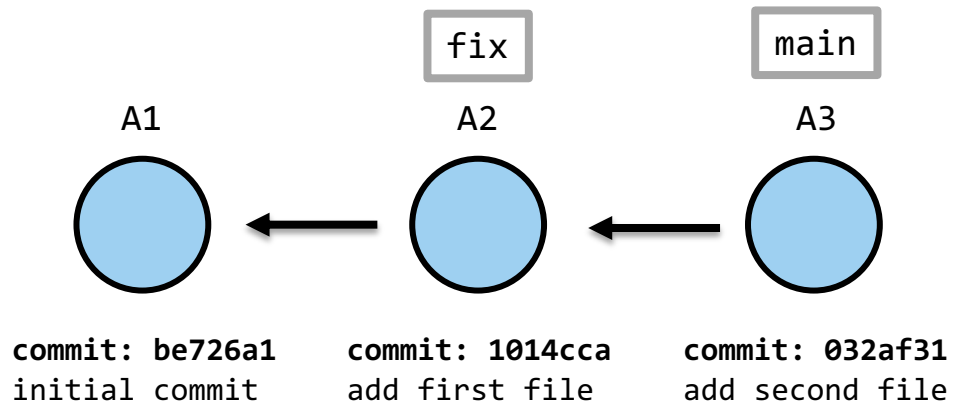
# *git merge*

merge another branch into  
your current branch

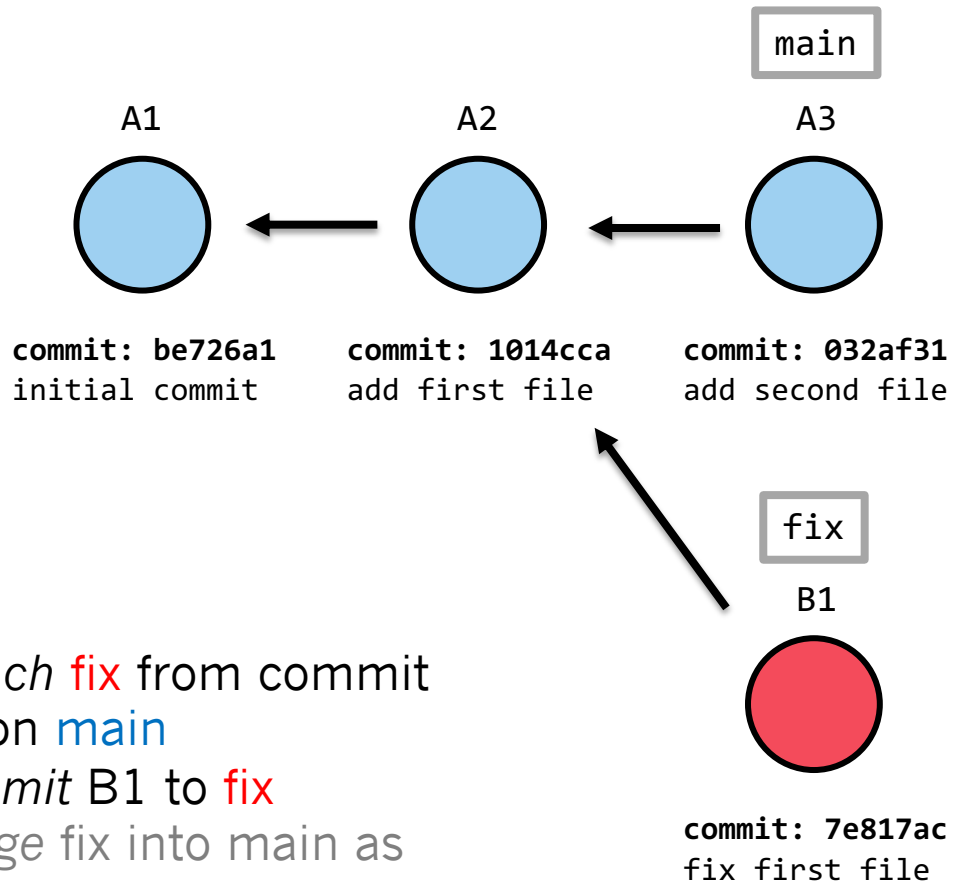
```
# start on the 'fix' branch
# add and commit new file
~/labs/lab-02a-mlandis$ touch file.txt
~/labs/lab-02a-mlandis$ git add file.txt
~/labs/lab-02a-mlandis$ git commit -m 'new file'
[fix 377c248] new file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file.txt
# switch back to 'master' branch
mlandis@biol4220-mlandis:~/labs/lab-02a-mlandis$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
# merge the 'fix' branch _into_ 'master'
~/labs/lab-02a-mlandis$ git merge fix
Updating 6105673..377c248
Fast-forward
 file.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file.txt
# new file now appears in 'master'
~/labs/lab-02a-mlandis$ ls
file.txt
```



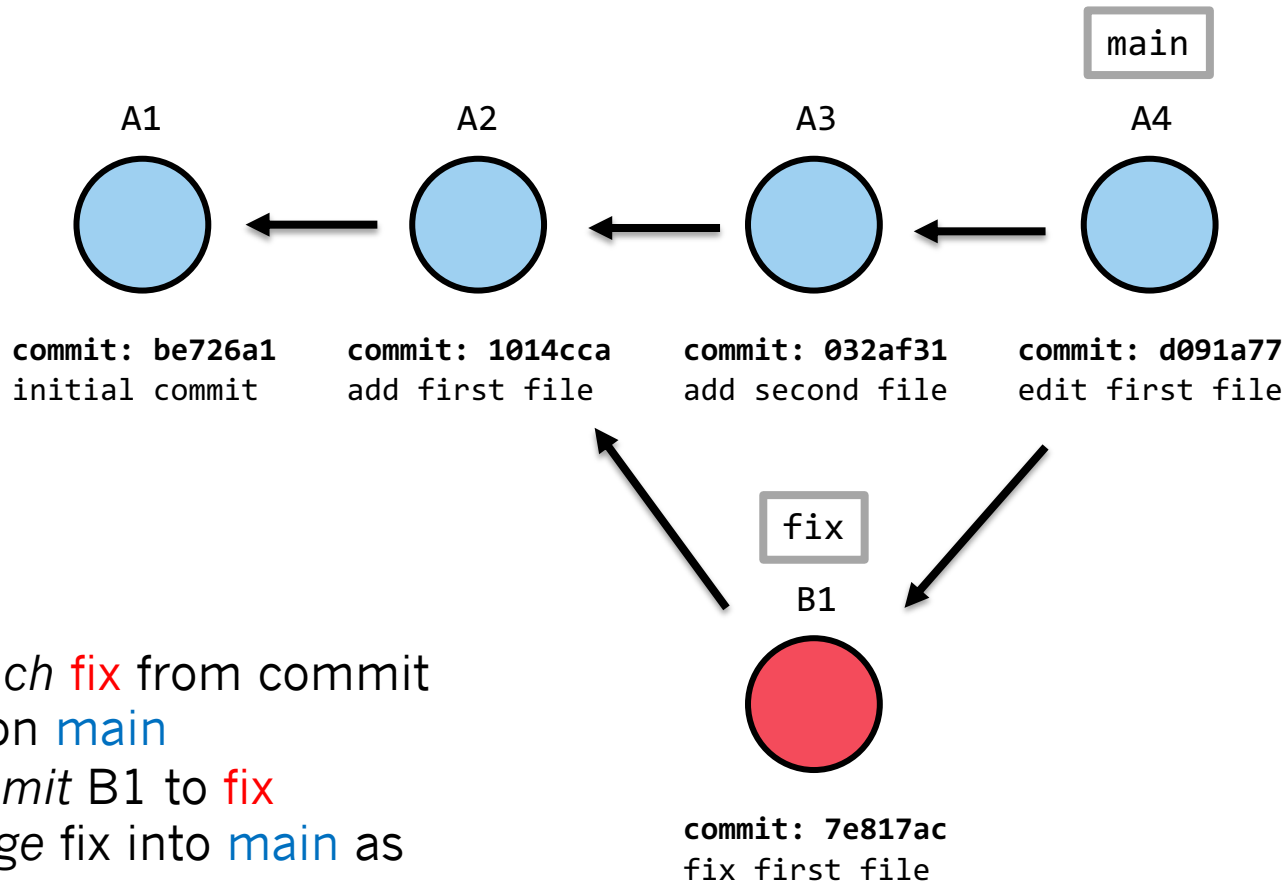
1. *branch* fix from commit A2 on main
2. *commit* B1 to fix
3. *merge* fix into main as commit A4



1. *branch* **fix** from commit A2 on **main**
2. *commit* B1 to fix
3. *merge* fix into main as commit A4



1. *branch* **fix** from commit A2 on **main**
2. *commit* B1 to **fix**
3. *merge* fix into main as commit A4

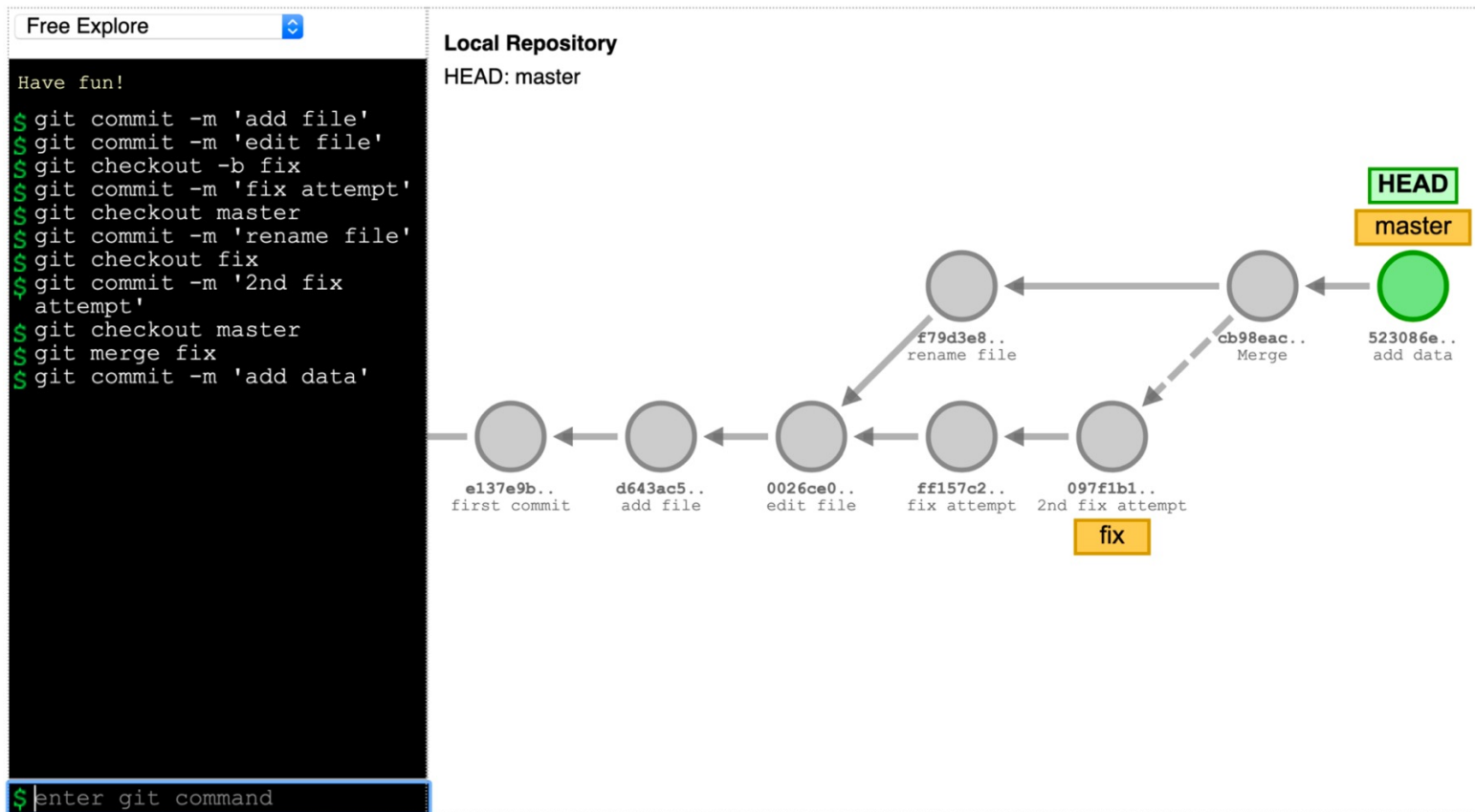


1. branch **fix** from commit A2 on **main**
2. commit B1 to **fix**
3. merge fix into **main** as commit A4



More tools to visualize  
relationships between  
commands, graphs,  
and filesystems

# <https://git-school.github.io/visualizing-git>



<https://learngitbranching.js.org>

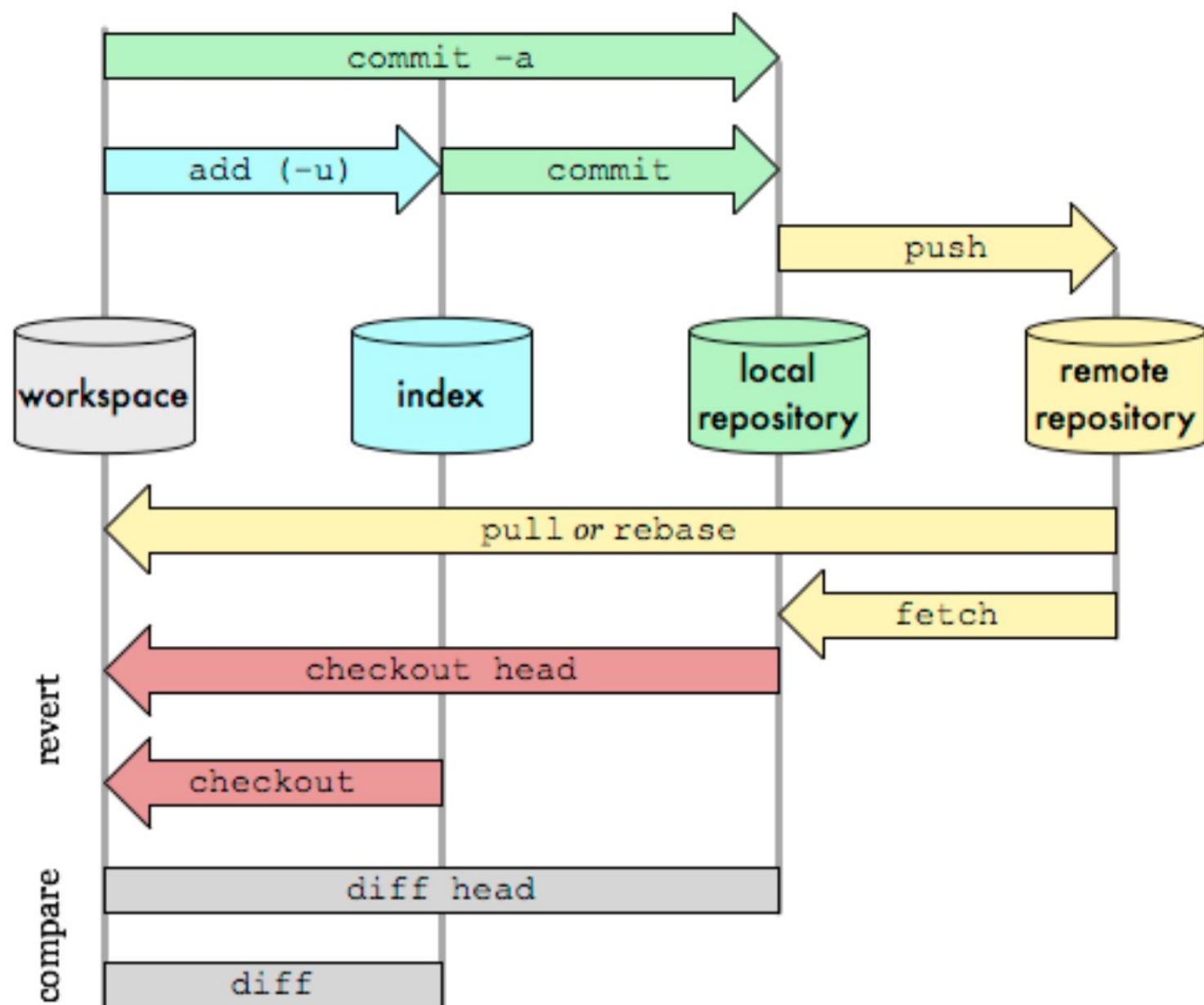
The screenshot displays the 'Learn Git Branching' website interface. On the left, a terminal window shows the following commands and their outputs:

```
$ level rampup1 [checked]
$ hint [checked]
Use the label (hash) on the commit for help!
$ delay 2000 [checked]
$ show goal [checked]
$ git checkout bugFix [checked]
$ git status [checked]
# On branch bugFix
# Changes to be committed:
#   modified:   cal/0skiCostume.stl
# Ready to commit! (as always in this demo)
$ objective [checked]
$ git checkout C4 [checked]
```

On the right, a Git branching diagram illustrates the workflow. The diagram shows a sequence of commits: C0 (purple) points to C1 (purple), which points to C2 (pink). A 'master' branch label points to C2. From C1, a branch 'bugFix' (green) is created, leading to C3 (green) and then C4 (green). A 'HEAD' label points to C4. A 'bugFix' branch label points to C4. A 'Fork me on GitHub' banner is visible in the top right corner. Social media icons for GitHub, Twitter, and Facebook are in the bottom right corner.

# Git Data Transport Commands

<http://osteele.com>



# Overview for Lab 03