Git and GitHub

CMPUT 301
Winter 2025

Akalanka Galappaththi
Krish Lal

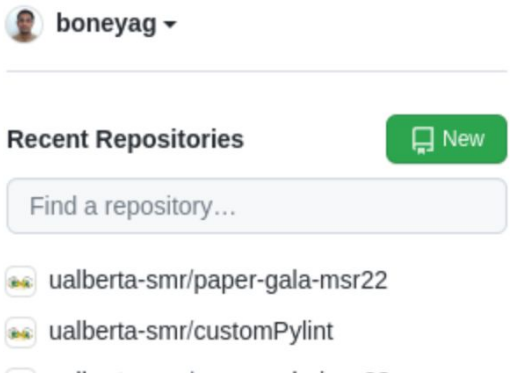
Git merge conflict demo is based on LN Wilson's lab worksheet (Dept. Math. & Computer Since, University of Lethbridge)

*Last updated: 2025/01/04

Create a repo in GitHub

- Visit the URL on your browser.
- Create a repo.

`https://github.com/`



Create a repo in GitHub

Repository template
Start your repository with a template repository's contents.

No template

Owner * **Repository name** *

boneyag / test-301-tue

Great repository names are short and memorable. Need inspiration? How about **sturdy-journey**?

Description (optional)

Test repo for Tuesday lab

Public
Anyone on the internet can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

Add a README file
This is where you can write a long description for your project. Learn more.

Add .gitignore
Choose which files not to track from a list of templates. Learn more.

.gitignore template: None

Choose a license
A license tells others what they can and can't do with your code. Learn more.

License: None

You are creating a public repository in your personal account.

Create repository

Short memorable name

Who could see your repo

Automatically create some files

Use the https opt. (ssh is out of the scope)

Quick setup — if you've done this kind of thing before

or HTTPS SSH `https://github.com/boneyag/test-301-tue.git`

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

Your username Your repo name

Copy URL and paste on the terminal (make sure you to change dir to a desired location

...or create a new repository on the command line

```
echo "# test-301-tue" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/boneyag/test-301-tue.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/boneyag/test-301-tue.git
git branch -M main
git push -u origin main
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

Clone a Git repo

```
akalanka@akalanka-ThinkPad: ~/GitDemo/test-301Tue
akalanka@akalanka-ThinkPad:~$ mkdir GitDemo
akalanka@akalanka-ThinkPad:~$ cd GitDemo
akalanka@akalanka-ThinkPad:~/GitDemo$ git clone https://github.com/boneyag/test-
301Tue.git
Cloning into 'test-301Tue'...
warning: You appear to have cloned an empty repository.
akalanka@akalanka-ThinkPad:~/GitDemo$ ls
test-301Tue
akalanka@akalanka-ThinkPad:~/GitDemo$ cd test-301Tue/
akalanka@akalanka-ThinkPad:~/GitDemo/test-301Tue$ ls
akalanka@akalanka-ThinkPad:~/GitDemo/test-301Tue$
```

1. Make a dir
2. Change the dir
3. Use the command `git clone`
4. Copy git URL after
5. Hit return

**Use `git clone` when copy the repo to your computer for the first time. After that we use `git pull`.

Add a file to the repo locally

```
akalanka@Akalanka-ThinkPad:~/Documents/TA/CMPUT301_Fall22/Lab4$ git clone https://github.com/boneyag/tes
t-301-tue.git
Cloning into 'test-301-tue'...
warning: You appear to have cloned an empty repository.
akalanka@Akalanka-ThinkPad:~/Documents/TA/CMPUT301_Fall22/Lab4$ ls
test-301-tue
akalanka@Akalanka-ThinkPad:~/Documents/TA/CMPUT301_Fall22/Lab4$ cd test-301-tue/
akalanka@Akalanka-ThinkPad:~/Documents/TA/CMPUT301_Fall22/Lab4/test-301-tue$ ls
akalanka@Akalanka-ThinkPad:~/Documents/TA/CMPUT301_Fall22/Lab4/test-301-tue$ code README.md
akalanka@Akalanka-ThinkPad:~/Documents/TA/CMPUT301_Fall22/Lab4/test-301-tue$ ls
README.md
akalanka@Akalanka-ThinkPad:~/Documents/TA/CMPUT301_Fall22/Lab4/test-301-tue$ more README.md
This is a test repository for CMPUT 301 Tuesday lab (Fall 2022)
akalanka@Akalanka-ThinkPad:~/Documents/TA/CMPUT301_Fall22/Lab4/test-301-tue$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README.md

nothing added to commit but untracked files present (use "git add" to track)
akalanka@Akalanka-ThinkPad:~/Documents/TA/CMPUT301_Fall22/Lab4/test-301-tue$
```

```
akalanka@Akalanka-ThinkPad:~/Documents/TA/CMPUT301_Fall22/Lab4/test-301-tue$ git add README.md
akalanka@Akalanka-ThinkPad:~/Documents/TA/CMPUT301_Fall22/Lab4/test-301-tue$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README.md

akalanka@Akalanka-ThinkPad:~/Documents/TA/CMPUT301_Fall22/Lab4/test-301-tue$ git commit -m "Add a readme
 file"
[main (root-commit) 077d32b] Add a readme file
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
```

```
akalanka@Akalanka-ThinkPad:~/Documents/TA/CMPUT301_Fall22/Lab4/test-301-tue$ git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 275 bytes | 275.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/boneyag/test-301-tue.git
 * [new branch]      main -> main
akalanka@Akalanka-ThinkPad:~/Documents/TA/CMPUT301_Fall22/Lab4/test-301-tue$
```

Create a repo locally

```
akalanka@Akalanka-ThinkPad:~/Documents/TA/CMPUT301_Fall22/Lab4$ mkdir test-301-tue-2
akalanka@Akalanka-ThinkPad:~/Documents/TA/CMPUT301_Fall22/Lab4$ ls
test-301-tue  test-301-tue-2
akalanka@Akalanka-ThinkPad:~/Documents/TA/CMPUT301_Fall22/Lab4$ cd test-301-tue-2
akalanka@Akalanka-ThinkPad:~/Documents/TA/CMPUT301_Fall22/Lab4/test-301-tue-2$ code README.md
akalanka@Akalanka-ThinkPad:~/Documents/TA/CMPUT301_Fall22/Lab4/test-301-tue-2$ more README.md
Creating a repo locally
akalanka@Akalanka-ThinkPad:~/Documents/TA/CMPUT301_Fall22/Lab4/test-301-tue-2$ git status
fatal: not a git repository (or any of the parent directories): .git
akalanka@Akalanka-ThinkPad:~/Documents/TA/CMPUT301_Fall22/Lab4/test-301-tue-2$
```

```
akalanka@Akalanka-ThinkPad:~/Documents/TA/CMPUT301_Fall22/Lab4/test-301-tue-2$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/akalanka/Documents/TA/CMPUT301_Fall22/Lab4/test-301-tue-2/.git
```

```
akalanka@Akalanka-ThinkPad:~/Documents/TA/CMPUT301_Fall22/Lab4/test-301-tue-2$ git branch -m main
```

```
akalanka@Akalanka-ThinkPad:~/Documents/TA/CMPUT301_Fall22/Lab4/test-301-tue-2$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README.md

nothing added to commit but untracked files present (use "git add" to track)
akalanka@Akalanka-ThinkPad:~/Documents/TA/CMPUT301_Fall22/Lab4/test-301-tue-2$
```

```
code README.md
git init
git status
git add README.md
git status
git commit -m "<message>"
git remote add origin <URL>
git push origin master
```

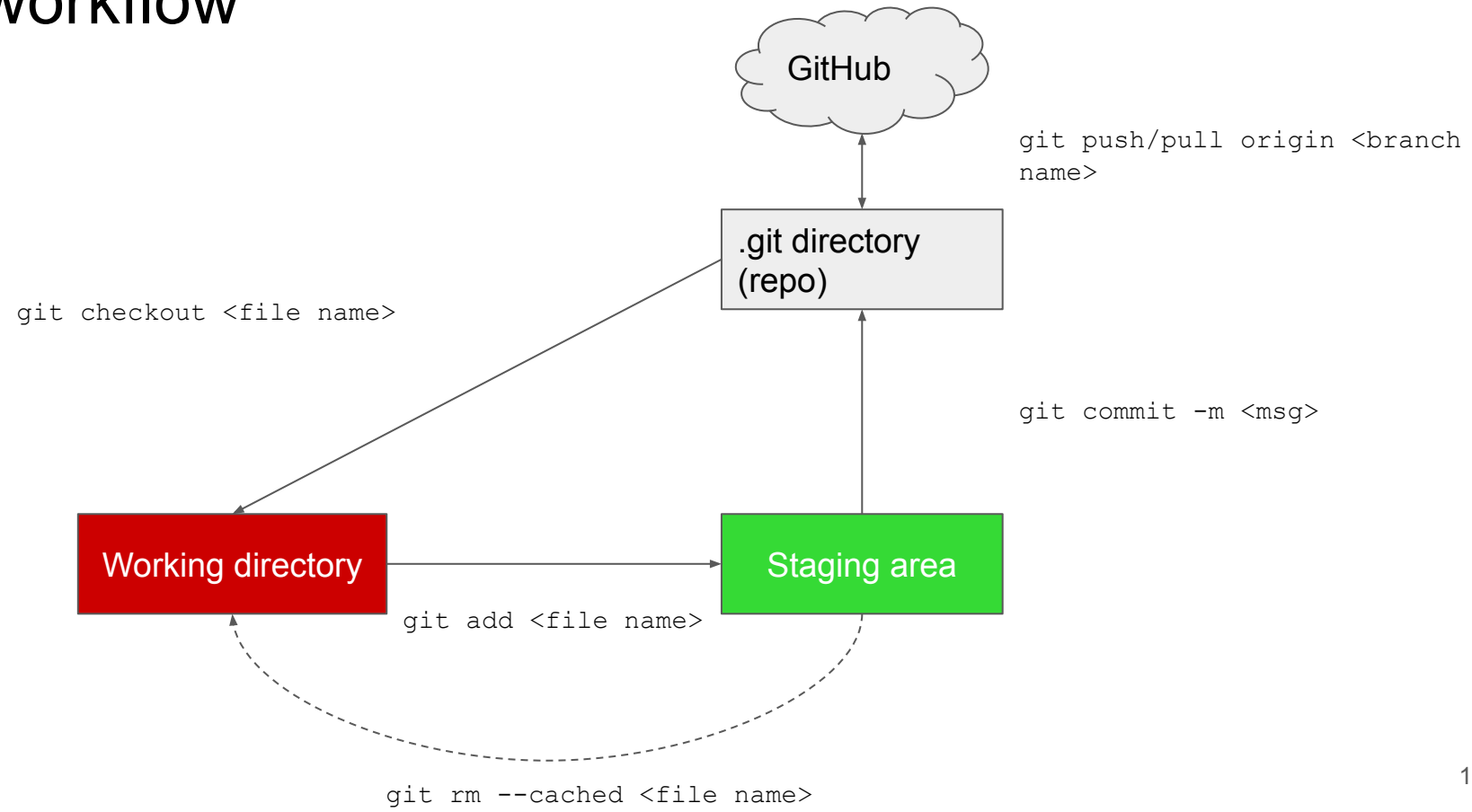
...or push an existing repository from the command line

```
git remote add origin https://github.com/boneyag/test-301-tue-2.git
git branch -M main
git push -u origin main
```

```
akalanka@Akalanka-ThinkPad:~/Documents/TA/CMPUT301_Fall22/Lab4/test-301-tue-2$ git remote add origin htt
ps://github.com/boneyag/test-301-tue-2.git
akalanka@Akalanka-ThinkPad:~/Documents/TA/CMPUT301_Fall22/Lab4/test-301-tue-2$ git remote -v
origin  https://github.com/boneyag/test-301-tue-2.git (fetch)
origin  https://github.com/boneyag/test-301-tue-2.git (push)
akalanka@Akalanka-ThinkPad:~/Documents/TA/CMPUT301_Fall22/Lab4/test-301-tue-2$
```

```
akalanka@Akalanka-ThinkPad:~/Documents/TA/CMPUT301_Fall22/Lab4/test-301-tue-2$ git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 238 bytes | 238.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/boneyag/test-301-tue-2.git
 * [new branch]      main -> main
```

Git workflow



Practice the common commands

- Create two new text files and push those to Github repo
  - `echo “Test 1” >> test1.txt`
  - `echo “Test 2” >> test2.txt`
  - `git add - A`
  - `git commit -m “<message>”`
  - `git push origin master`
- Useful git commands
  - `git pull origin <branch>`
  - `git reset -` remove files from the staging area
  - `git rm --cached <filename> -` remove a file from working index
  - `git rm -f <filename> -` remove a file forcefully `(-f)`
  - `git rm -rf <dir name> -` remove a directory forcefully `(-r` recursively`)`
  - `git log -` view commit history
  - `git clone <https URL>`

Merge conflicts

Ex: Modifying files in the project.

- TM1: change file1
- TM2: change file2
- Both push changes

- TM1: change file1 -> push
- TM2: change file1, file2 -> try to push

https://www.flaticon.com/

Let’s get your hands dirty -- demo a merge conflict

Leave the current repo dir (cd ..), open two terminals (pretend those as two users of the repo).
Make sure to follow the order of execution of commands.

	Terminal 1	Terminal 2
1	`mkdir t1`	`mkdir t2`
2	`cd t1`	
3	`git clone <your repo URL>`	`cd t2`
4		`git clone <your repo URL>`
5		`cd test-301-tue`
6		`mkdir one`
7		`echo “Lin1” >> one/file1.txt`

Let’s get your hands dirty -- demo a merge conflict

```
8                                   git status
9                                   git add one/file1.txt
10                                  git commit -m “informative-message”
11                                  git push origin main
12  cd test-301Tue
13  git pull origin main
14  mkdir two
15  code two/file2.txt
16  git add two/
```

Let’s get your hands dirty -- demo a merge conflict

```
17  git commit -m “<msg>”
18  git push origin main
19                                  cd one
20                                  code file1.txt # add two lines
21                                  git add file1.txt
22                                  git commit -m “<msg>”
23                                  git push origin main
24                                  git fetch origin
25                                  git merge origin main
```

No conflict as users pushed different files

Will not push until you pull remote changes

Let’s get your hands dirty -- demo a merge conflict

```
26                                                  git push origin main
27  git pull origin main
28  code two/file2.txt
29  git add two/file2.txt
30  git commit -m “<msg>”
31  git push origin main
32                                                  cd two
33                                                  code file2.txt
34                                                  git add two/file2.txt
```

Need to pull changes from origin

Add two more lines

No conflict as users pushed different files

Add two more lines

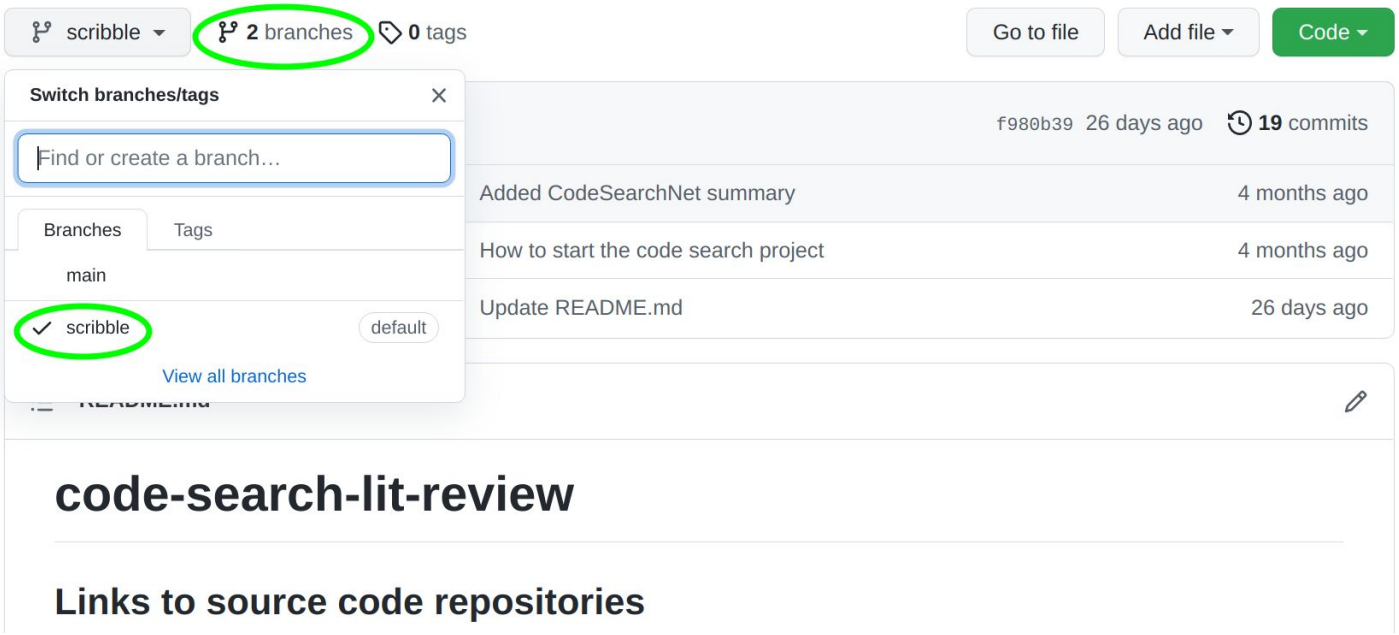
Let’s get your hands dirty -- demo a merge conflict

```
35                                  git commit -m “<msg>”
36                                  git push origin main
37                                  git fetch origin
38                                  git merge origin main
39                                  # Fix the conflict as you see fit #
40                                  git add two/
41                                  git commit -m “<msg>”
42                                  git push origin master
43
```

Need to pull changes from origin

Conflict

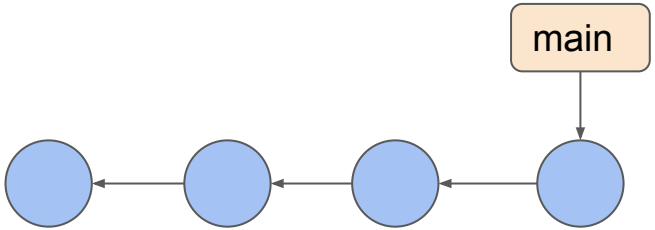
Git branches



Git branches

What is a branch?

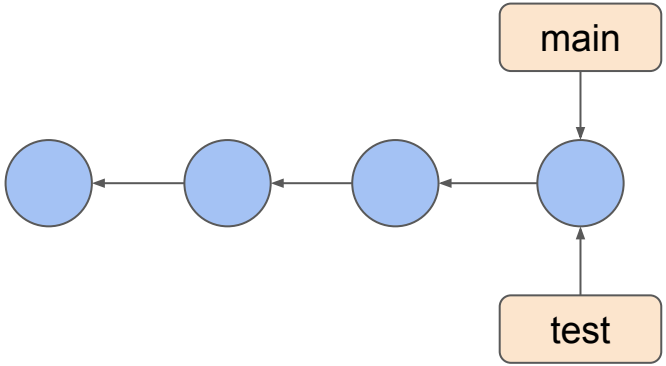
- Branch shows the evolution of your project (commits)
- Each commit has SHA-1 value which allows you to revert back to that state.



Git branches

- Default branch -- main
- Nothing special about this branch.
  - You can rename it.
  - Nobody bother to do that, so that it remains with the default name.
- Create a branch in command line

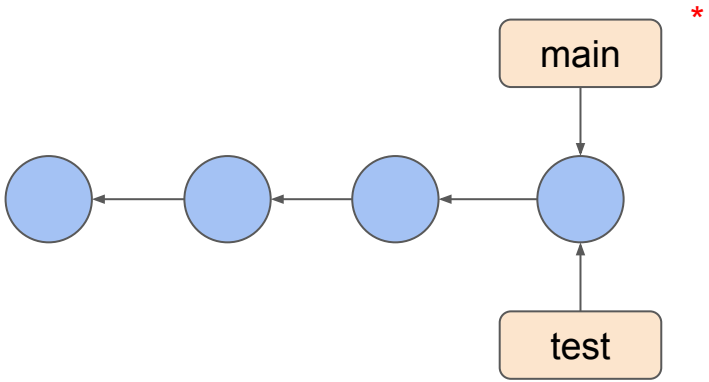
```
git branch test
```



Git branches

- Current working branch
- Change the branch

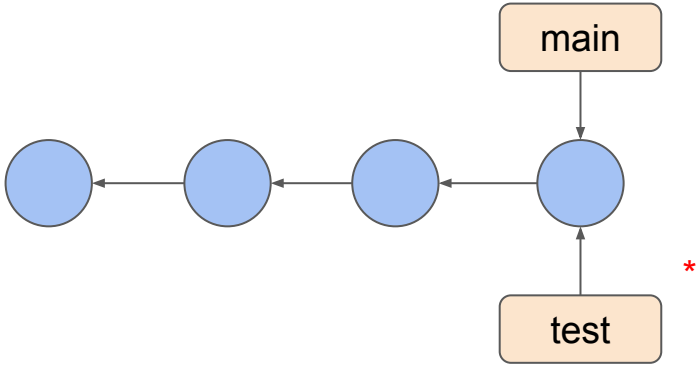
```
git status
```



Git branches

- Current working branch
- Change the branch

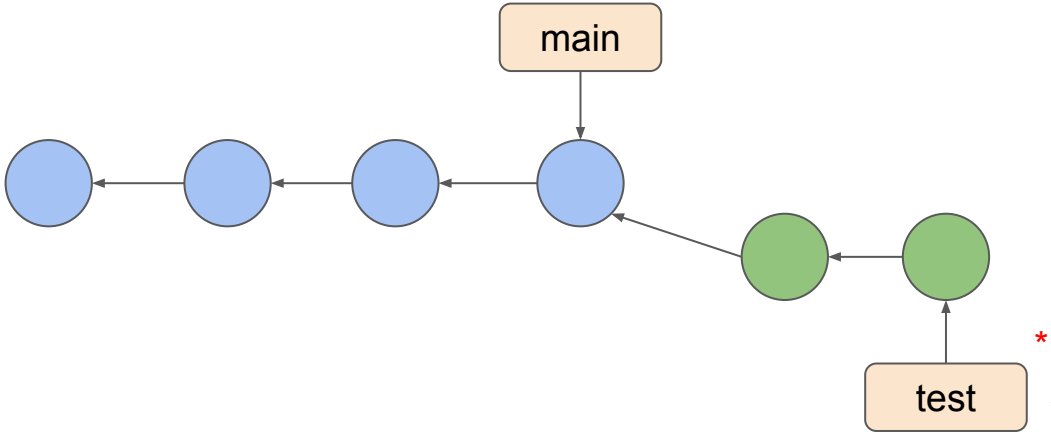
```
git status

git checkout test
```



Git branches

- Do some work on `test` branch

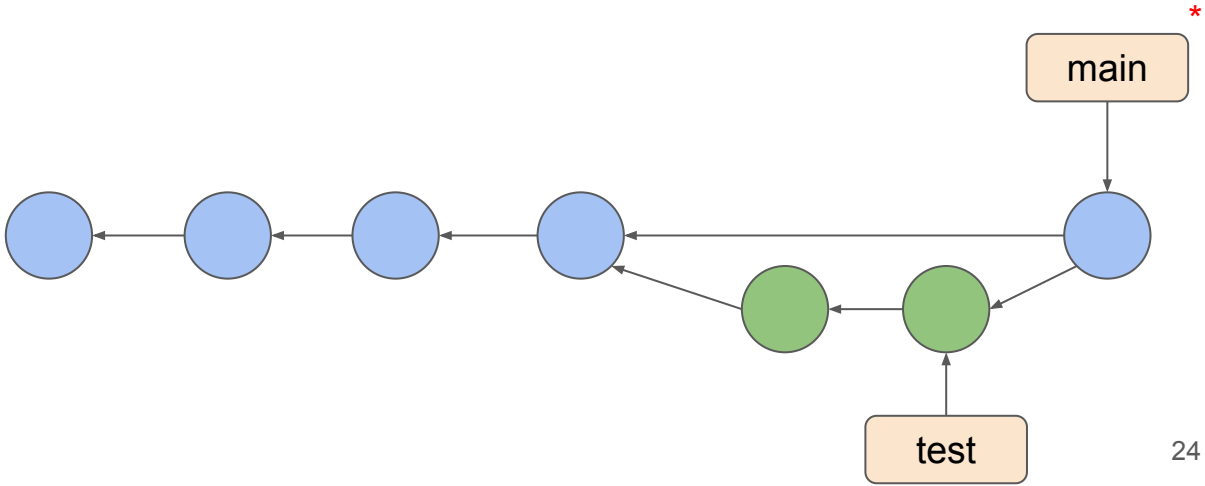
```
git checkout test
code two/file2.txt
git add two/
git commit -m “<msg>”
code one/file2.txt
git add one/
git commit -m “<msg>”
```



Git branches

- Merge test to master

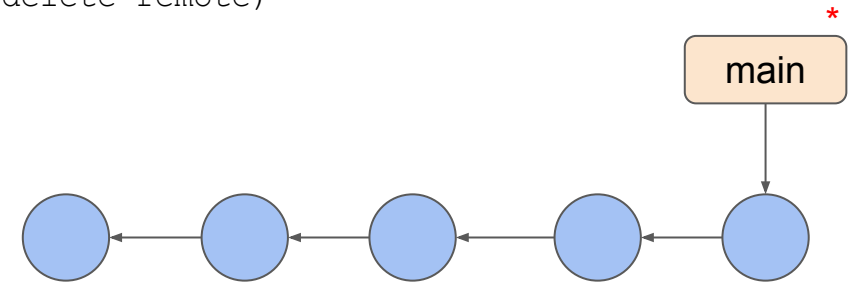
```
git checkout main
git merge test
```



Git branches

- Merge test to master

```
git checkout master
git merge test
git branch -d test (delete local)
git push -d origin test (delete remote)
```



Skip files from adding to the staging area

.gitignore is a special file that contains file patterns that skip when adding files to the staging area

https://github.com/github/gitignore/blob/main/Android.gitignore

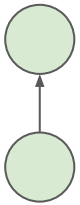
```
# Gradle files
.gradle/
build/

# Local configuration file (sdk path, etc)
local.properties
```

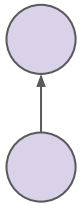
git rebase

Realistic scenario for rebasing

remote `main`

local `main`





https://www.youtube.com/watch?v=f1wnYdLEpgI&t=310s&ab_channel=TheModernCoder

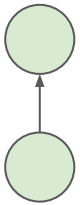
git rebase

Realistic scenario for rebasing

remote `main`

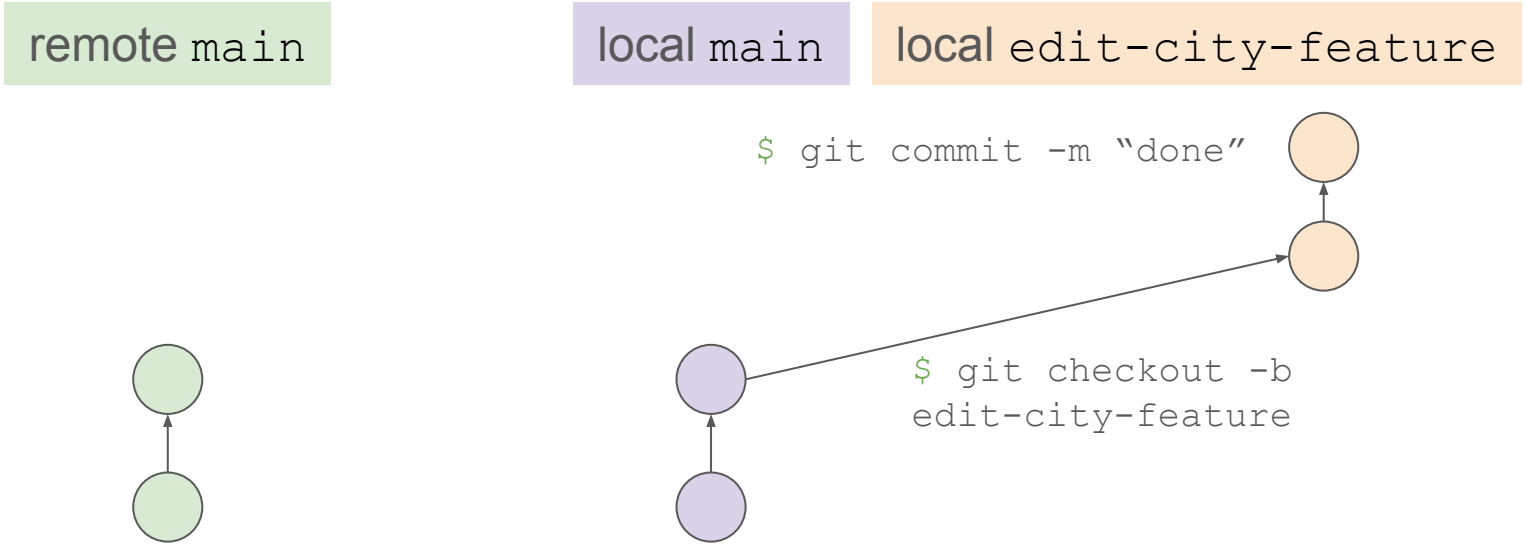
local `main`

local `edit-city-feature`



git rebase

Realistic scenario for rebasing

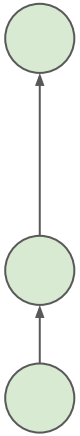


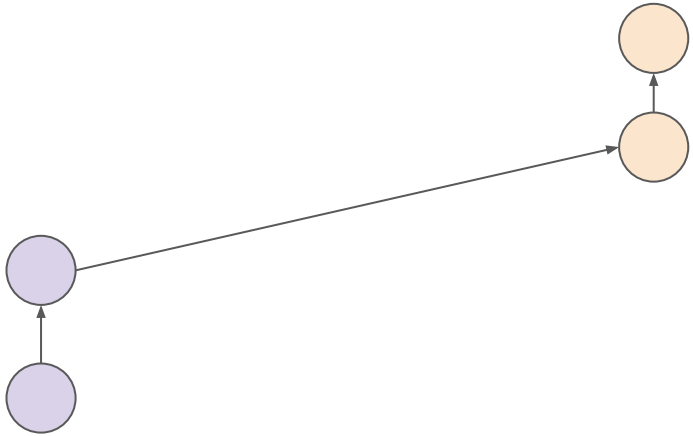
git rebase

remote `main`

local `main`

local `edit-city-feature`





```
$ git checkout main
```

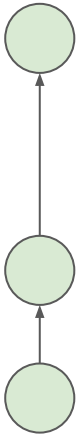
git rebase

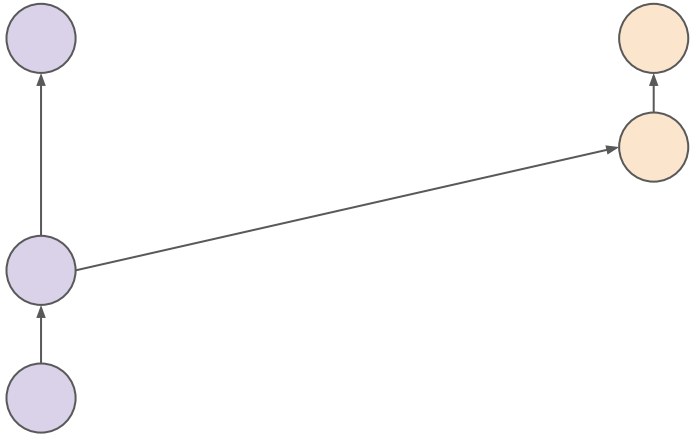
remote `main`

local `main`

local `edit-city-feature`

```
$ git pull origin main
```

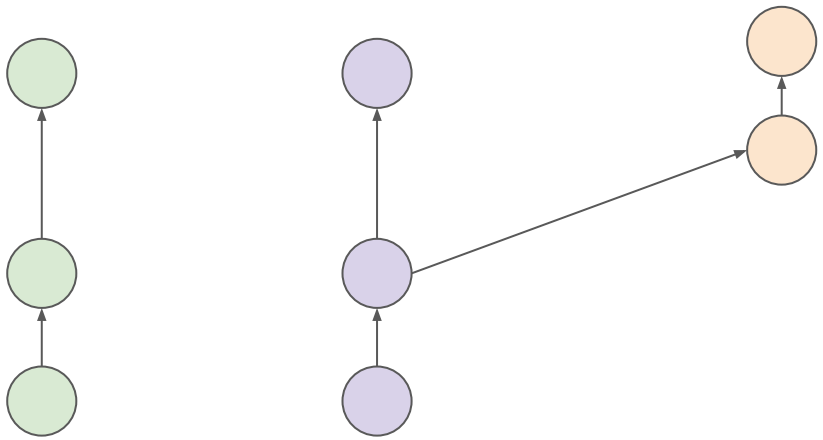




```
$ git checkout main
```

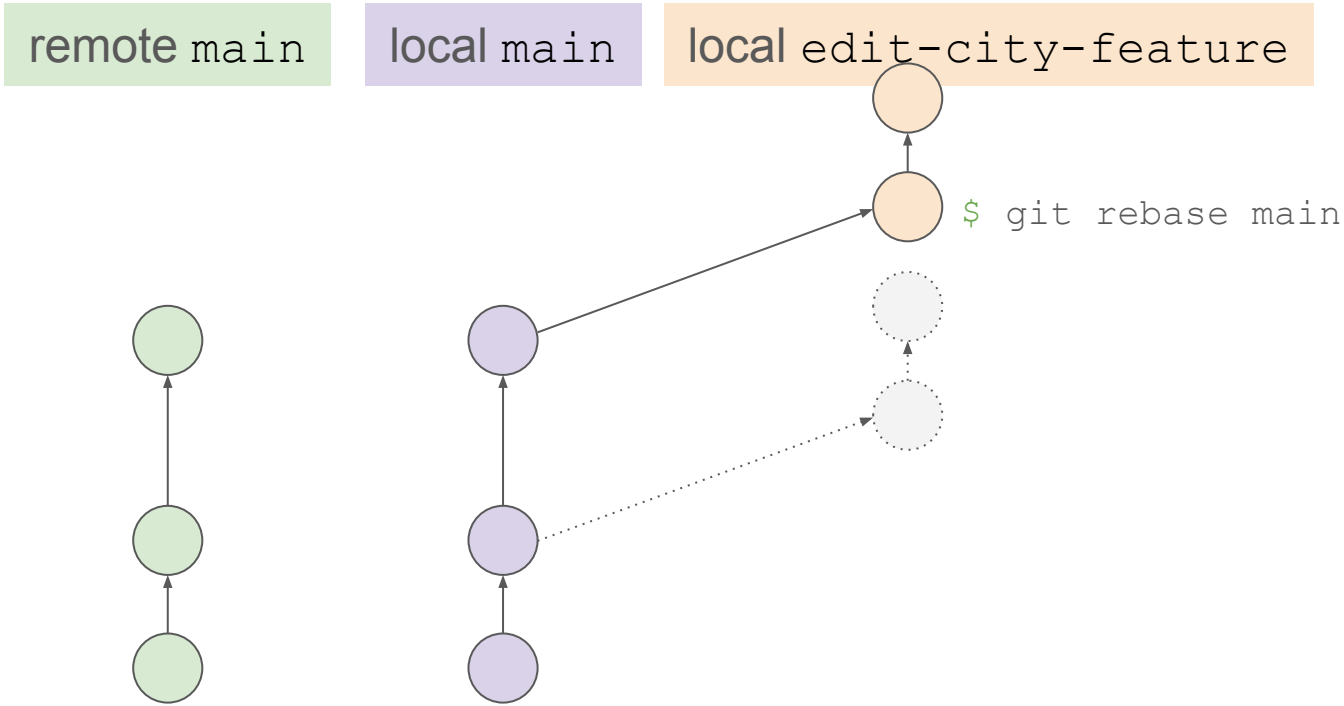
git rebase

remote `main` | local `main` | local `edit-city-feature`



```
$ git checkout edit-city-feature
```

git rebase

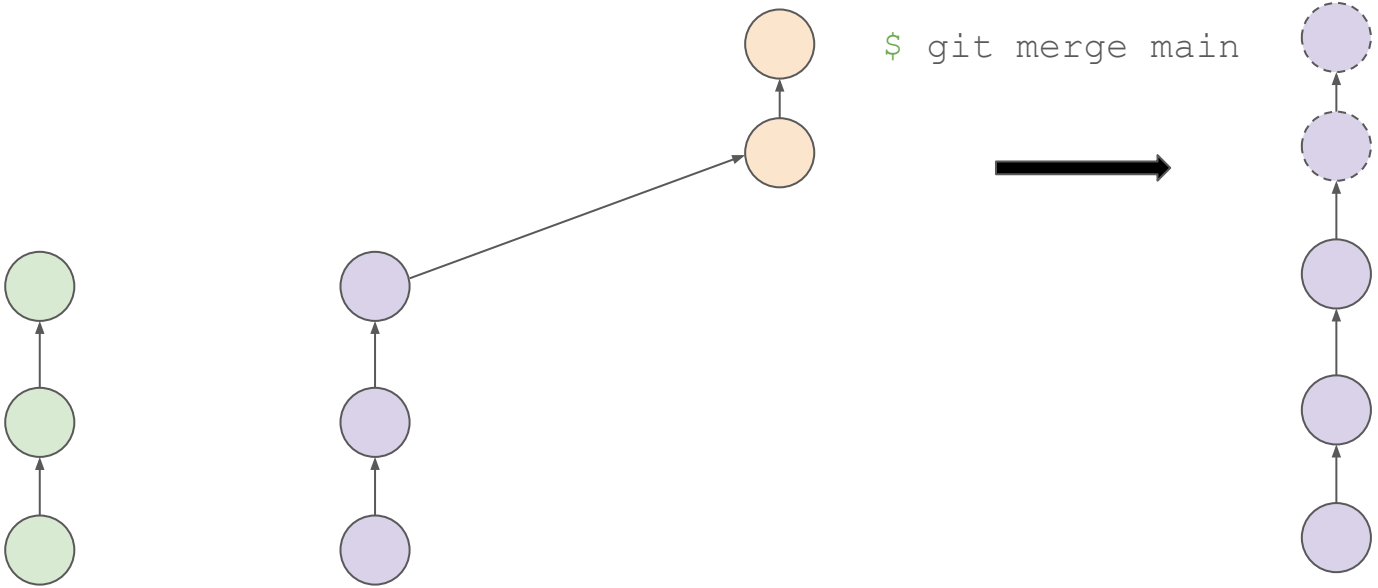

remote main
local main
local edit-city-feature
$ git rebase main

git rebase

remote `main`

local `main`

local `edit-city-feature`



git log

```
commit ea7fc24fcc3b4debee068534ef8069ae6730bc0e (HEAD -> main, origin/main)
Author: akalankag <boney.ag@gmail.com>
Date:   Thu Jan 9 01:01:24 2025 -0700

    make list select/deselect, hide/view delete button, delete city feature

commit 37ae3710d34549f3559806785b068b50d3dd3619
Author: akalankag <boney.ag@gmail.com>
Date:   Thu Jan 9 01:00:50 2025 -0700

    minor changes

commit 807fb568a5f0a457a09b265caed67e5c611b6f1c
Author: akalankag <boney.ag@gmail.com>
Date:   Thu Jan 9 01:00:08 2025 -0700

    add a new color code
```

git log

```
git log --oneline
```

```
ea7fc24 (HEAD -> main, origin/main) make list select/deselect, hide/view delete
button, delete city feature
37ae371 minor changes
807fb56 add a new color code
522d9c8 minor layout modifications
e637c43 edit city is done
0ab82e6 Create README.md
1c0d9a1 add logic and UI stuff to add a city
```

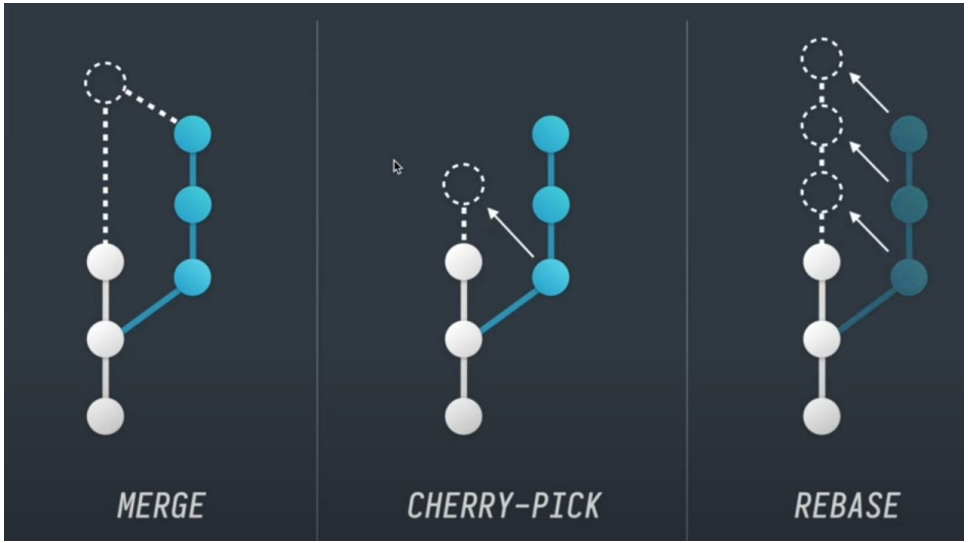
git log

```
*   commit 49779f27bbb51568d28acc1d517b62722252dc58 (HEAD -> main)
|\  Merge: f57c754 2a75e00
| | Author: akalankag <boney.ag@gmail.com>
| | Date:   Sun Jan 26 23:00:14 2025 -0700
| |
| |     Merge branch 'test'
| |
| * commit 2a75e00450a32ebed8f99f2656ca9a46706fb618 (test)
| | Author: akalankag <boney.ag@gmail.com>
| | Date:   Sun Jan 26 22:46:03 2025 -0700
| |
| |     branch demo
| |
| * commit de9c5c72f81528e5cecdbfafa5c2698ad9b114e9
| | Author: akalankag <boney.ag@gmail.com>
| | Date:   Sun Jan 26 22:42:58 2025 -0700
| |
| |     Another file
| |
* | commit f57c75451735d1e3e22f85d54b68d868ecc11a00
|/  Author: akalankag <boney.ag@gmail.com>
|   Date:   Sun Jan 26 22:47:36 2025 -0700
|
```

```
git log --graph
```

git cherry-pick

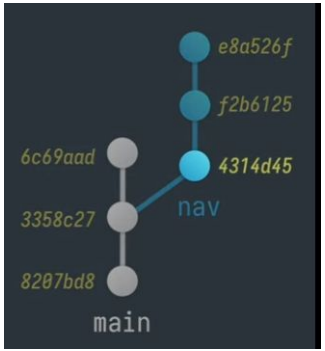
What if you only wanted to introduce specific commits from a different branch into your own branch? Cherry-pick helps!



git cherry-pick

Suppose you have two branches, branch2 and branch1

Say you want specific changes from branch2 in your own branch1, like a bug fix.




```
#this is to grab the git-hashes from the branch u want
the changes from
git log branch-2
git cherry-pick <grabbed-commit-hash>
```

You can also specify multiple git hashes to cherry-pick multiple commits as well.

Note that cherry-picking creates new commits. It is simply applying the specified commits one by one in the order of the hashes specified.

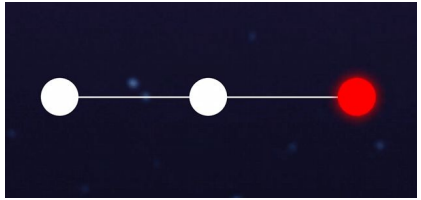
Can also give rise to merge-conflicts! You should know how to resolve them.

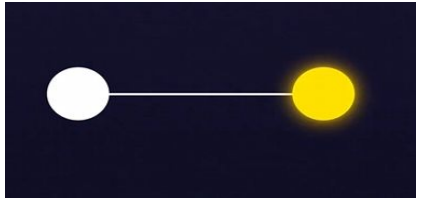
git reset

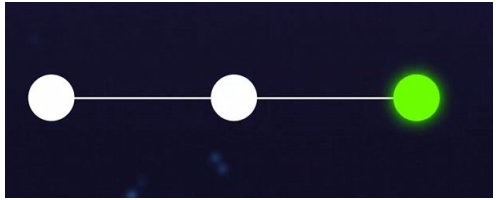
```
git reset <commit-hash>
```

Purpose of git reset:

Take a commit, undo it to go back to another commit in the past (while still keeping the changes since that reset command in the staging area), making more edits and making a new commit.







git reset --hard

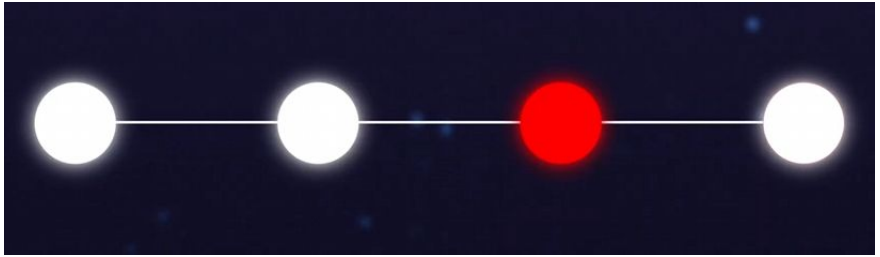
```
git reset --hard <commit-hash>
```

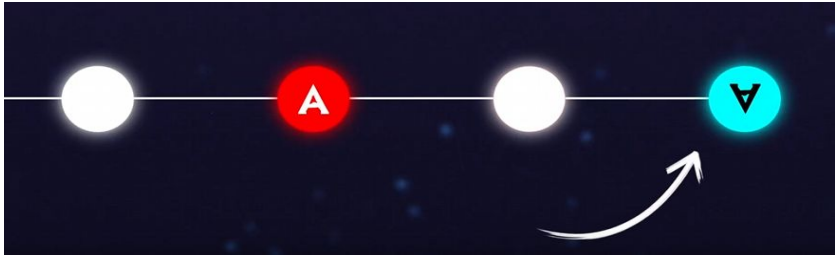
Using git reset with the `--hard` flag resets the HEAD of the git history to that particular commit hash, effectively “throwing away” the changes. This option does not keep any changes that have been made unlike the normal `git reset`, so use this with caution.

git revert

```
git revert <commit-hash>
```

Revert is used for when you would like to remove changes from specific commits while still keeping the rest of them intact. Revert makes a “revert commit” which essentially undoes the changes made, so it reverses effects without actually deleting from the original commit history and keeping it intact.



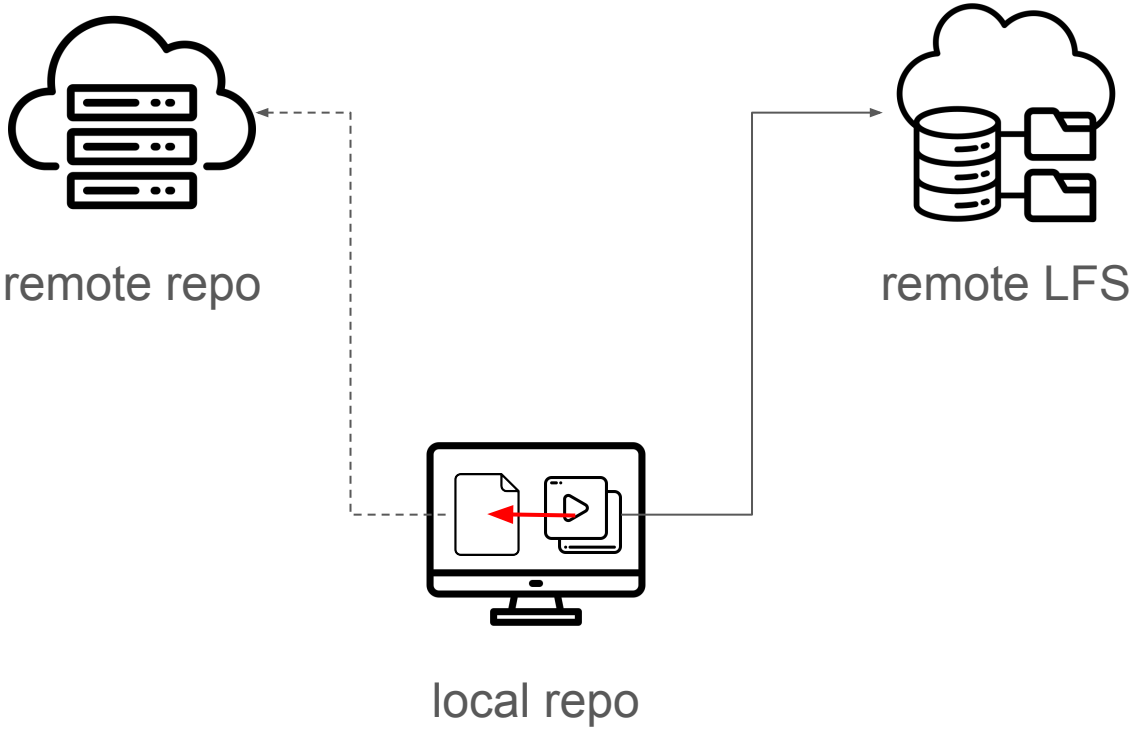


git revert

You can also revert multiple commits together. Here are some example revert commands below.

```
# This will create three separate revert commits, use non merge commits only:
-> git revert a867b4af 25eee4ca 0766c053

# It also takes ranges. This will revert the last two commits:
-> git revert HEAD~2..HEAD

# Similarly, you can revert a range of commits using commit hashes (non inclusive of first hash):
-> git revert 0d1d7fc..a867b4a

# Reverting a merge commit. You can also use a range of merge commits here.
-> git revert -m 1 <merge_commit_sha>
```

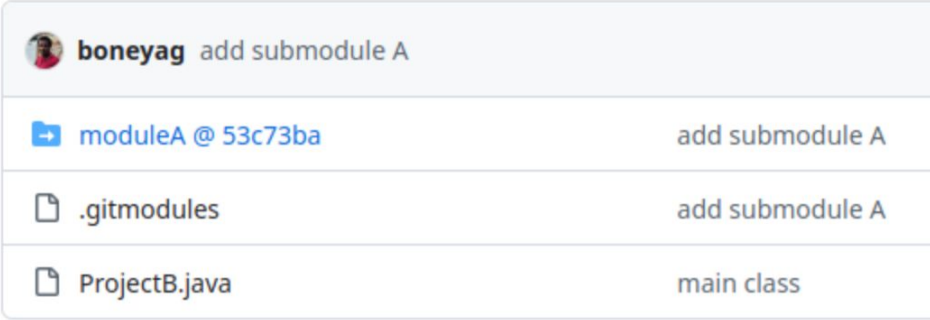
git LFS (Large File Storage)



https://www.flaticon.com/

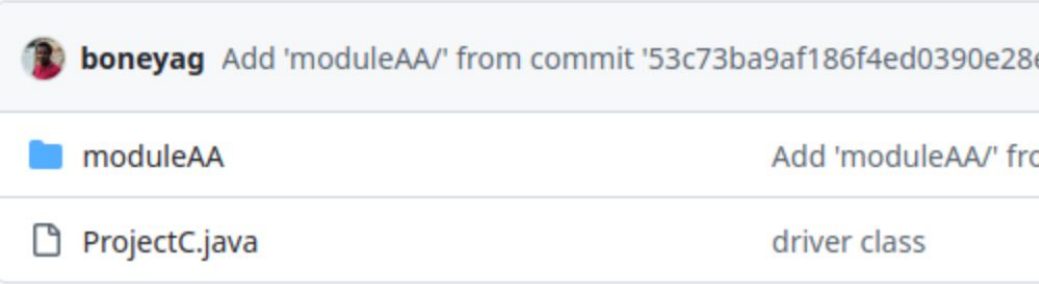
git submodules

```
$ mkdir projectB
$ cd projectB
$ git init
$ git branch -m main
$ code ProjectB.java
$ git add ProjectB.java
$ git commit -m "..."
$ git submodule add <git-url>
$ git add moduleA
$ git add .gitmodules
$ git commit -m "..."
```

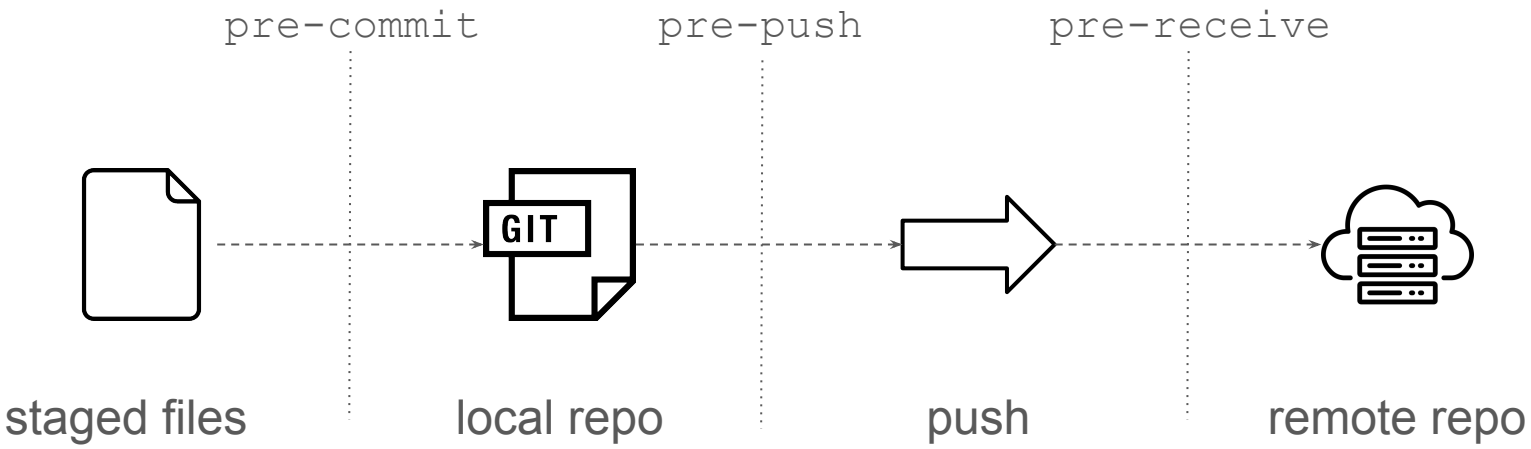


git subtree

```
$ mkdir projectC
$ cd projectC
$ git init
$ git branch -m main
$ code ProjectC.java
$ git add ProjectC.java
$ git commit -m "..."
$ git subtree add --prefix=moduleA <git-url-without-
.git> <branch-name>
```



git hooks


pre-commit
pre-push
pre-receive
GIT
staged files
local repo
push
remote repo

git gc (garbage collection)

- Clean up detached commits (resulted from `git reset`/`git rebase`)
- `git gc` will automatically run when `pull/commit/rebase/merge` is executed

git prune (child command of git gc)

- Not advised run this command
- For fun – try this
  a. Create a local git repo (git init), then add a couple of commits (git log should show your commits)
  b. git log --oneline
     sha1 (HEAD -> main) msg4
     sha2 msg3
     sha3 msg2
     sha4 msg1
  c. git reset --hard sha3
  d. git checkout sh2
  e. git checkout main
  f. git prune --dry-run --verbose