

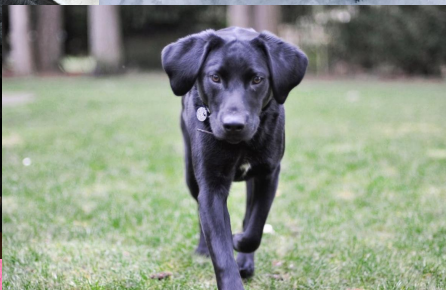


GIT II

07-131 Great Practical Ideas
By: Lucy and Deepti



PET TAX





GIT ReVIEW



WHAT IS GIT?

1.

version control

Keep track of previous work
Fix mistakes by reverting code

2.

centralized workflow

One main "copy" of the project
Clean way of keeping track

3.

collaboration

Work with others on projects
Merge code (branches) with others

4.

online

Examine others' projects (if public repo)
*** Git is NOT for storing and collaborating on projects online

BUT FIRST

**ADDING and
COMMITING**

```
$ git add .
```

```
$ git commit -m "message"
```


ALL THINGS BRANCHES

SWITCH

```
$ git checkout mybranch
```

create

```
$ git branch mybranch
```

← BOTH

```
$ git checkout -b mybranch
```

merge

```
$ git merge thebranch
```

DELETE

```
$ git branch -d thebranch
```

LIST BRANCHES

```
$ git branch
```




CODE



The basic commands of git

** side note: we can use `$ git status` to check the working tree of our repo

Everything else is the same as last week





GIT 2.0



UNDO ACTIONS

undo changes that weren't committed

```
git checkout <file_name>
```

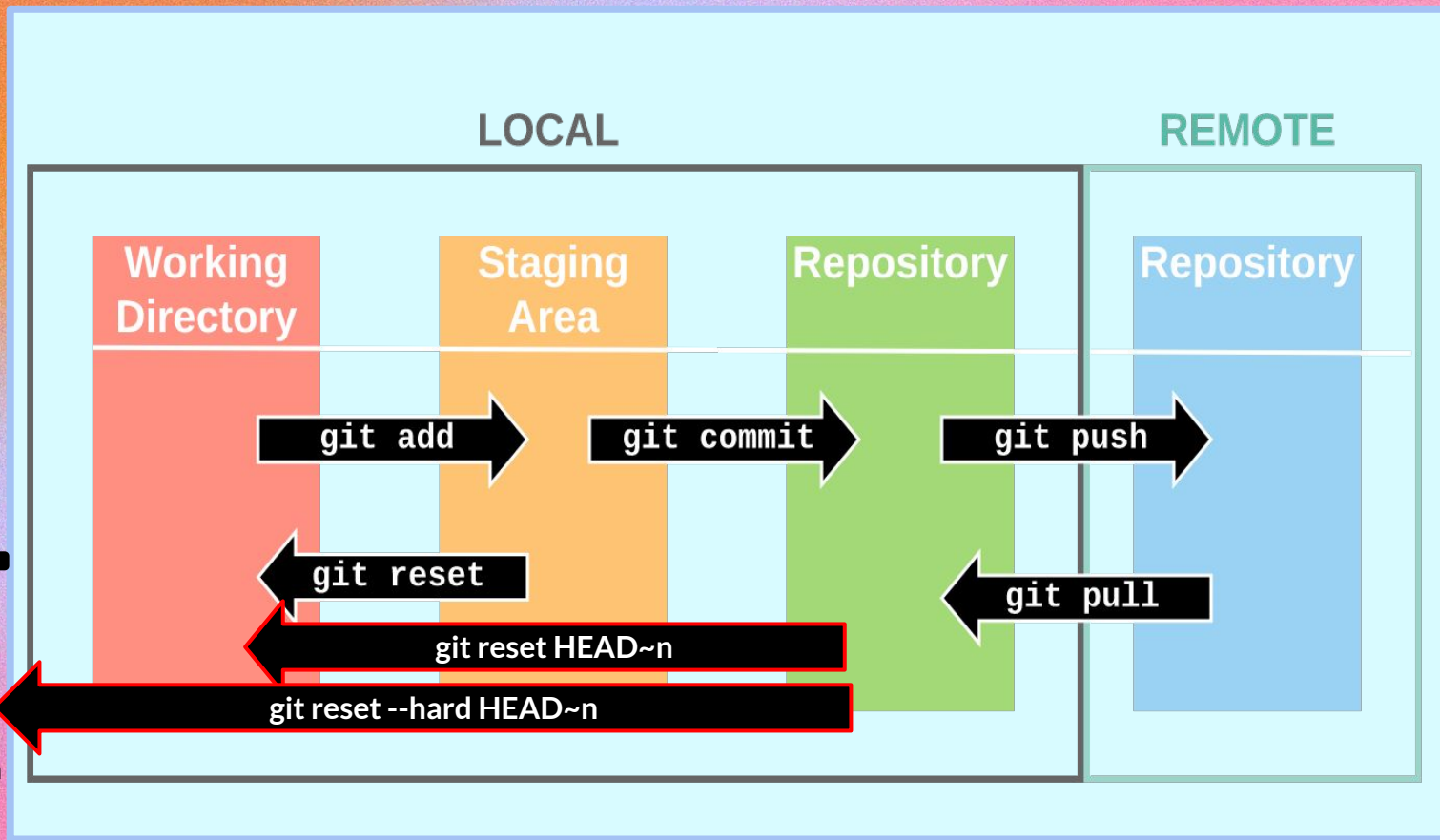
unstage changes (after git add, before git commit)

```
git reset HEAD <file_name> /  
git reset <file_name>
```

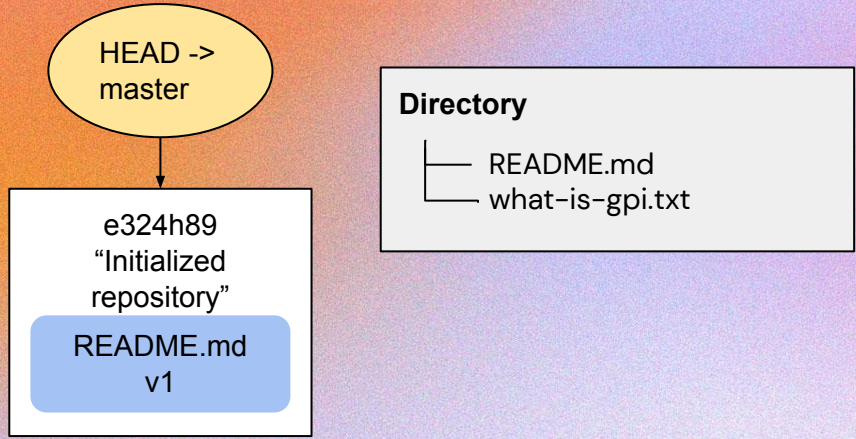
remove commits by moving the HEAD pointer to the commit specified:

```
git reset --hard HEAD~n (destructive)
```

```
git reset HEAD~n (changes from last n commits  
remain in working directory)
```

changes in last n
commits are
lost



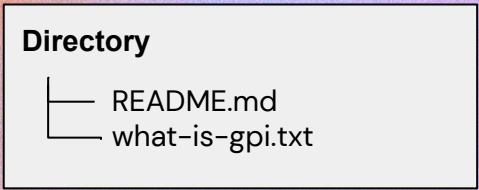
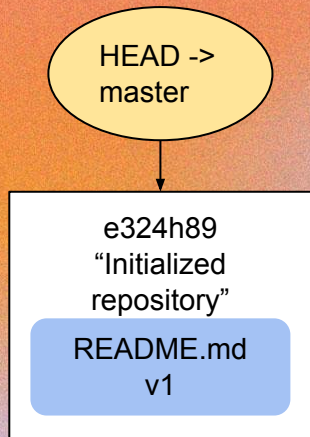
EXAMPLE 1

How to undo changes that haven't been committed?

```
$ vim what-is-gpi.txt
```

```
1 GPI is your favorite class
```

```
1 GPI is your class  
:wq
```

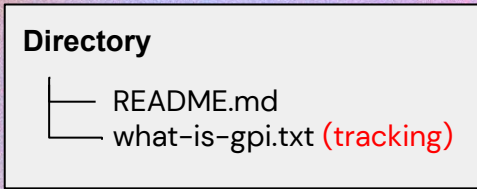
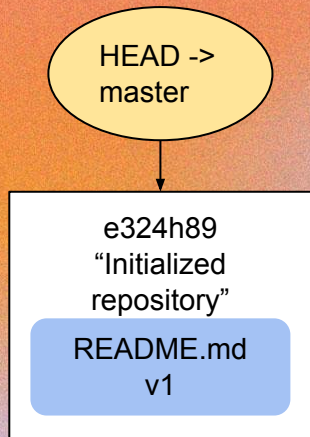
```
$ vim what-is-gpi.txt
```

```
1 GPI is your favorite  
class
```

EXAMPLE 1

How to undo changes that haven't been committed?

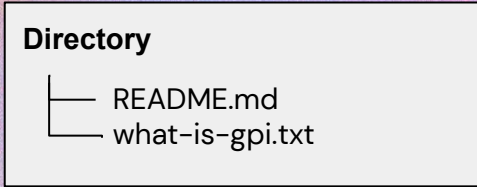
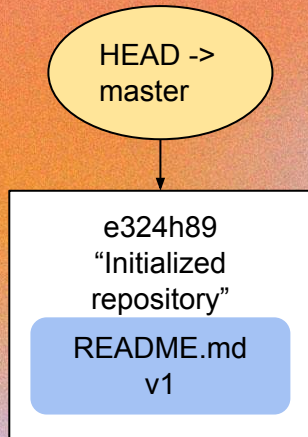
```
Answer: git checkout what-is-gpi.txt
```

```
dsunkara@Venkat-Dell-XPS:~/git-demo$ git add what-is-gpi.txt
dsunkara@Venkat-Dell-XPS:~/git-demo$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   what-is-gpi.txt
```

EXAMPLE 2

How to undo staging (tracking by git) of changes?



```
dsunkara@Venkat-Dell-XPS:~/git-demo$ git add what-is-gpi.txt
dsunkara@Venkat-Dell-XPS:~/git-demo$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   what-is-gpi.txt
```

EXAMPLE 2

How to undo staging (tracking by git) of changes?

Answer:

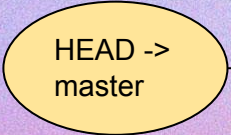
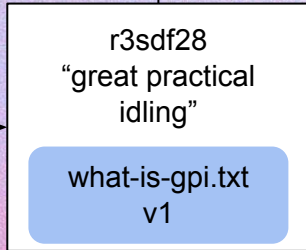
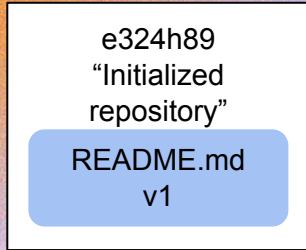
```
git reset what-is-gpi.txt
```

```
dsunkara@Venkat-Dell-XPS:~/git-demo$ git reset HEAD what-is-gpi.txt
dsunkara@Venkat-Dell-XPS:~/git-demo$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    what-is-gpi.txt

nothing added to commit but untracked files present (use "git add" to track)
```

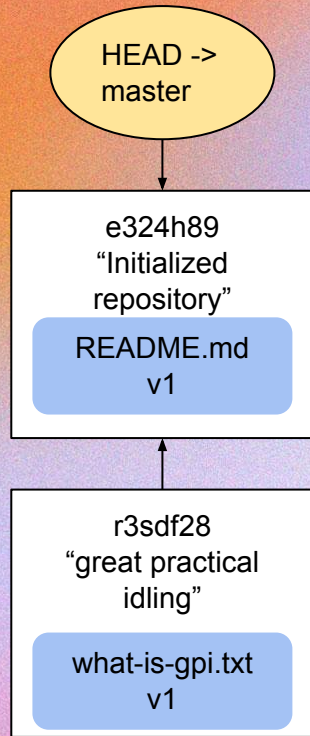

Directory

```
├── README.md
└── what-is-gpi.txt
```



EXAMPLE 3

How to undo a commit without losing changes?



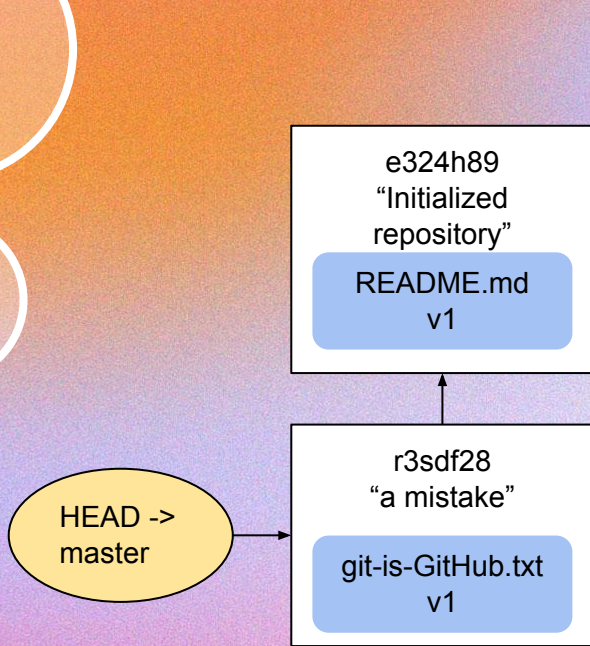
Directory

```
├── README.md  
└── what-is-gpi.txt
```

EXAMPLE 3

How to undo a commit without losing changes?

Answer: `git reset HEAD~1`

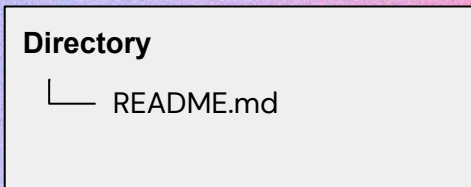
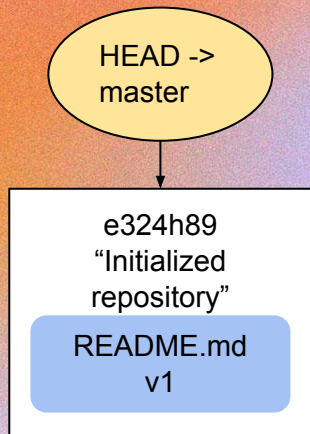


Directory

```
├── README.md  
└── git-is-GitHub.txt
```

EXAMPLE 4

How to undo a commit while
obliterating changes?



EXAMPLE 4

How to undo a commit while
obliterating changes?

Answer: `git reset --hard HEAD~1`

GIT STASH

Store **staged/tracked** changes since last commit
(will remove from working directory):

```
$ git stash
```

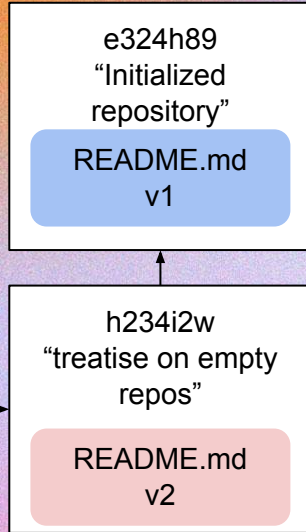
Seeing the stash:

```
git stash list
```

Retrieving changes from stash (will now appear in
working directory and as staged changes):

```
git stash apply stash@{n}
```


HEAD ->
master

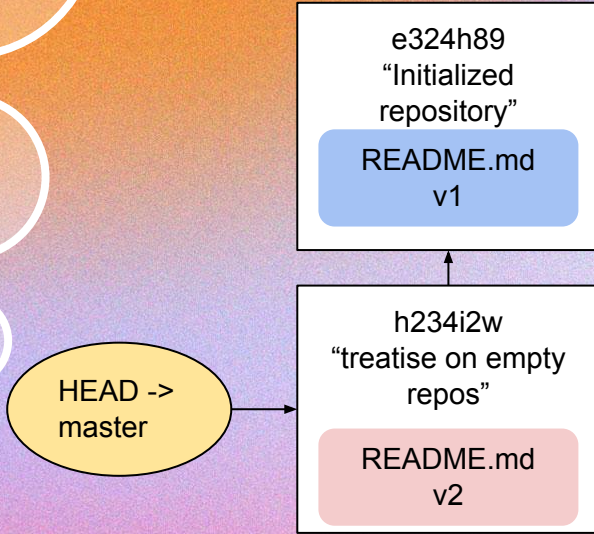


Directory

```
├── README.md  
└── how-to-stash.txt
```

EXAMPLE

How to commit changes to new file `how-to-stash.txt` after first commit (`e324h89`)? There should be no trace of the last commit. [Hint: 5 steps]



Directory

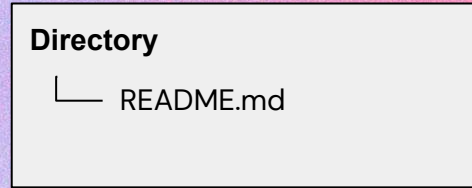
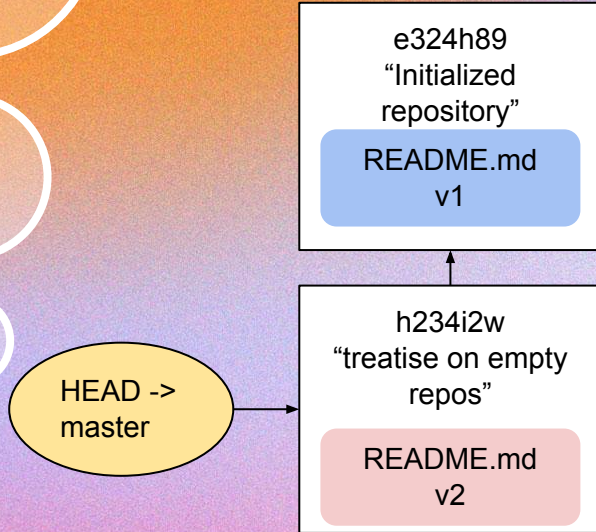
```
├── README.md
└── how-to-stash.txt (tracking)
```

EXAMPLE

How to commit changes to new file `how-to-stash.txt` after first commit (`e324h89`)? There should be no trace of the last commit. [Hint: 5 steps]

Answer:

1. `git add how-to-stash.txt`

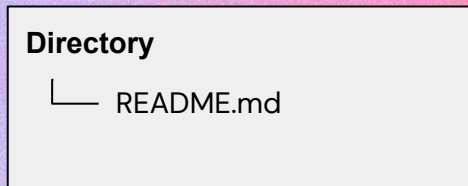
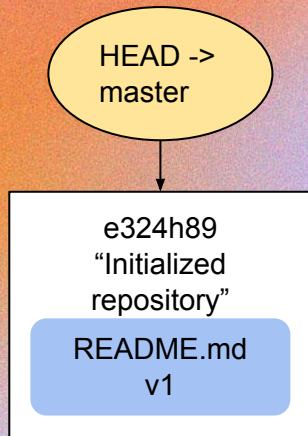


EXAMPLE

How to commit changes to new file `how-to-stash.txt` after first commit (`e324h89`)? There should be no trace of the last commit. [Hint: 5 steps]

Answer:

1. `git add how-to-stash.txt`
2. `git stash`

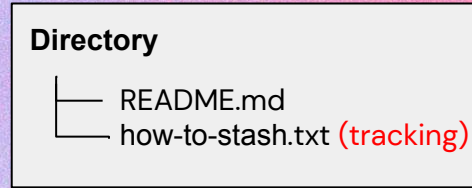
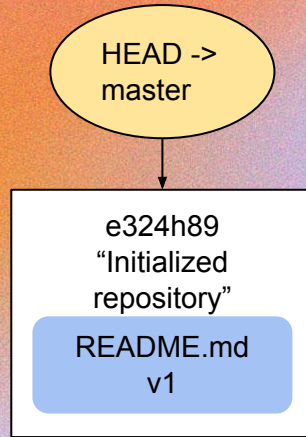


EXAMPLE

How to commit changes to new file `how-to-stash.txt` after first commit (`e324h89`)? There should be no trace of the last commit. [Hint: 5 steps]

Answer:

1. `git add how-to-stash.txt`
2. `git stash`
3. `git reset --hard HEAD~1`

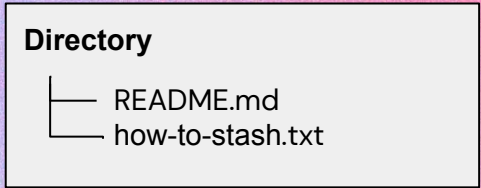
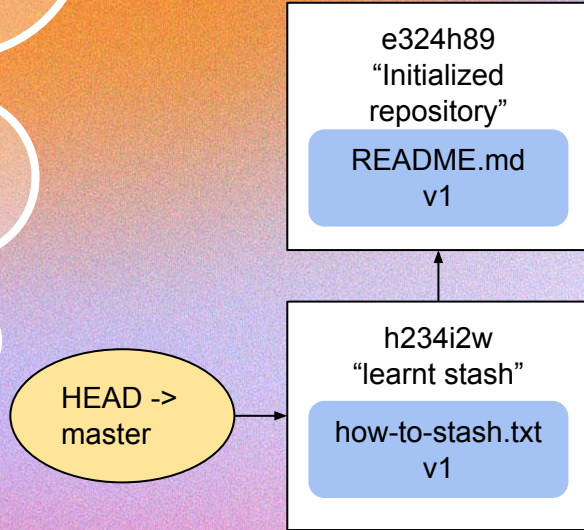


EXAMPLE

How to commit changes to new file how-to-stash.txt after first commit (e324h89)? There should be no trace of the last commit. [Hint: 5 steps]

Answer:

1. `git add how-to-stash.txt`
2. `git stash`
3. `git reset --hard HEAD~1`
4. `git stash apply stash`



EXAMPLE

How to commit changes to new file `how-to-stash.txt` after first commit (`e324h89`)? There should be no trace of the last commit. [Hint: 5 steps]

Answer:

- `git add how-to-stash.txt`
- `git stash`
- `git reset --hard HEAD~1`
- `git stash apply stash`
- `git commit -m "learnt stash"`

reverting

- To check our commit history
\$ git log
- To revert a previous commit
\$ git revert commit-hash
- Different from reset!
 - Makes a new commit for undone changes (maintains existing commit history)
 - Does not require you to undo all past commits until the wanted commit

```
linmo@linux-14:~/private/gpi-interview/gpi-demo2$ git log
commit 190e4215964d548d39d4e9f7c80faf2fb36a0bb8 (HEAD -> master)
Author: Lin Mo <linmo@linux-14.andrew.cmu.edu>
Date:   Mon May 23 20:16:24 2022 -0400

    added succulent

commit 0b36cc8ffc29ced2b9d825aa3760875941cee6fc
Author: Lin Mo <linmo@linux-14.andrew.cmu.edu>
Date:   Mon May 23 18:56:31 2022 -0400

    This is an informative message
linmo@linux-14:~/private/gpi-interview/gpi-demo2$ ls
a-cactus.txt  a-succulent.py
linmo@linux-14:~/private/gpi-interview/gpi-demo2$ git revert 0b36cc8ffc29ced2b9d825aa3760875941cee6fc
Removing a-cactus.txt
[master 96c6d29] Revert "This is an informative message"
  Committer: Lin Mo <linmo@linux-14.andrew.cmu.edu>
  Your name and email address were configured automatically based
  on your username and hostname. Please check that they are accurate.
  You can suppress this message by setting them explicitly. Run the
  following command and follow the instructions in your editor to edit
  your configuration file:

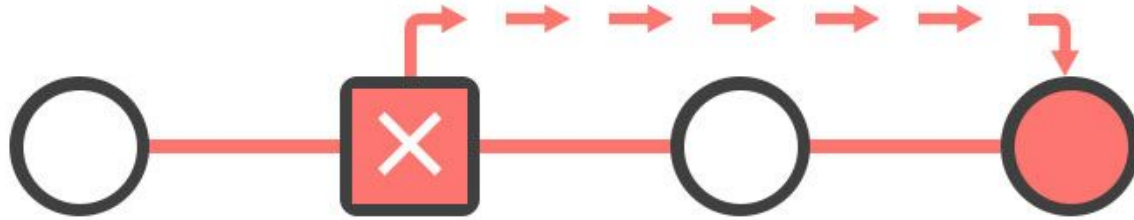
    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

1 file changed, 0 insertions(+), 0 deletions(-)
delete mode 100644 a-cactus.txt
linmo@linux-14:~/private/gpi-interview/gpi-demo2$ ls
a-succulent.py
```


Reverting



Resetting

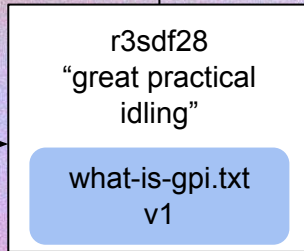
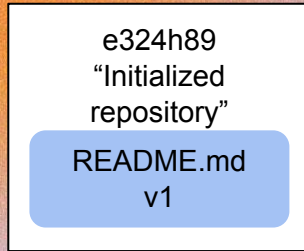


EXAMPLE

How to revert a commit?

Directory

```
├── README.md
└── what-is-gpi.txt
```



```
dsunkara@Venkat-Dell-XPS:~/git-demo$ git log
commit fd1d435c822a56c32cdc5c71be856c8992b5d77e (HEAD -> master)
Author: Deepti Sunkara <dsunkara@andrew.cmu.edu>
Date: Sun Oct 9 22:20:57 2022 -0400

    great practical idling

commit 011cd85286f9959b335572b195dc3c65d779daea
Author: Deepti Sunkara <dsunkara@andrew.cmu.edu>
Date: Sun Oct 9 21:08:59 2022 -0400

    initialized repo
```

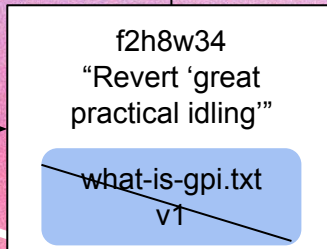
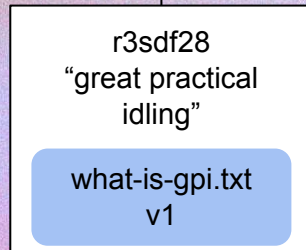
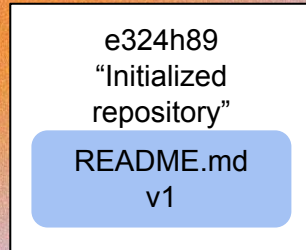

EXAMPLE

How to revert a commit?

Answer: `git revert r3sdf28`

Directory

└─ README.md



HEAD -> master

```
dsunkara@Venkat-Dell-XPS:~/git-demo$ git revert fd1d435c822a56c32cdc5c71be856c8992b5d77e
[master e6840c2] Revert "great practical idling"
 1 file changed, 0 insertions(+), 0 deletions(-)
 delete mode 100644 what-is-gpi.txt
dsunkara@Venkat-Dell-XPS:~/git-demo$ git log
commit e6840c2a9a84b83a86229160ec4cca32493e8d7f (HEAD -> master)
Author: Deepti Sunkara <dsunkara@andrew.cmu.edu>
Date: Sun Oct 9 22:21:39 2022 -0400

    Revert "great practical idling"

    This reverts commit fd1d435c822a56c32cdc5c71be856c8992b5d77e.

commit fd1d435c822a56c32cdc5c71be856c8992b5d77e
Author: Deepti Sunkara <dsunkara@andrew.cmu.edu>
Date: Sun Oct 9 22:20:57 2022 -0400

    great practical idling

commit 011cd85286f9959b335572b195dc3c65d779daea
Author: Deepti Sunkara <dsunkara@andrew.cmu.edu>
Date: Sun Oct 9 21:08:59 2022 -0400

    initialized repo
```


rebase vs. merge

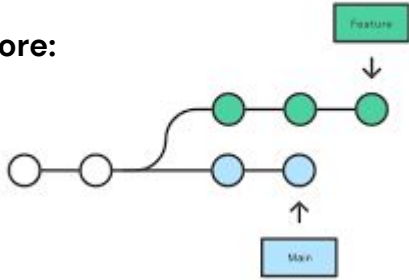
rebase

- moves branch to the tip of another
- not as safe as merge
- **please don't do this on public branches**

merge

- new commit to "merge" changes into the branch
- usually safer
- does not change existing branches

Before:



rebase

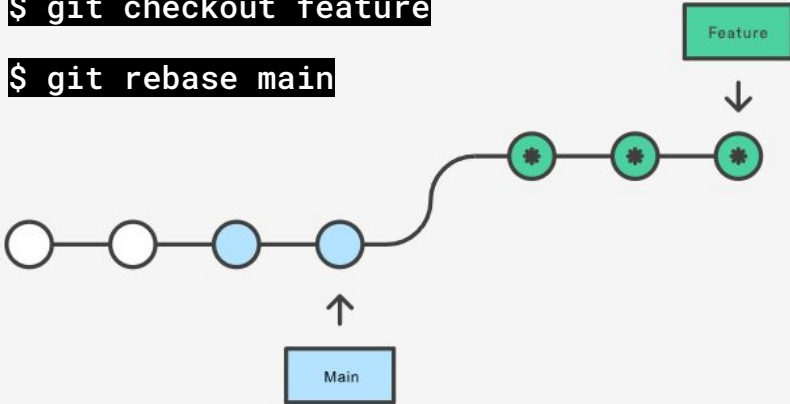
```
git rebase <branch_to_base_on>
```

Rebasing the feature branch onto main

After rebase:

```
$ git checkout feature
```

```
$ git rebase main
```



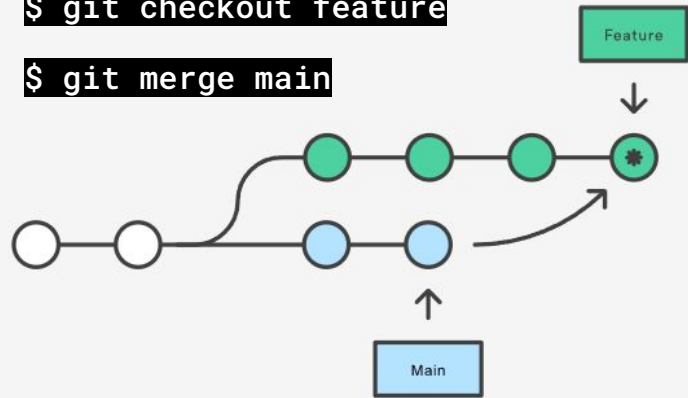
★ Brand New Commit

Merging main into the feature branch

After merge:

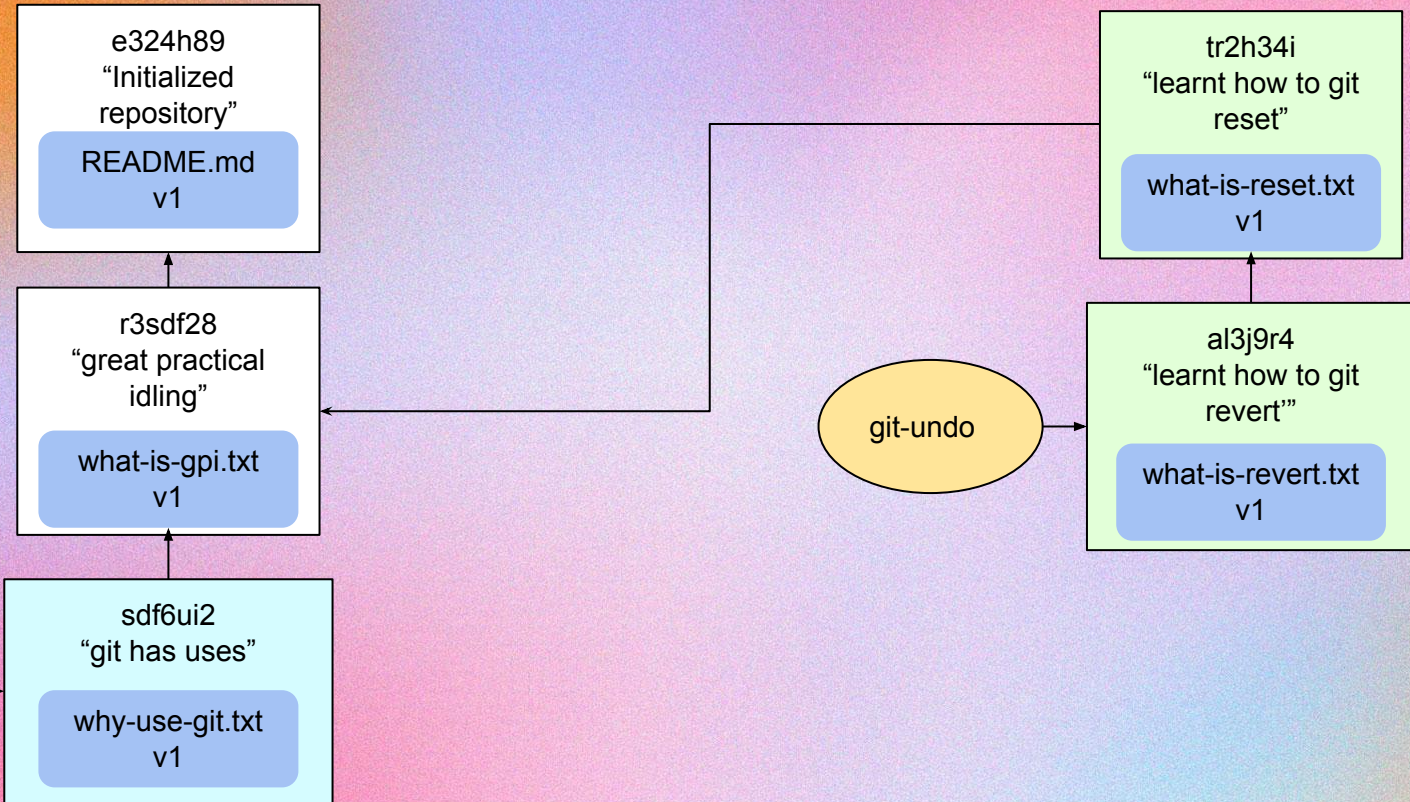
```
$ git checkout feature
```

```
$ git merge main
```

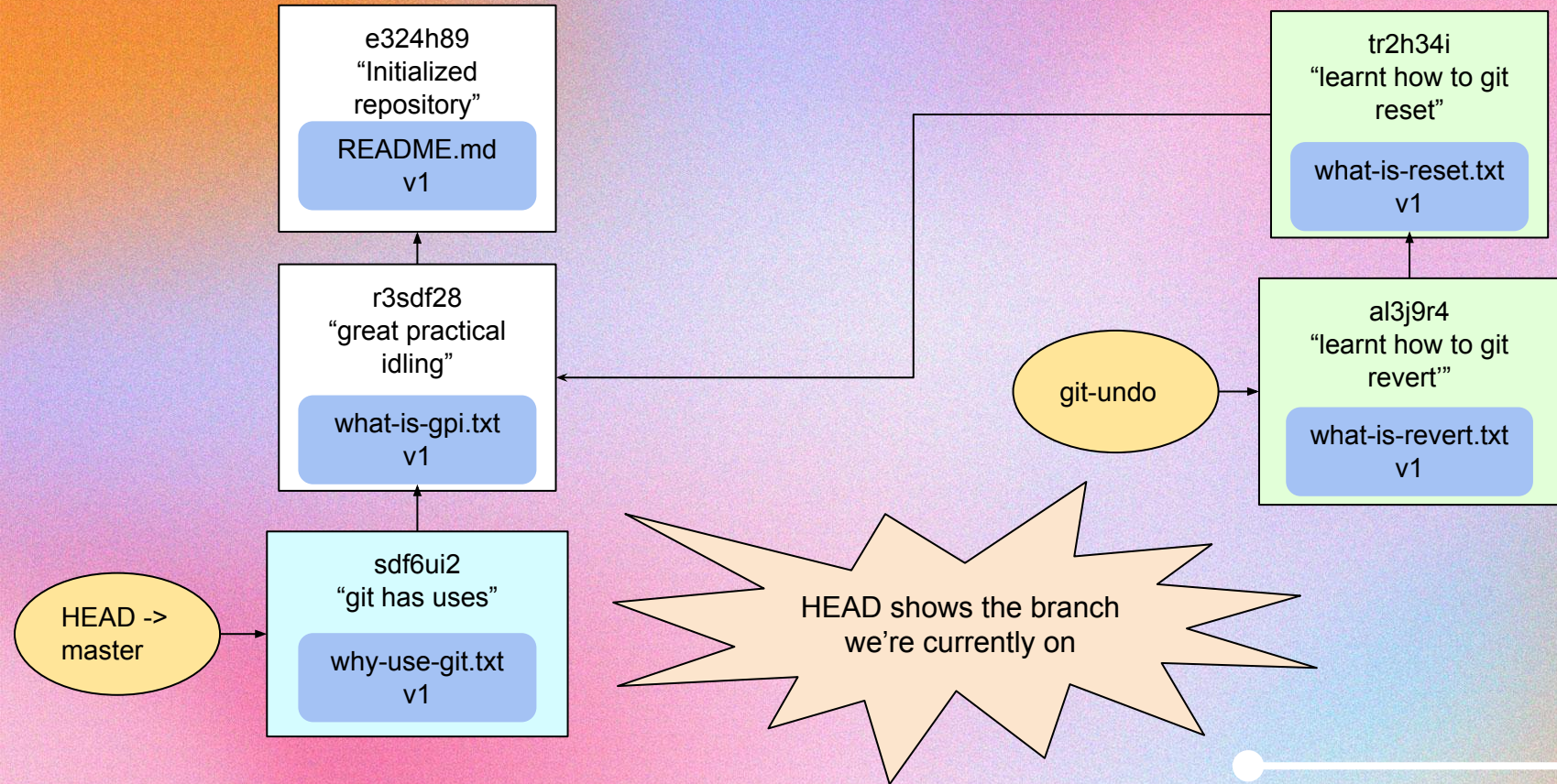


★ Merge Commit

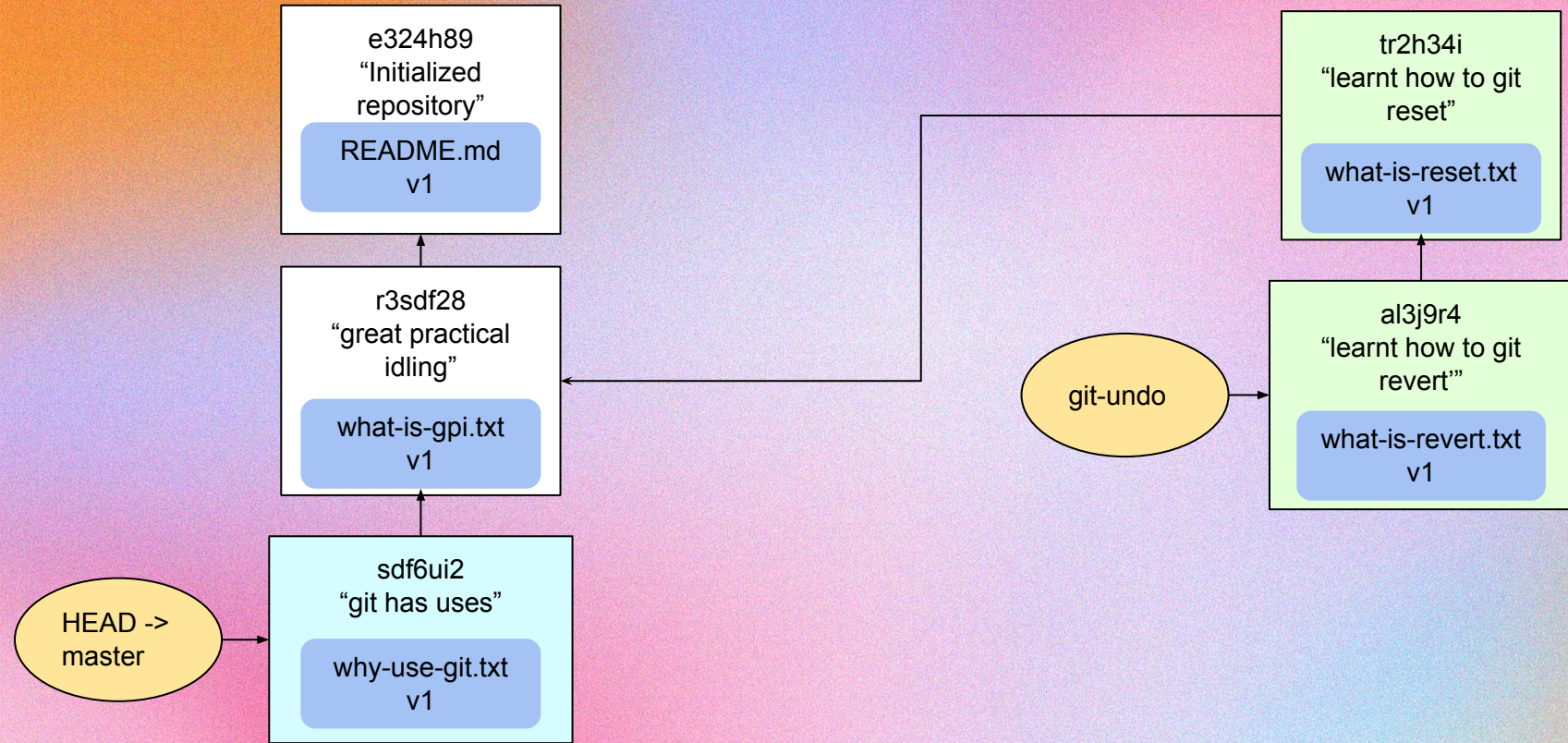
EXAMPLE 1 - ORIGINAL



EXAMPLE 1 - ORIGINAL

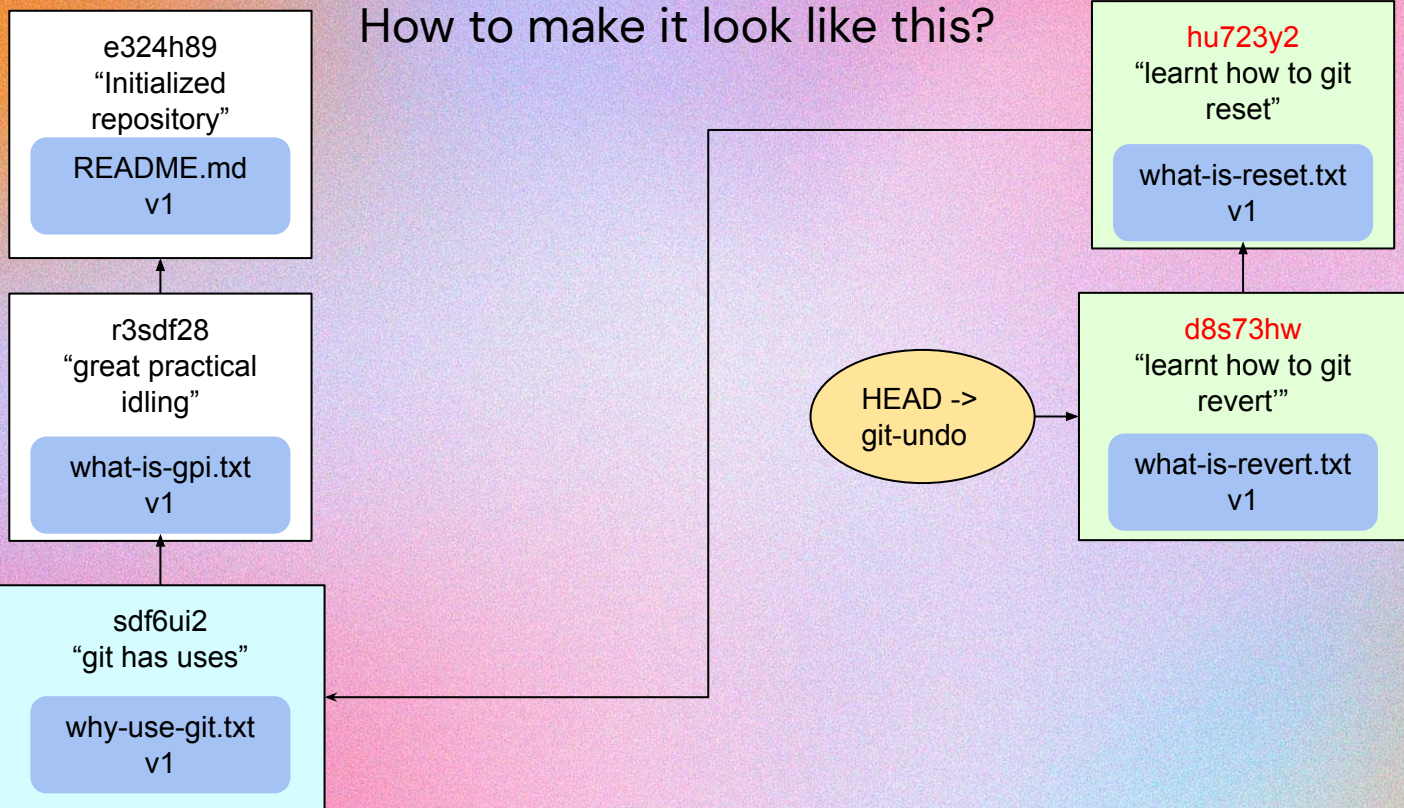


EXAMPLE 1 - ORIGINAL

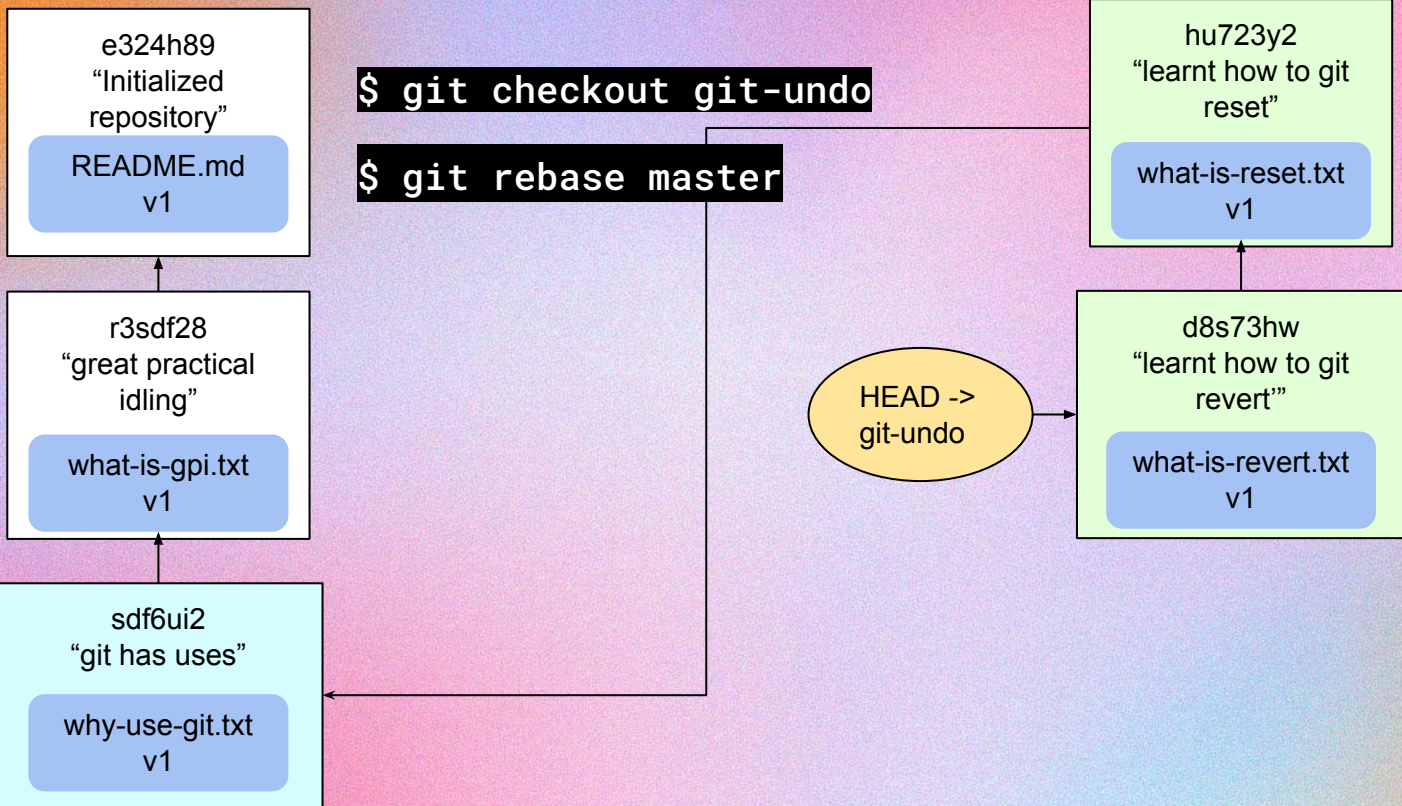


EXAMPLE 1

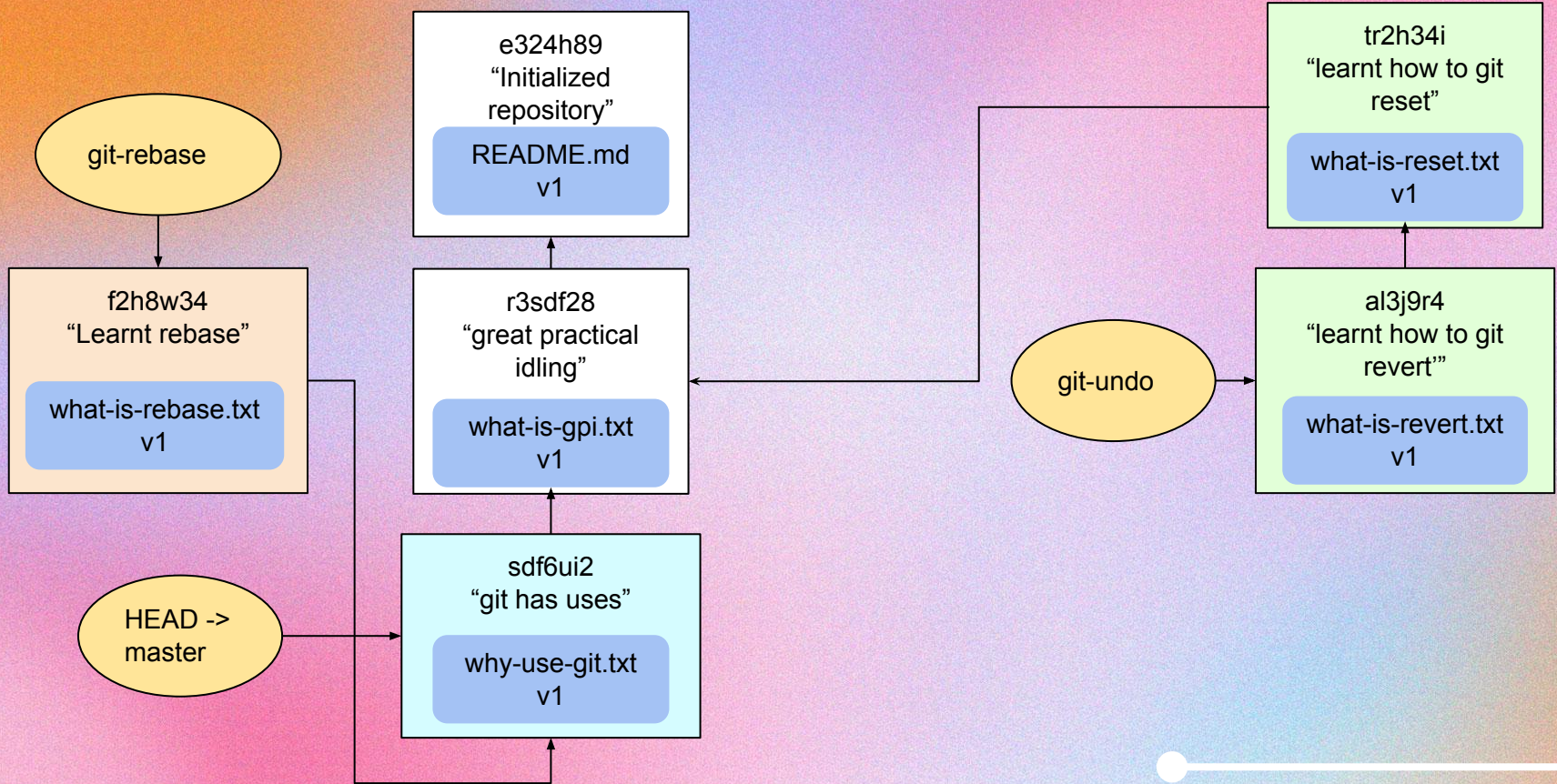
How to make it look like this?



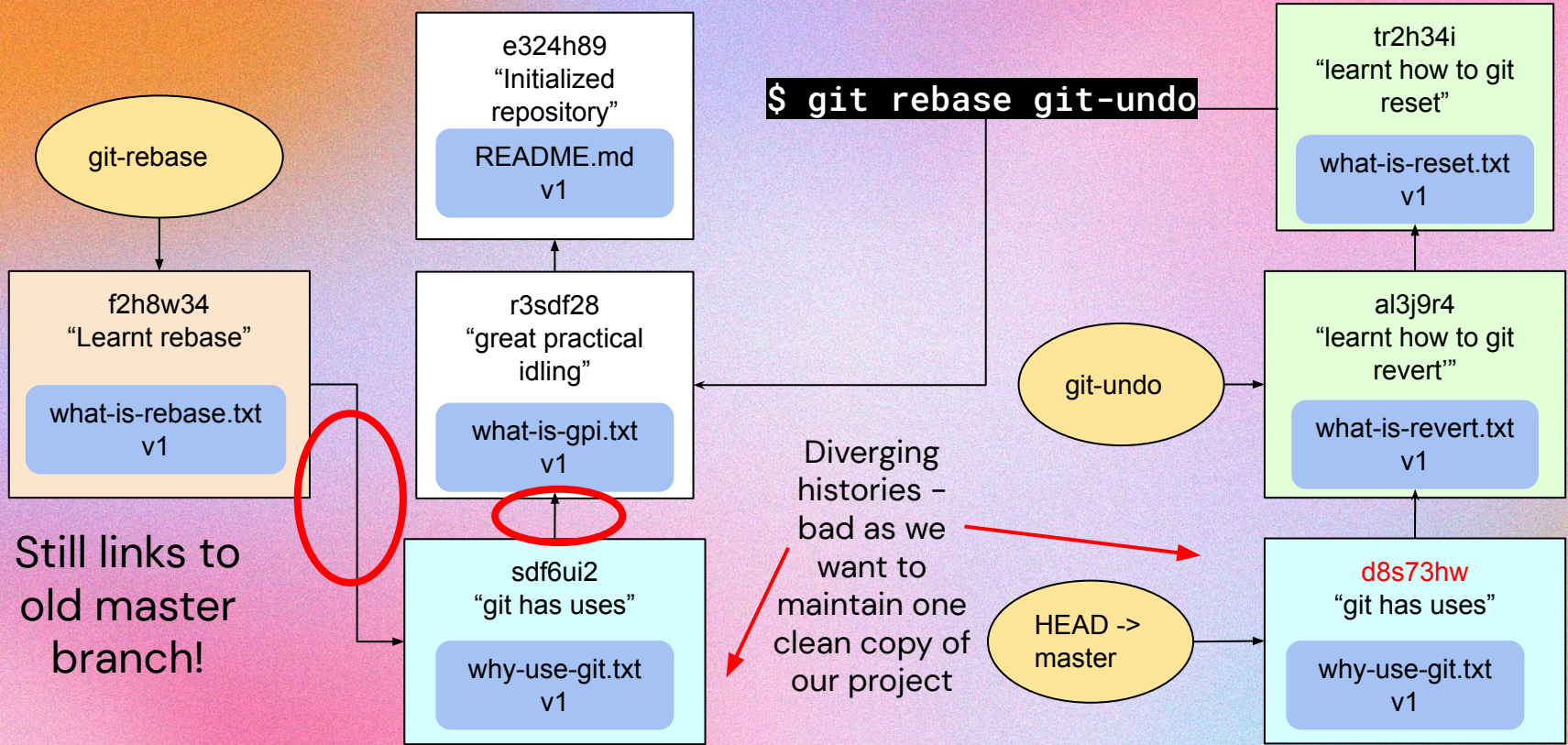
EXAMPLE 1 - rebase (GOOD)



EXAMPLE 2 - ORIGINAL

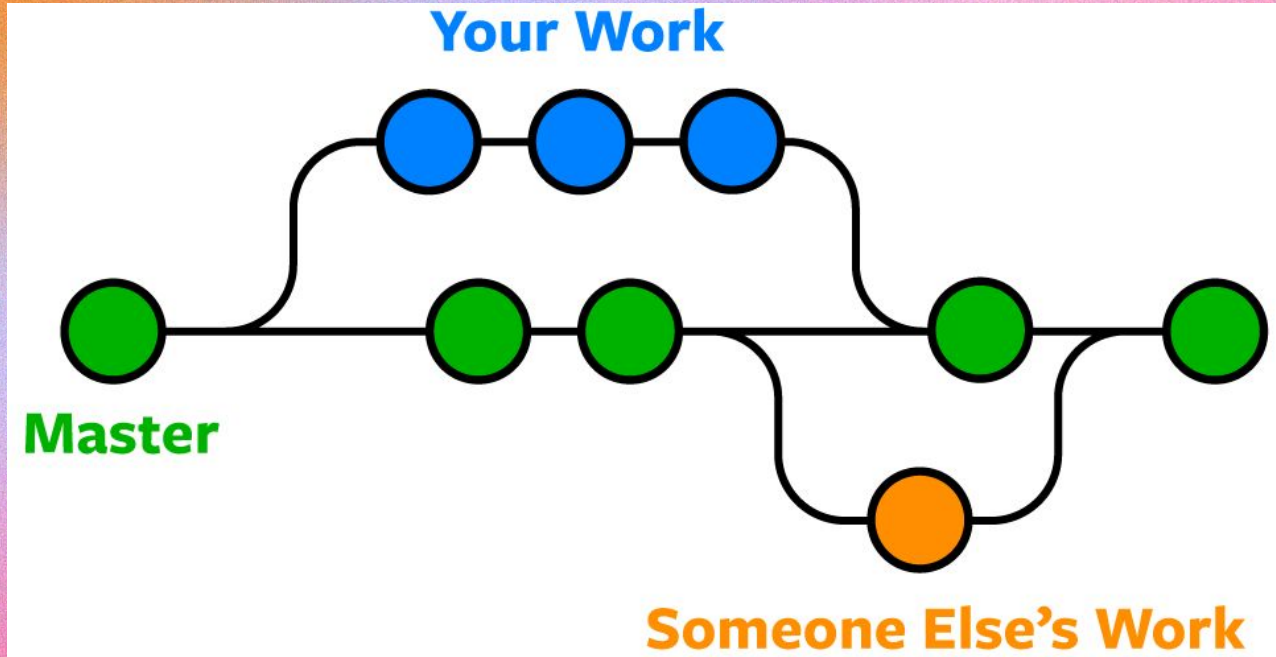


EXAMPLE 2 - rebase (BAD)



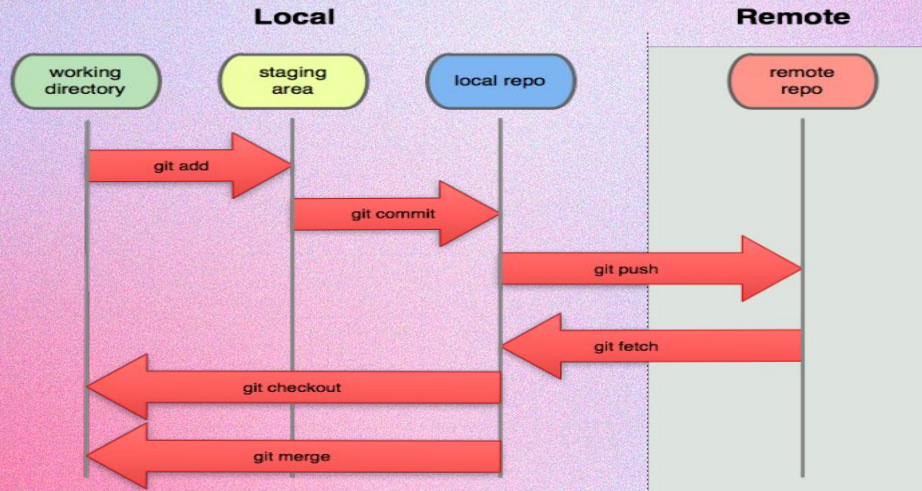
COLLABORATION

BRANCHES? LIKE A TREE?



remotes

- These are basically copies of your repo stored in the cloud
- **Default name: origin**
- This enables collaboration, but you have to update it!!!
- Drawback: Hard to be consistent across versions





MULTIPLE REMOTES

You can have multiple remotes!

To check existing remotes:

```
$ git remote
```

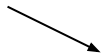
The default name of a cloned remote is origin.

The `-v` flag shows the URLs linked to the remotes.

```
$ git remote add <name> <url>
```



creating a project



git



TWO ways TO access GITHUB

GITHUB DESKTOP

Desktop app that's easy to use,
but there is limited
functionality

Terminal / COMMAND PROMPT

Great flexibility and speed
after knowing just a few
commands



Search or jump to...



[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)

[Overview](#) [Repositories 31](#) [Projects](#) [Packages](#) [Stars 1](#)



Lucy Mo
Lululucyyyyyy

web development | machine learning
CMU SCS '25

[Edit profile](#)

15 followers · 22 following

[XdHacks Mini](#)
[Vancouver, Canada](#)
lululucyyyyyy@gmail.com
www.lululucyyyyyy.com

Achievements



Find a repository...

Type ▾

Language ▾

Sort ▾

[New](#)

portfolio Public

[Star](#) ▾

CSS Updated on Mar 26

git-practice Public

[Star](#) ▾

Forked from cmugpi/git-practice

736 Updated on Oct 21, 2021

kevin Private

[Star](#) ▾

JavaScript Updated on Aug 6, 2021

NLP_course1 Public

[Star](#) ▾

on coursera, the assignments

Jupyter Notebook Updated on Jun 20, 2021

lucyswebsite Private

[Star](#) ▾

Lucy's website

JavaScript Updated on May 26, 2021

Fill in corresponding information


It is good practice to add a README.md to tell users about your project and how to run it

This is in markdown

Create a new repository


A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)


Owner * **Repository name ***

 Lulucyyyyyyy ▾ /

Great repository names are short and memorable. Need inspiration? How about [didactic-lamp?](#)

Description (optional)

 **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.](#)

Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

ⓘ You are creating a public repository in your personal account.



**USE PRIVATE
REPOS TO
FOLLOW
ACADEMIC INTEGRITY**



**USE PRIVATE
REPOS TO
HIDE HOW
BAD YOUR CODE IS**

Lululucyyyyyy / **gpi-demonstration** Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags

Go to file Add file - Code -

Lululucyyyyyy Initial commit

README.md Initial commit

README.md

gpi-demonstration

Clone

HTTPS SSH GitHub CLI

https://github.com/Lululucyyyyyy/gpi-

Use Git or checkout with SVN using the web URL.

Open with GitHub Desktop

Download ZIP

About

No description, website, or topics provided.

Readme

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

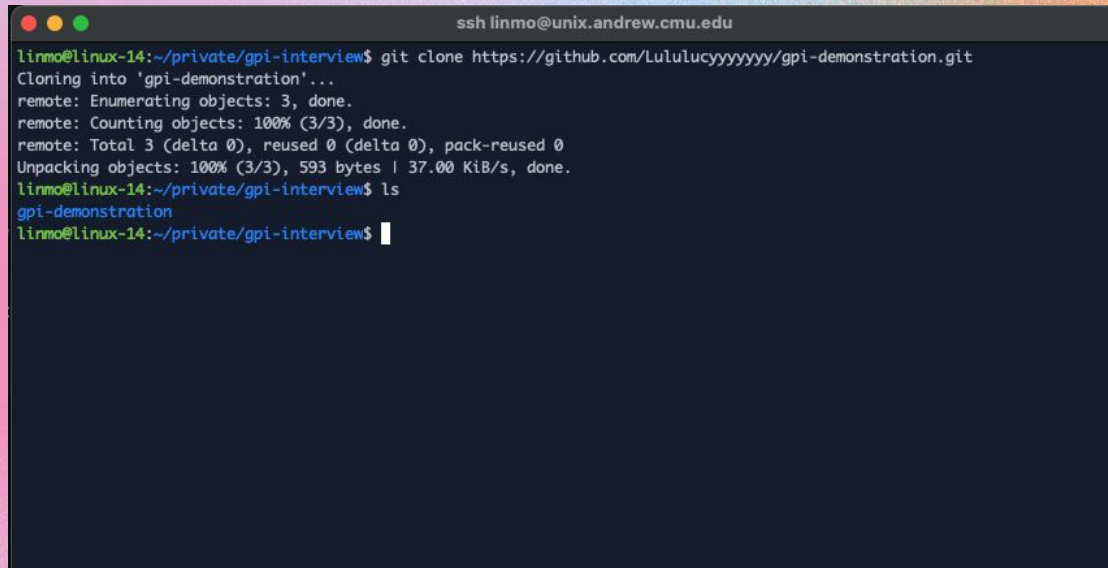
© 2022 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

Under “code”, copy the link to your repo

To clone:

```
$ git clone github-link
```

This creates a new folder with
your github project

A terminal window with a dark background and light text. The title bar at the top reads "ssh linmo@unix.andrew.cmu.edu". The terminal content shows a user running a git clone command to clone a repository from GitHub. The output shows the cloning process, including enumerating and counting objects, and unpacking. Finally, the user runs 'ls' and the output shows a new folder named 'gpi-demonstration' has been created.

```
linmo@linux-14:~/private/gpi-interview$ git clone https://github.com/Lululucyyyyyy/gpi-demonstration.git
Cloning into 'gpi-demonstration'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 593 bytes | 37.00 KiB/s, done.
linmo@linux-14:~/private/gpi-interview$ ls
gpi-demonstration
linmo@linux-14:~/private/gpi-interview$
```


UPDATING your repo

Pushing:

```
$ git push <remote name> <remote branch>
```

When your local branch isn't on the remote:

```
$ git push --set-upstream <remote name> <remote  
branch>
```

Pushing to current remote:

```
$ git push
```

Pulling:

```
$ git pull <remote name> <local branch>
```

Pulling current remote:

```
$ git pull
```


demo !

FETCH + merge = PULL

- Fetch allows you to review the changes before you merge your code with the remote code
- It does not merge your code yet!
- `$ git fetch <remote>`



FORKS

These are **duplicate remotes** of another remote giving you your own **private copy**

Why use forks?

- You don't have write permissions for the original remote
 - You want to have one just for you to use while the original is for your group
- 

PULL requests (PR)

- If you want your changes to be merged into the original remote version, make a PR!
- Just
 1. Push your code
 2. Go to the remote and make a PR
- The owner of the repo will then review your code and hopefully merge your PR

Announcements

- No lecture in the next 2 weeks – happy break!
- **Make sure to submit your username using this form:**
https://docs.google.com/forms/d/e/1FAIpQLSeLIL8Q8xKph-dSL-TUAuQ3OrKUE9knvASjLwxierOpJjbSyg/viewform?usp=sf_link
- Feedback form:
<https://forms.gle/wLZsrgcee1kFQ2Vs9>
- Regardless of how many late days you have left, all labs must be turned in by the hard deadline of **October 23rd**. After this date, unsubmitted labs will be marked as a 0.

Lab Clarifications

- When running “git status” or “git diff”, do it inside the “binsearch” directory
- If you have trouble accessing github, let a TA know!
- GitHub recently implemented 2FA.. :)
Search “github personal access token” and follow the StackOverflow link to set this up