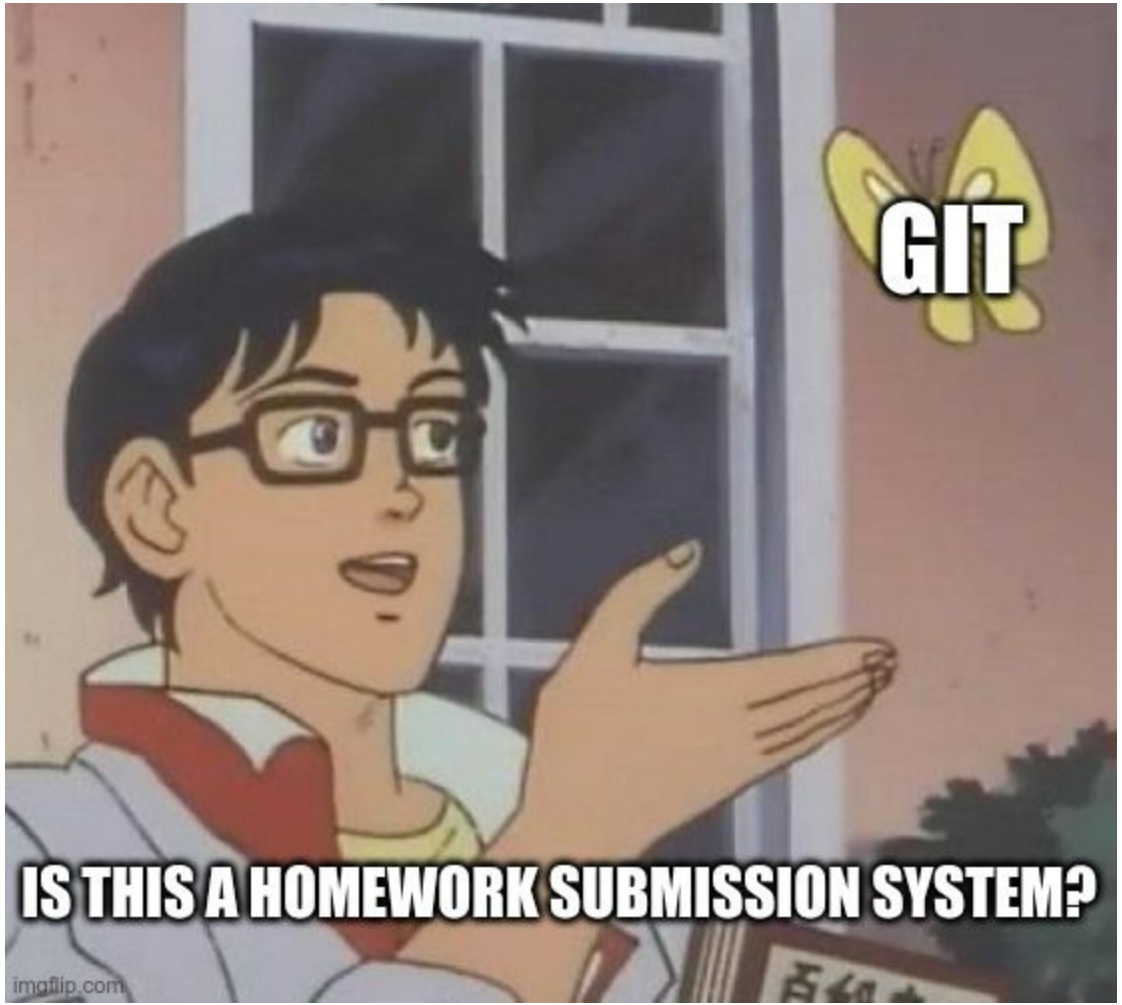
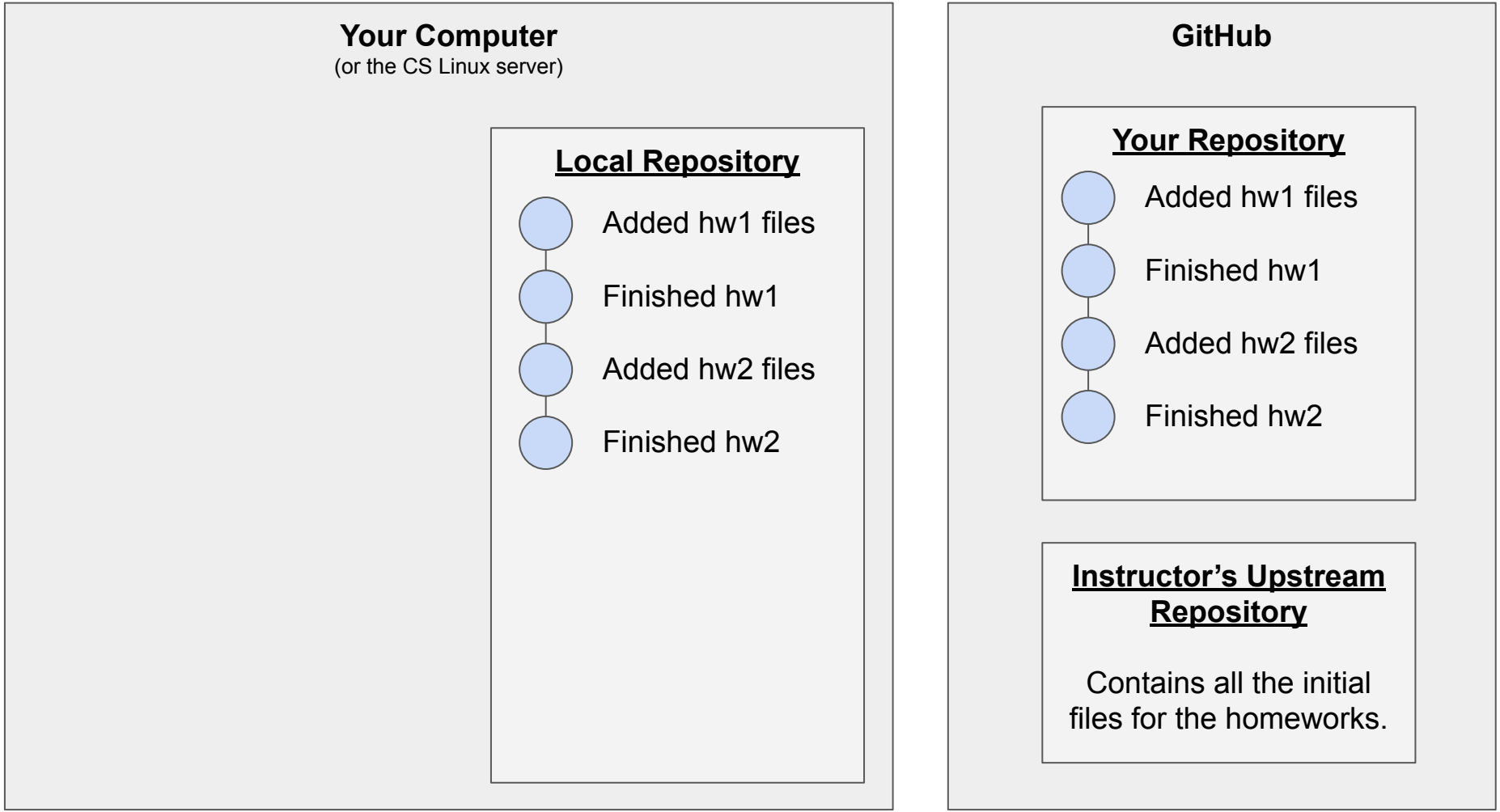
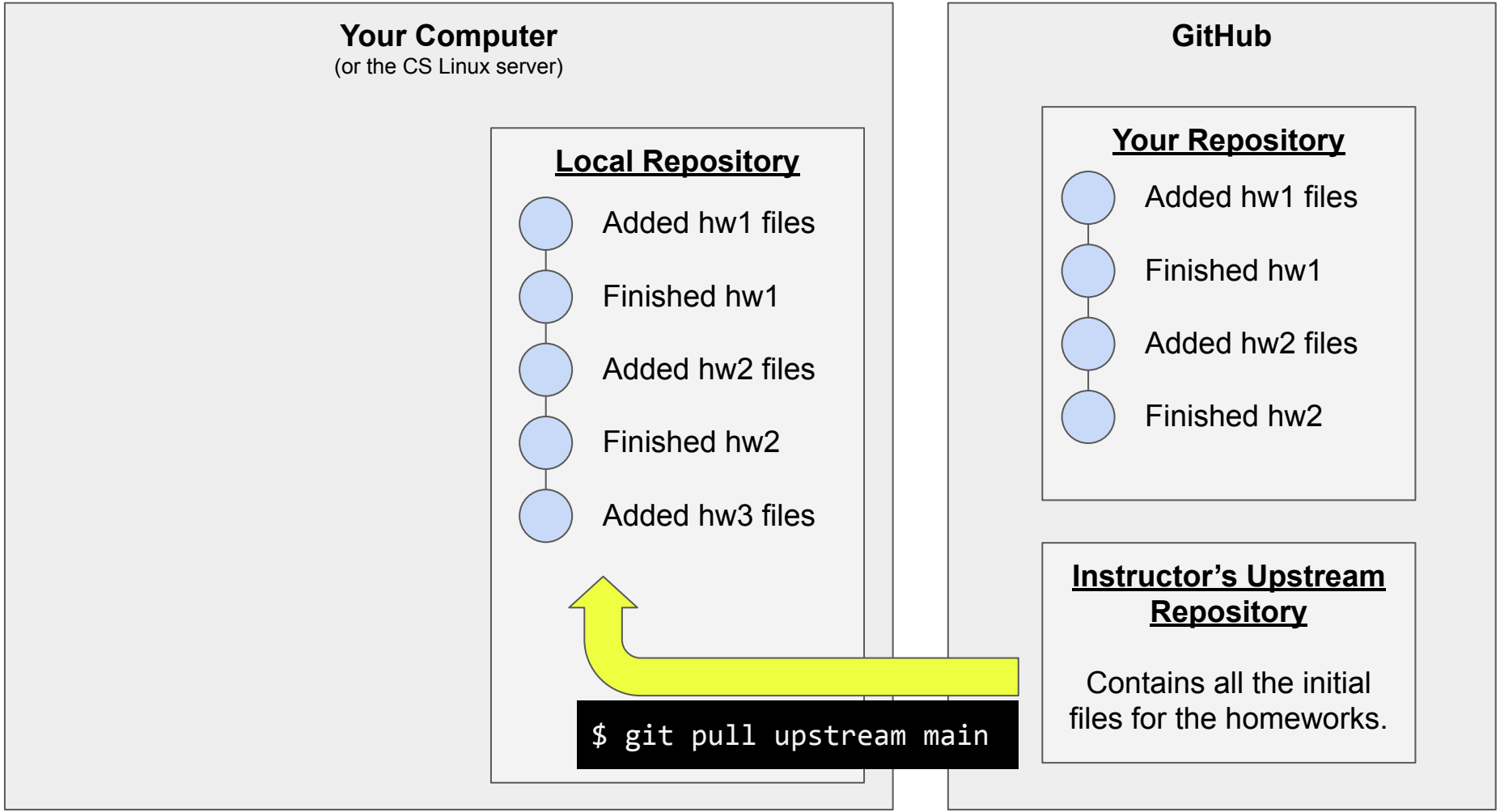
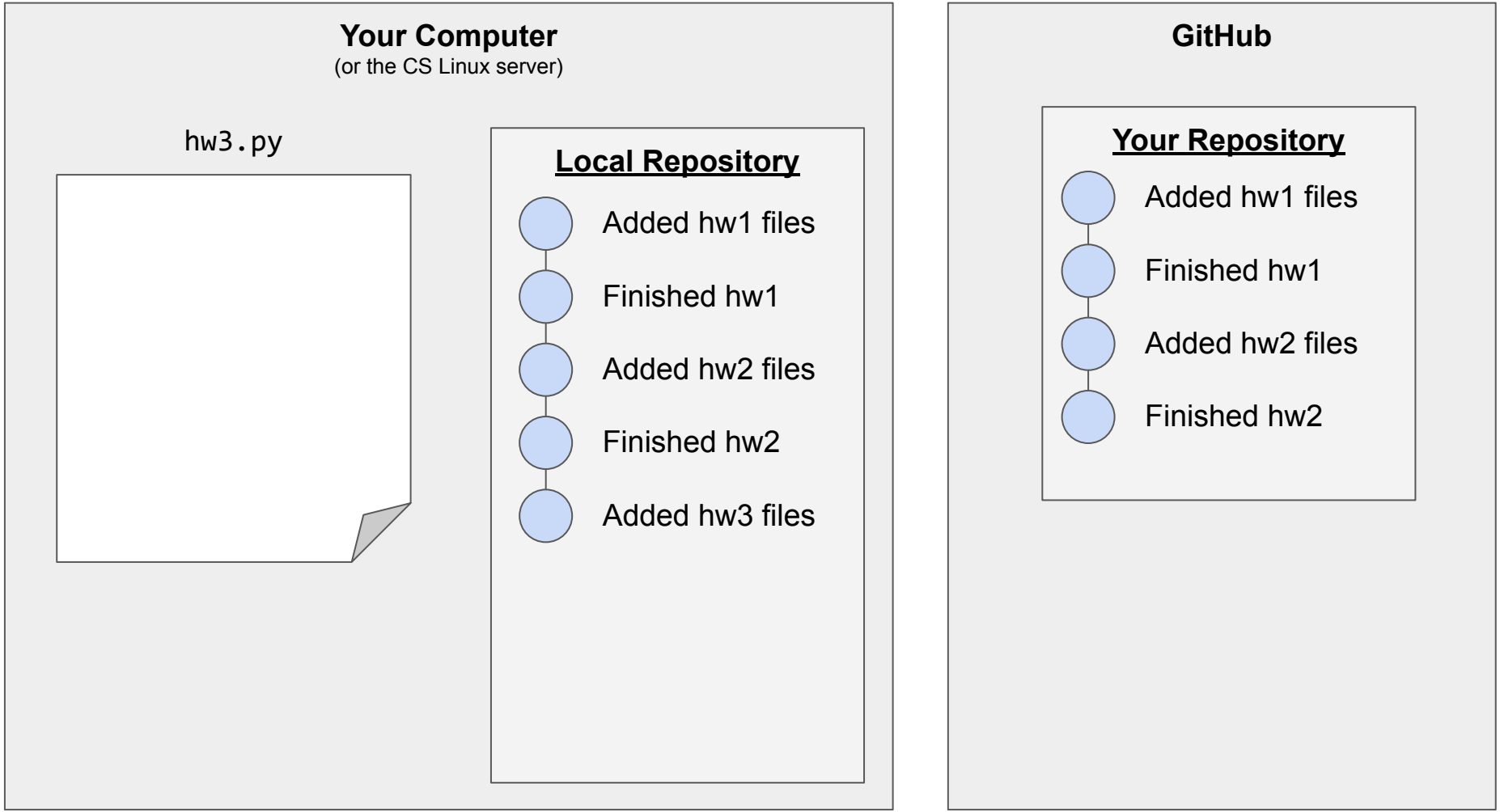
How You Should Be Using Git

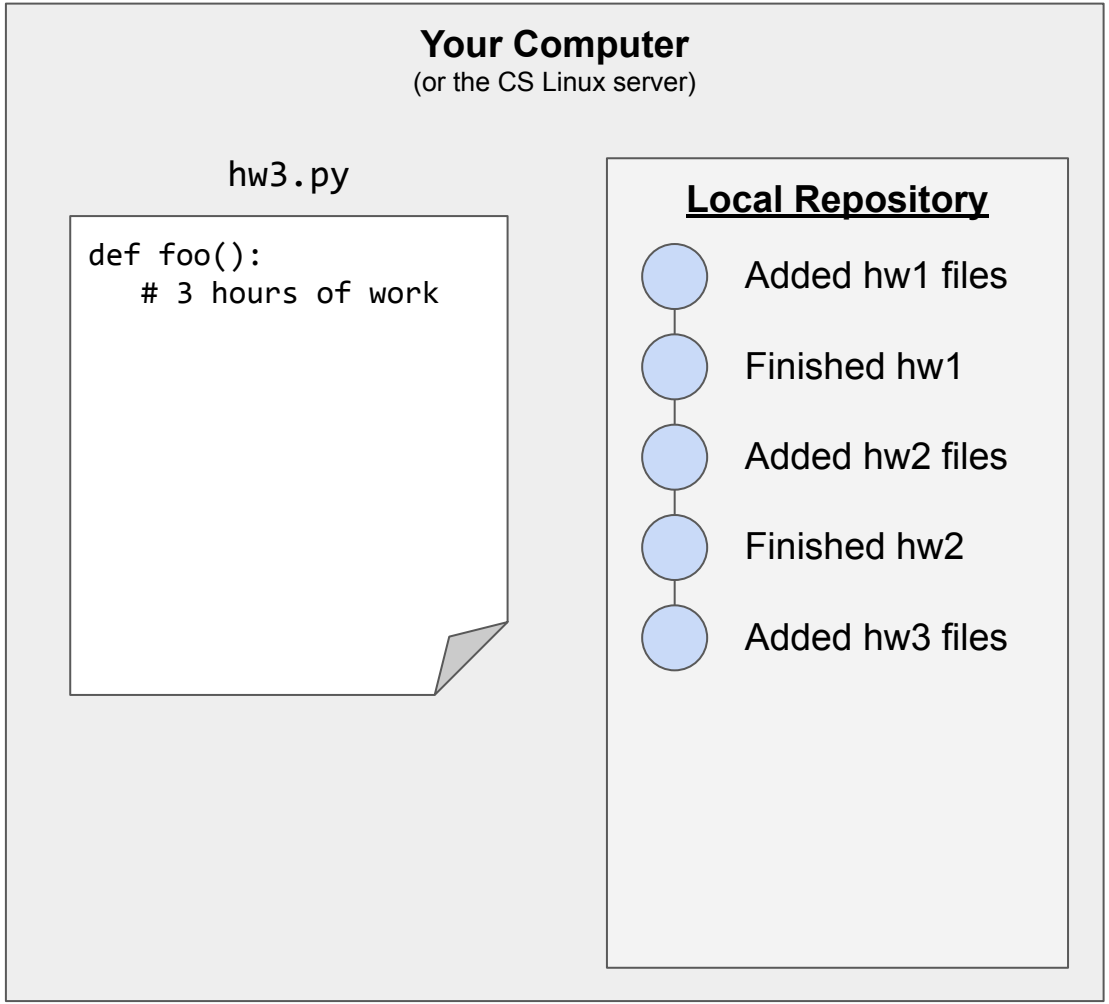
Let’s actually start by looking at how *not* to use Git

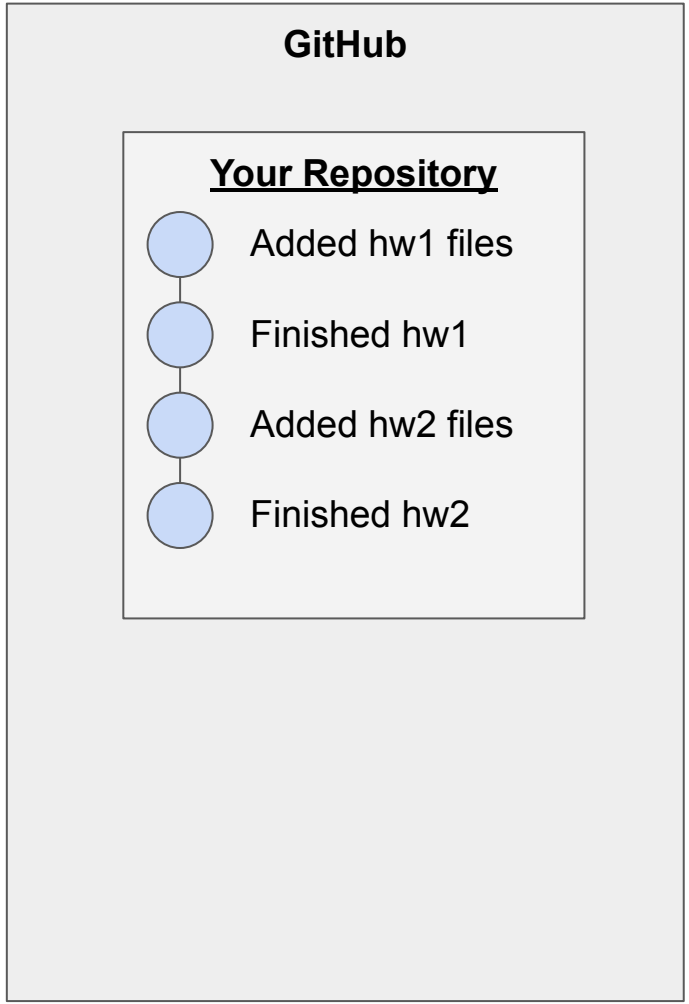

GIT
IS THIS A HOMEWORK SUBMISSION SYSTEM?
imgflip.com

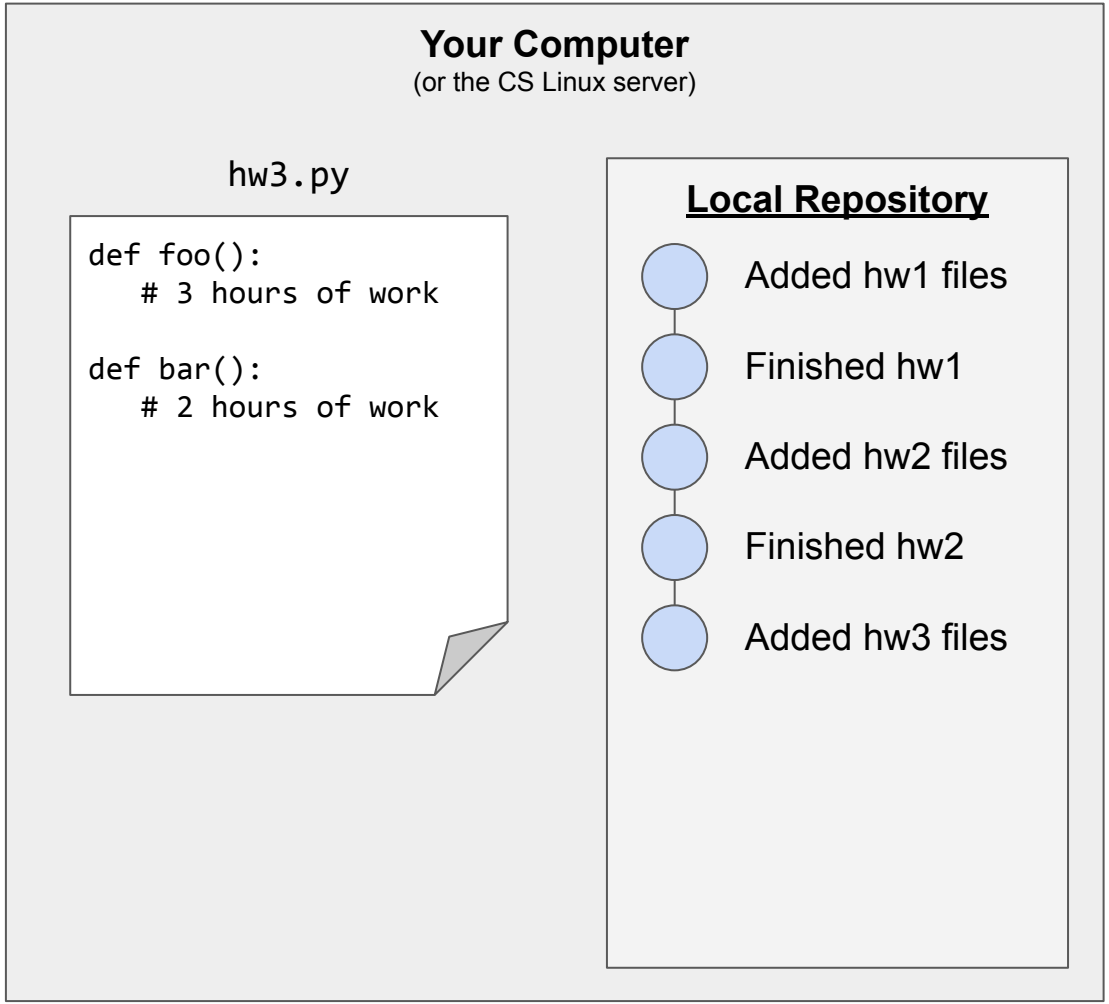

Your Computer
(or the CS Linux server)
Local Repository
Added hw1 files
Finished hw1
Added hw2 files
Finished hw2
GitHub
Your Repository
Added hw1 files
Finished hw1
Added hw2 files
Finished hw2
Instructor's Upstream Repository
Contains all the initial files for the homeworks.

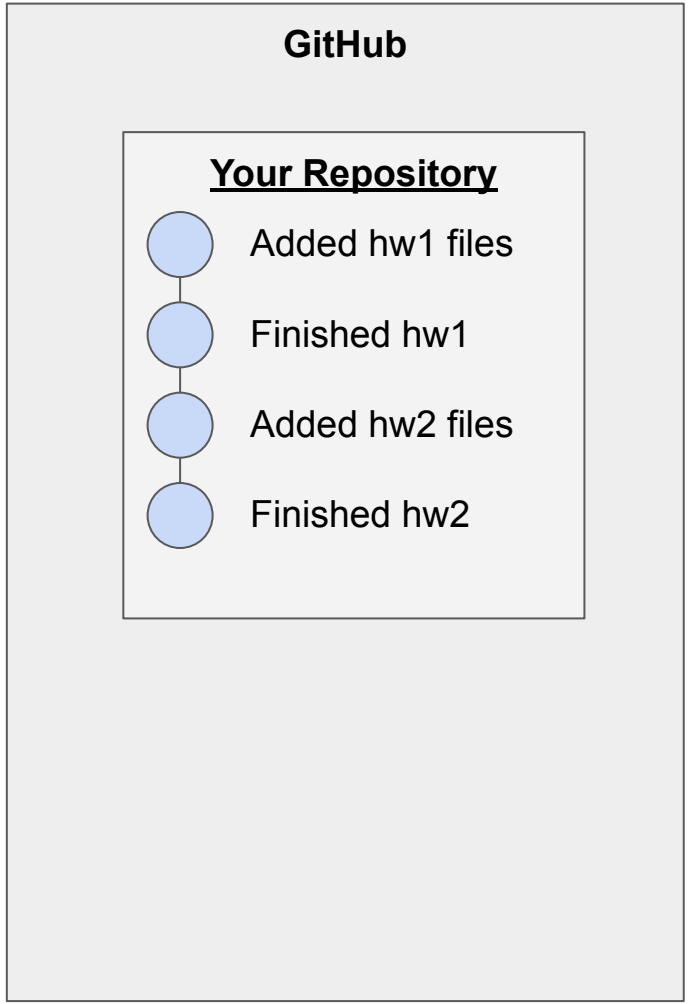

Your Computer
(or the CS Linux server)
Local Repository
Added hw1 files
Finished hw1
Added hw2 files
Finished hw2
Added hw3 files
$ git pull upstream main
GitHub
Your Repository
Added hw1 files
Finished hw1
Added hw2 files
Finished hw2
Instructor's Upstream Repository
Contains all the initial files for the homeworks.


Your Computer
(or the CS Linux server)
hw3.py
Local Repository
Added hw1 files
Finished hw1
Added hw2 files
Finished hw2
Added hw3 files
GitHub
Your Repository
Added hw1 files
Finished hw1
Added hw2 files
Finished hw2


Your Computer
(or the CS Linux server)
hw3.py
def foo():
# 3 hours of work
Local Repository
Added hw1 files
Finished hw1
Added hw2 files
Finished hw2
Added hw3 files


GitHub
Your Repository
Added hw1 files
Finished hw1
Added hw2 files
Finished hw2


Your Computer
(or the CS Linux server)
hw3.py
def foo():
# 3 hours of work
def bar():
# 2 hours of work
Local Repository
Added hw1 files
Finished hw1
Added hw2 files
Finished hw2
Added hw3 files


GitHub
Your Repository
Added hw1 files
Finished hw1
Added hw2 files
Finished hw2

Your Computer
(or the CS Linux server)

hw3.py

```
def foo():
   # 3 hours of work

def bar():
   # 2 hours of work

def baz():
   # 4 hours of work
```

Local Repository

- Added hw1 files
- Finished hw1
- Added hw2 files
- Finished hw2
- Added hw3 files

GitHub

Your Repository

- Added hw1 files
- Finished hw1
- Added hw2 files
- Finished hw2

Your Computer
(or the CS Linux server)

hw3.py

```
def foo():
   # 3 hours of work

def bar():
   # 2 hours of work

def baz():
   # 4 hours of work

def xyz():
   # 1 hour of work
```

Local Repository

- Added hw1 files
- Finished hw1
- Added hw2 files
- Finished hw2
- Added hw3 files

GitHub

Your Repository

- Added hw1 files
- Finished hw1
- Added hw2 files
- Finished hw2

Your Computer
(or the CS Linux server)

hw3.py

```
def foo():
   # 3 hours of work

def bar():
   # 2 hours of work

def baz():
   # 4 hours of work

def xyz():
   # 1 hour of work
```

Local Repository

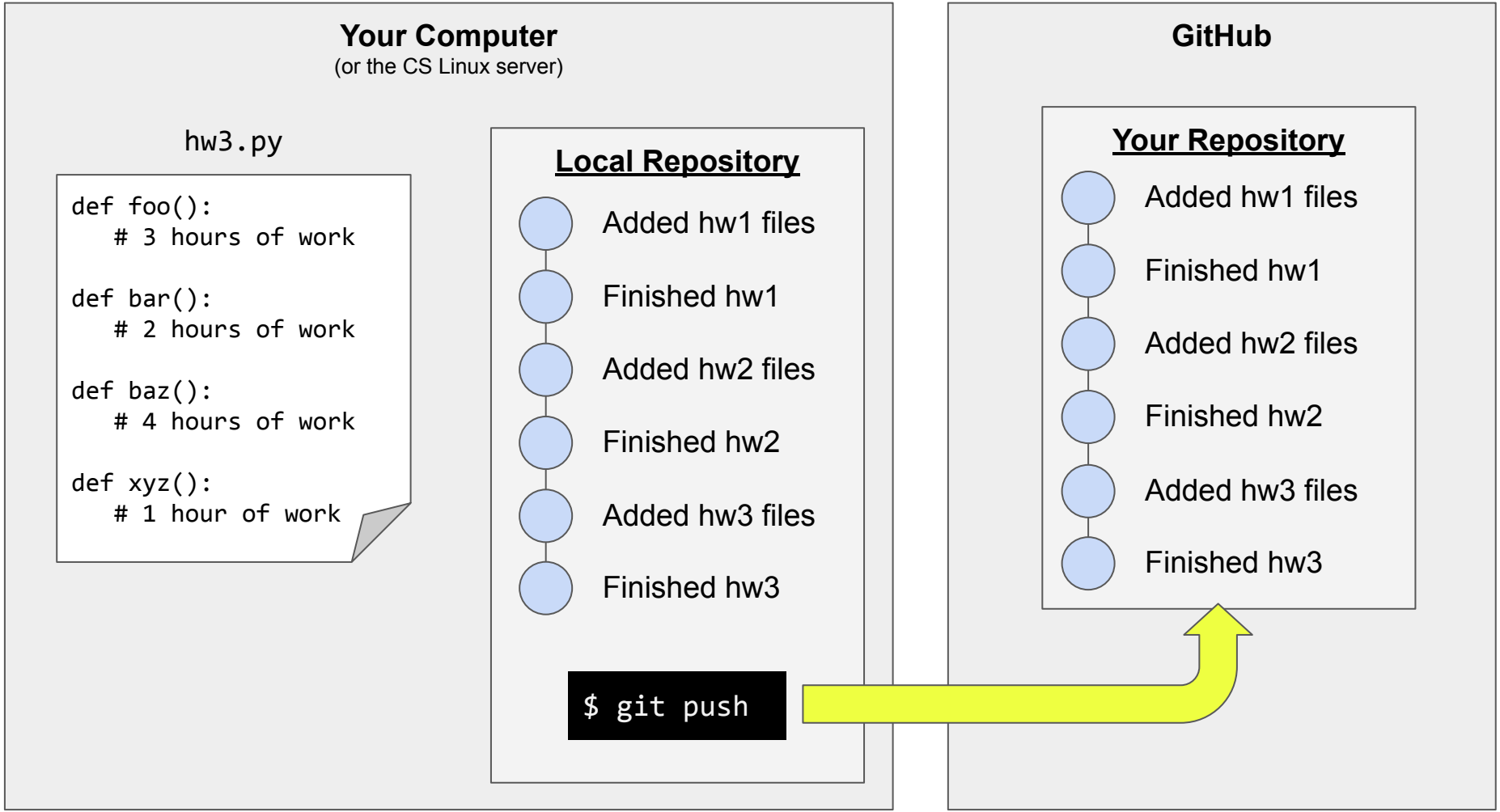
- Added hw1 files
- Finished hw1
- Added hw2 files
- Finished hw2
- Added hw3 files
- Finished hw3

```
$ git add hw3.py
$ git commit -m "Finished hw3"
```

GitHub

Your Repository

- Added hw1 files
- Finished hw1
- Added hw2 files
- Finished hw2


Your Computer
(or the CS Linux server)
hw3.py
def foo():
# 3 hours of work
def bar():
# 2 hours of work
def baz():
# 4 hours of work
def xyz():
# 1 hour of work
Local Repository
Added hw1 files
Finished hw1
Added hw2 files
Finished hw2
Added hw3 files
Finished hw3
$ git push
GitHub
Your Repository
Added hw1 files
Finished hw1
Added hw2 files
Finished hw2
Added hw3 files
Finished hw3

If your laptop breaks down / is lost / is stolen / etc. before you make the “Finished hw3” commit…

→ You lose all your work

If you delete/overwrite `hw3.py` by mistake before you make the “Finished hw3” commit…

→ You lose all your work

If you want to get some work done in the computer lab / back home / etc., but you left your laptop in your dorm…

→ No way to access your files

If you go down an unproductive path, and decide you want to go back to an earlier version of your code…

→ The editor you are using *may* let you do this but, in general, you’re probably out of luck (specially if the older version was several saves ago)

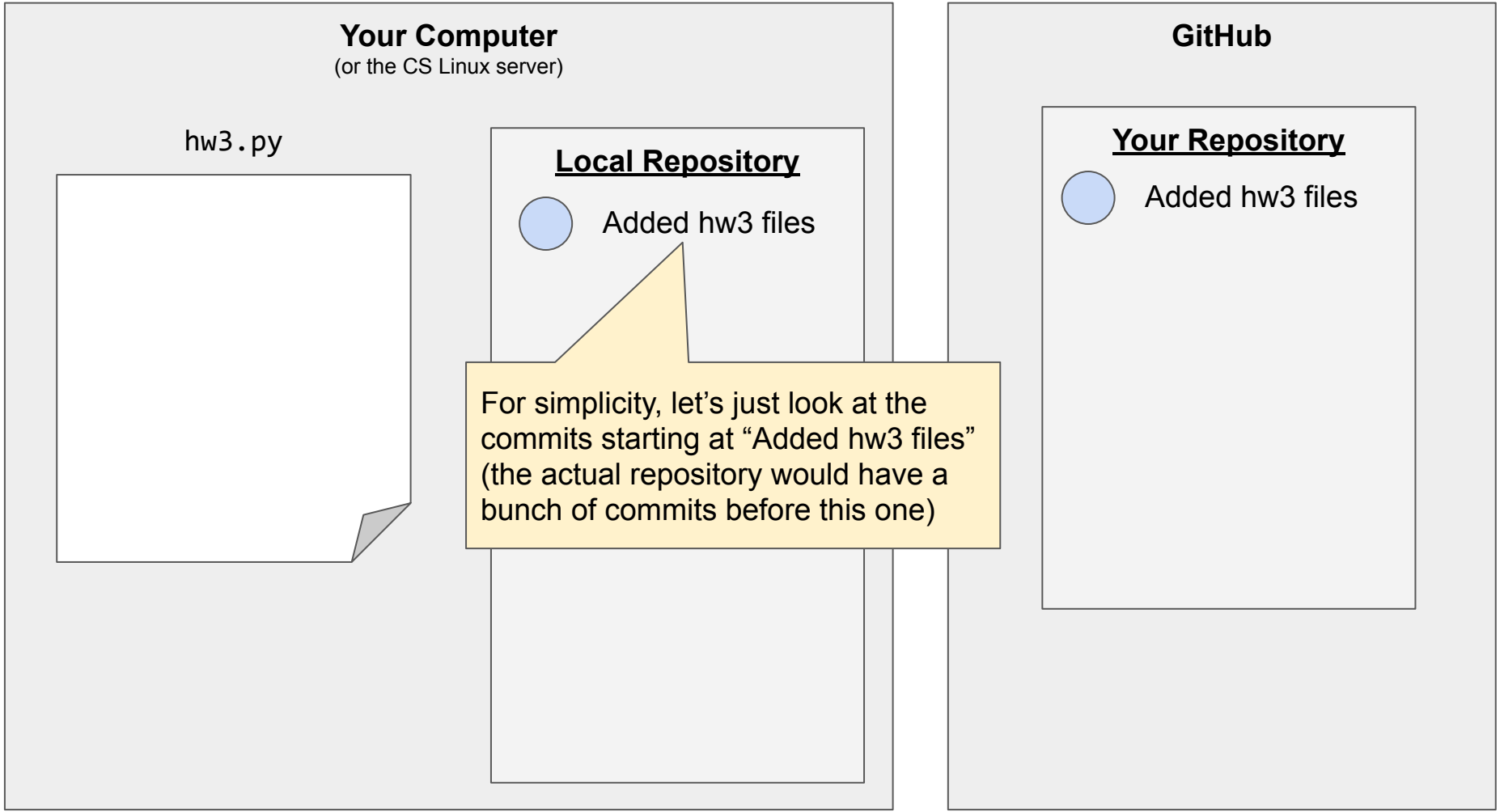
Git is much more than just a mechanism to submit your homework.

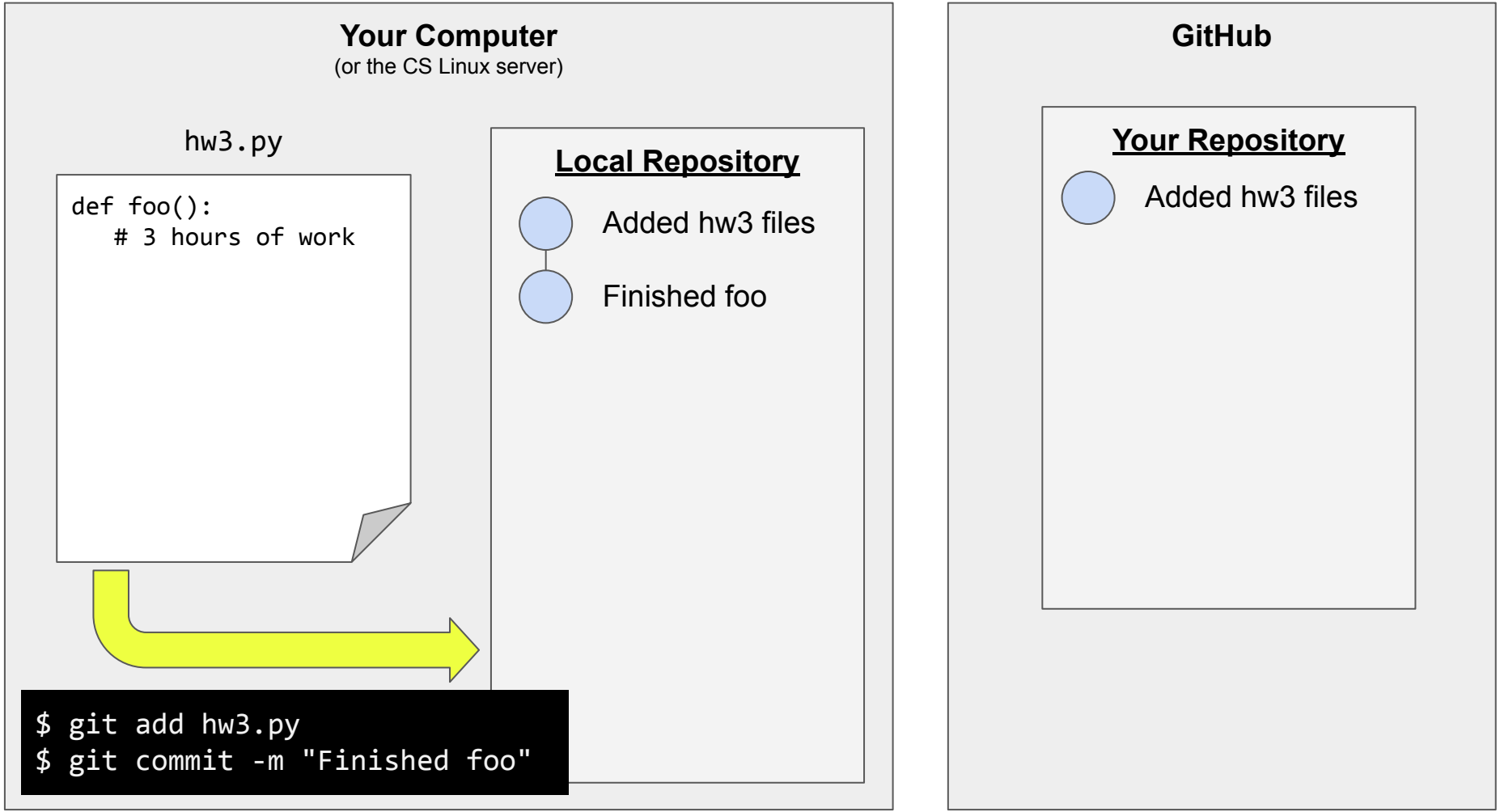
Git is a…

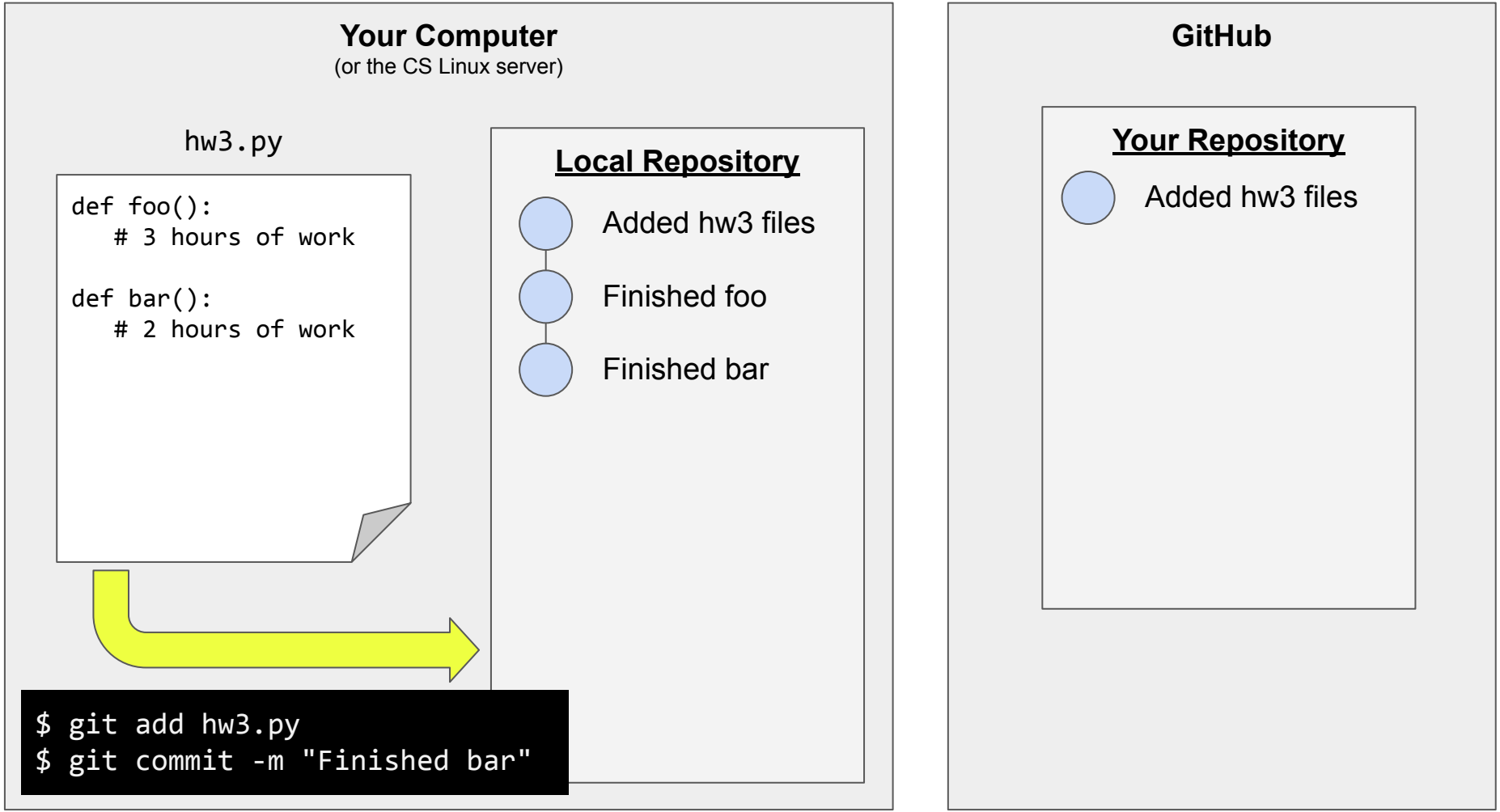
Distributed Version Control System

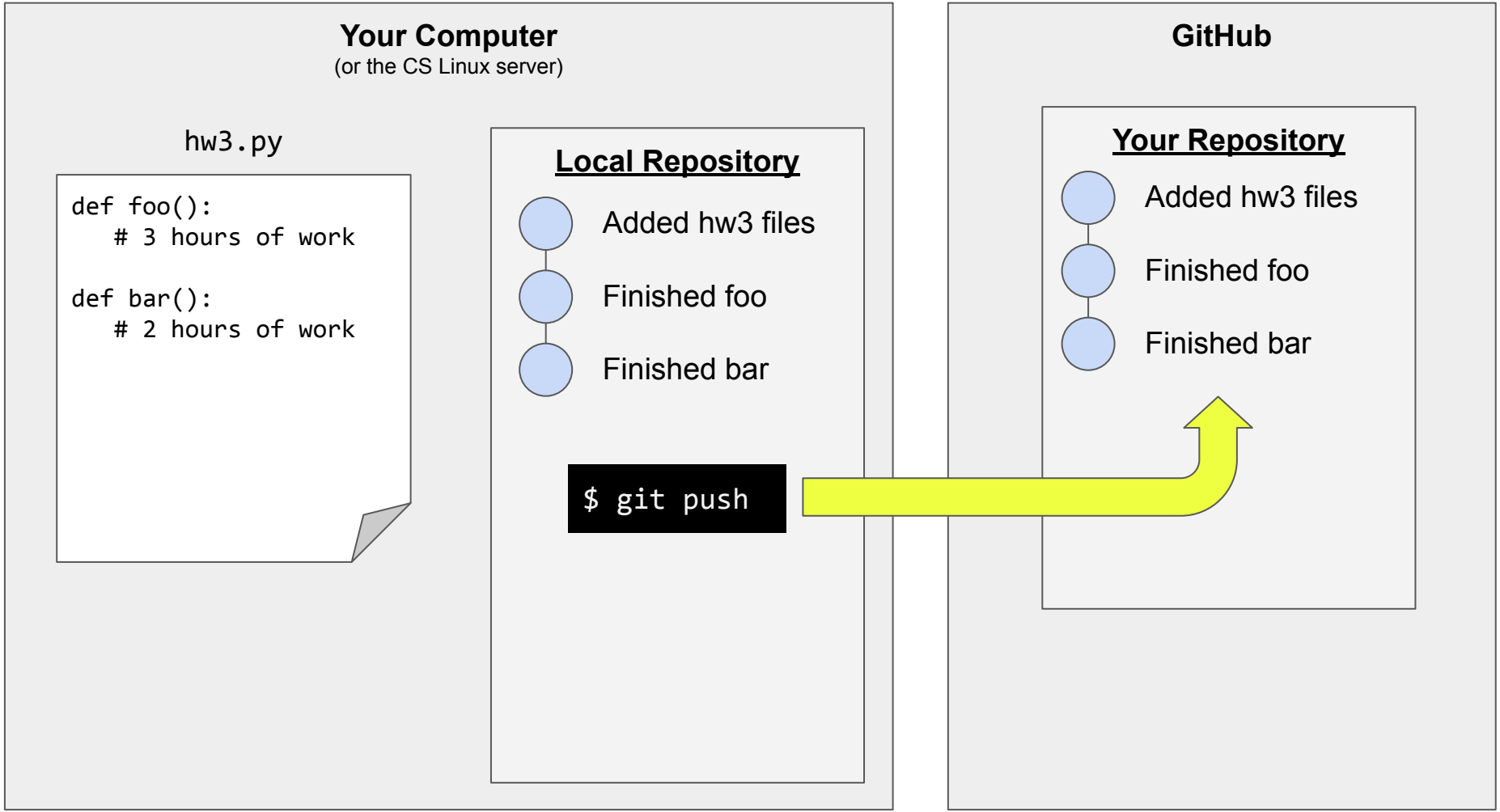
It is specifically designed to manage multiple copies (or “clones”) of the same repository in different locations (your computer, GitHub, the CS Linux server, etc.) and to reconcile conflicts between those copies. This will be very useful when working in a team.

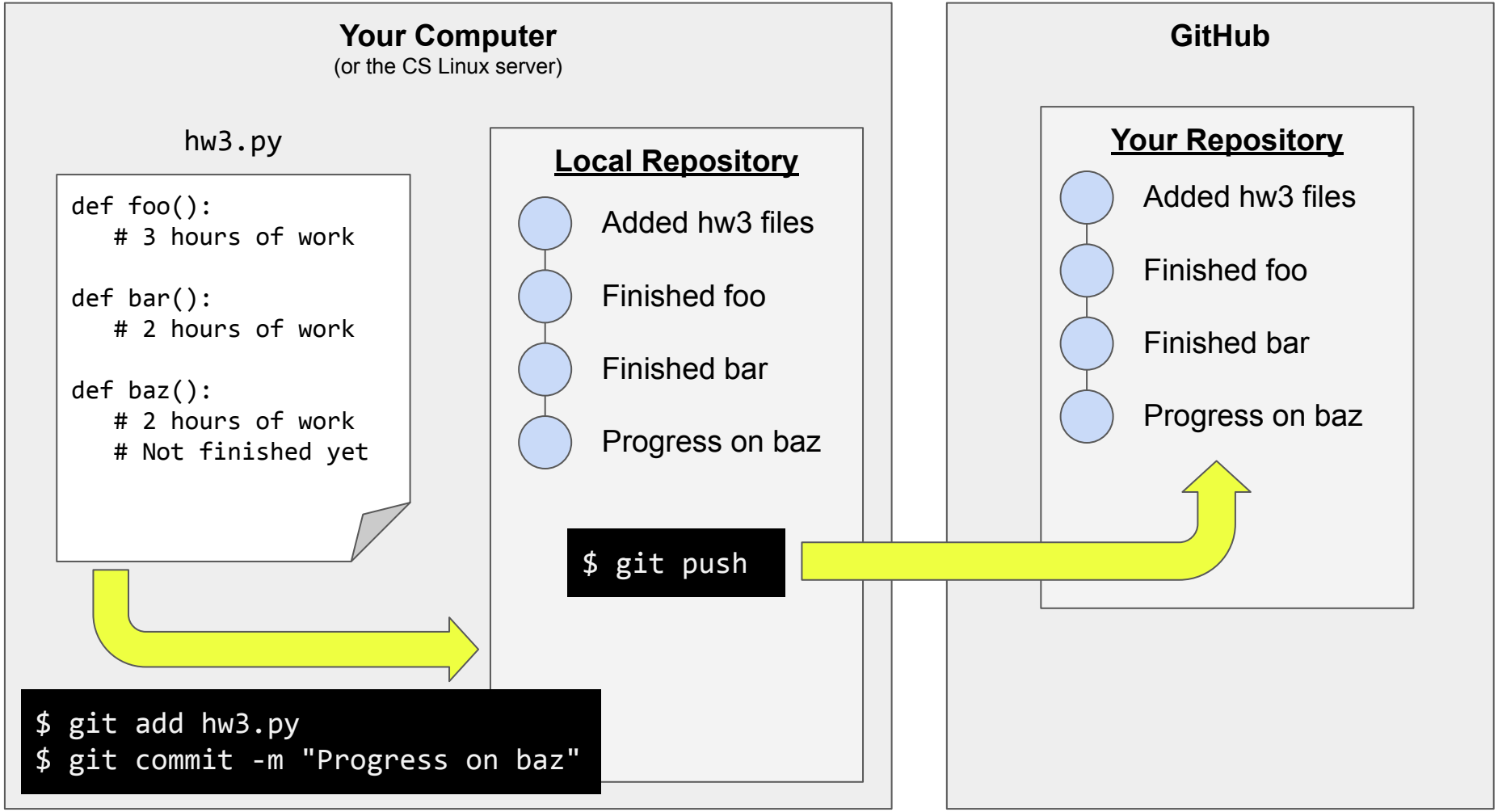
It is able to keep track of the various changes and revisions (“versions”) that a piece of code goes through


Your Computer
(or the CS Linux server)
hw3.py
Local Repository
Added hw3 files
For simplicity, let's just look at the commits starting at "Added hw3 files" (the actual repository would have a bunch of commits before this one)
GitHub
Your Repository
Added hw3 files


Your Computer
(or the CS Linux server)
hw3.py
def foo():
# 3 hours of work
Local Repository
Added hw3 files
Finished foo
$ git add hw3.py
$ git commit -m "Finished foo"
GitHub
Your Repository
Added hw3 files


Your Computer
(or the CS Linux server)
hw3.py
def foo():
# 3 hours of work
def bar():
# 2 hours of work
Local Repository
Added hw3 files
Finished foo
Finished bar
$ git add hw3.py
$ git commit -m "Finished bar"
GitHub
Your Repository
Added hw3 files


Your Computer
(or the CS Linux server)
hw3.py
def foo():
# 3 hours of work
def bar():
# 2 hours of work
Local Repository
Added hw3 files
Finished foo
Finished bar
$ git push
GitHub
Your Repository
Added hw3 files
Finished foo
Finished bar


Your Computer
(or the CS Linux server)
hw3.py
def foo():
# 3 hours of work
def bar():
# 2 hours of work
def baz():
# 2 hours of work
# Not finished yet
Local Repository
Added hw3 files
Finished foo
Finished bar
Progress on baz
$ git push
$ git add hw3.py
$ git commit -m "Progress on baz"
GitHub
Your Repository
Added hw3 files
Finished foo
Finished bar
Progress on baz

Your Computer
(or the CS Linux server)

hw3.py

```
def foo():
   # 3 hours of work

def bar():
   # 2 hours of work

def baz():
   # 4 hours of work
```

Local Repository

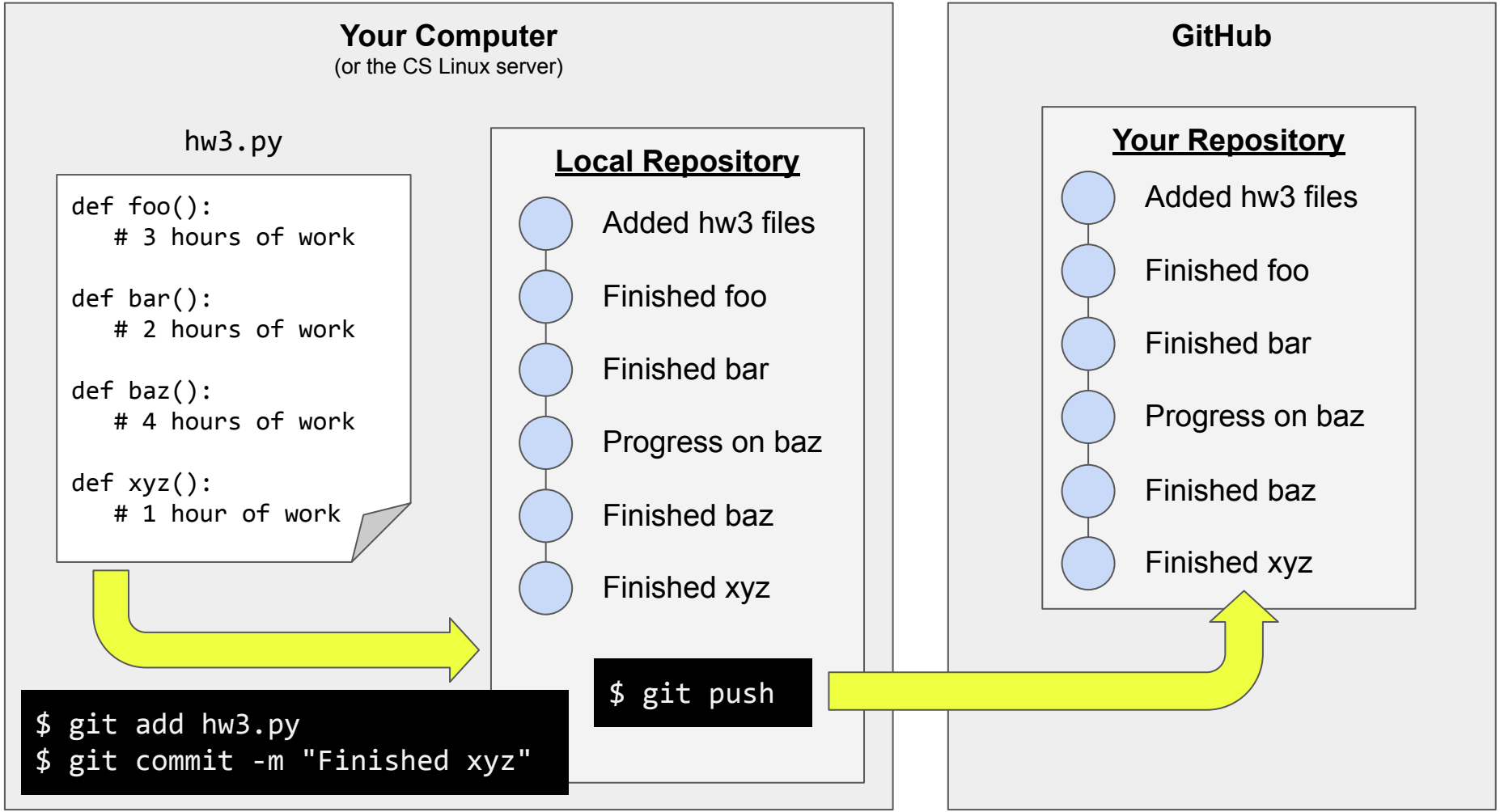
- Added hw3 files
- Finished foo
- Finished bar
- Progress on baz
- Finished baz

```
$ git add hw3.py
$ git commit -m "Finished baz"
```

GitHub

Your Repository

- Added hw3 files
- Finished foo
- Finished bar
- Progress on baz


Your Computer
(or the CS Linux server)
hw3.py
def foo():
# 3 hours of work
def bar():
# 2 hours of work
def baz():
# 4 hours of work
def xyz():
# 1 hour of work
Local Repository
Added hw3 files
Finished foo
Finished bar
Progress on baz
Finished baz
Finished xyz
$ git add hw3.py
$ git commit -m "Finished xyz"
$ git push
GitHub
Your Repository
Added hw3 files
Finished foo
Finished bar
Progress on baz
Finished baz
Finished xyz

Two good rules of thumb:

1. Make a commit any time you complete a piece of work that can be described with a phrase like “Finished <task>”, “Progress on <task>”, “Fixed bug in <task>”, etc. or with a specific outcome like “All tests passing now”, “<task> now type-checks”.

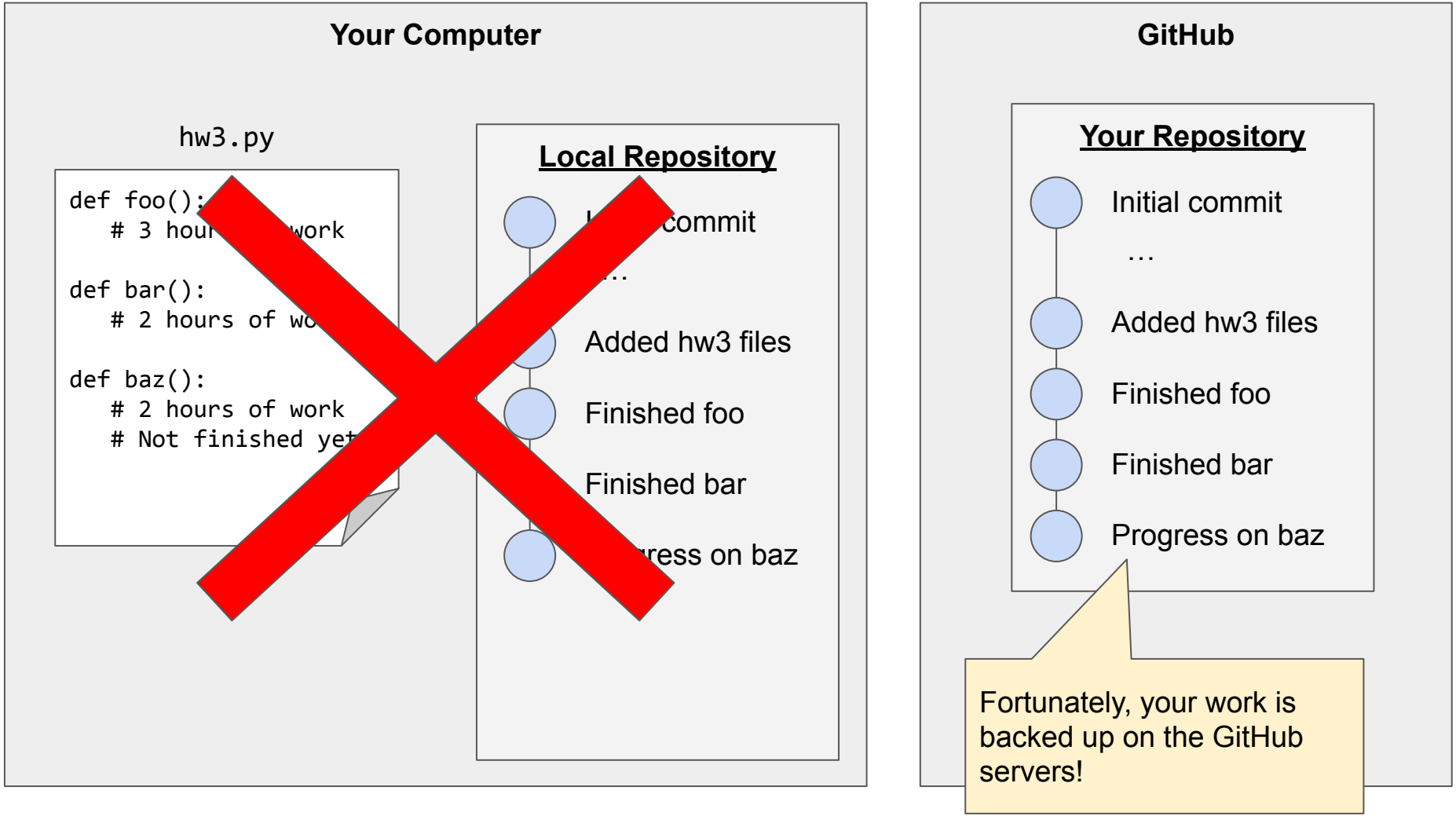
2. Push every time you step away from your computer.

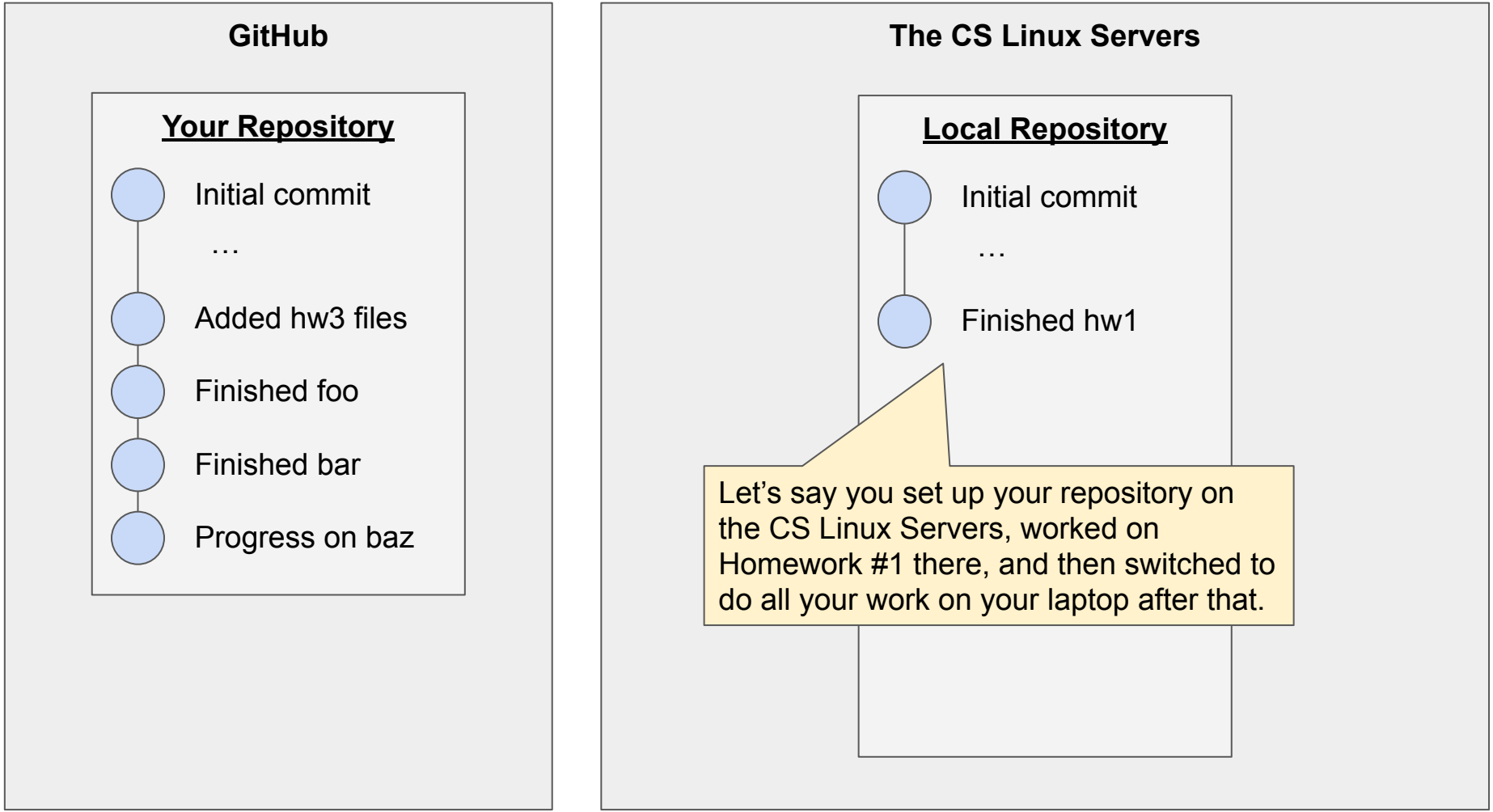
These are not strict rules, but they can help prevent all the scenarios we described earlier.

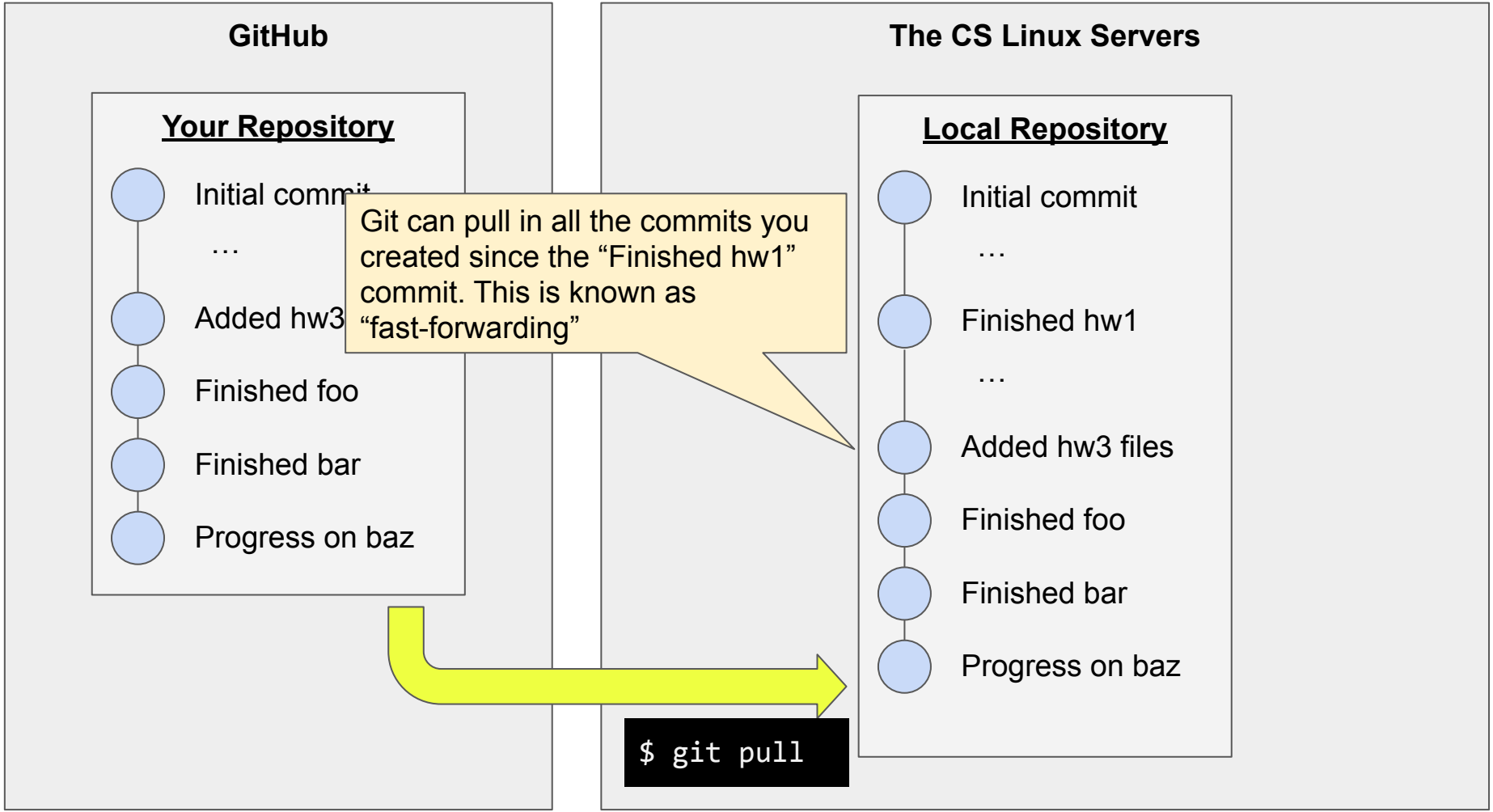
- Committing is like saving your progress in a game. If you die in the game, you’ll respawn at your last save point (i.e., your last commit)
- Pushing is like uploading your save data to a server, so you can resume playing the game in the same spot even if you switch to a different PC/console.

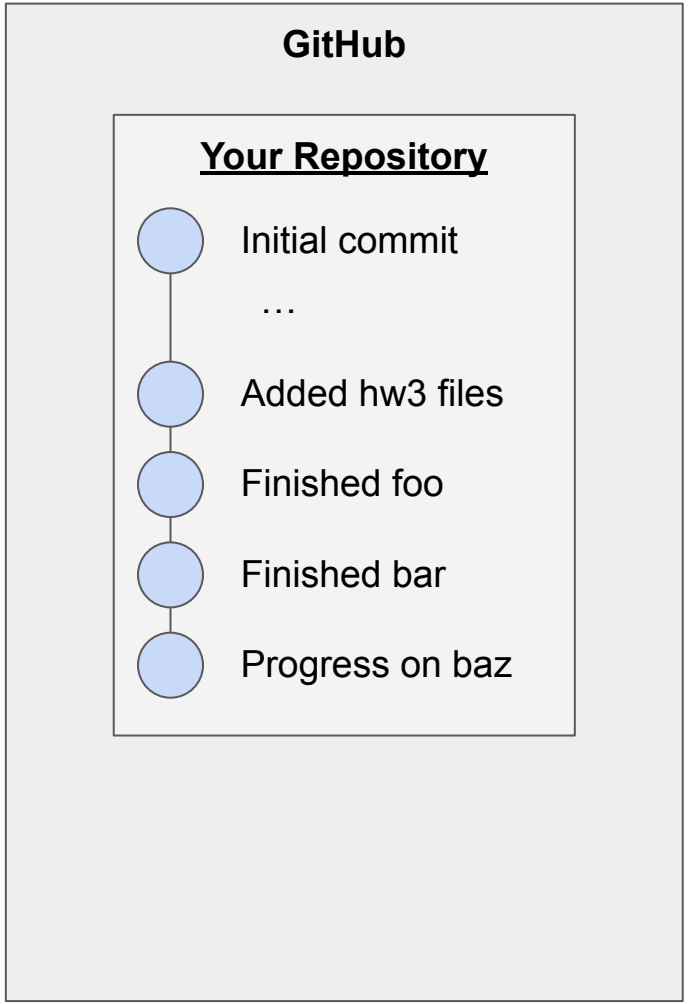
Let’s revisit the nightmare scenarios we saw earlier…

Your laptop becomes inaccessible in some way (you left it in your dorm, it is broken, lost, stolen, etc.)

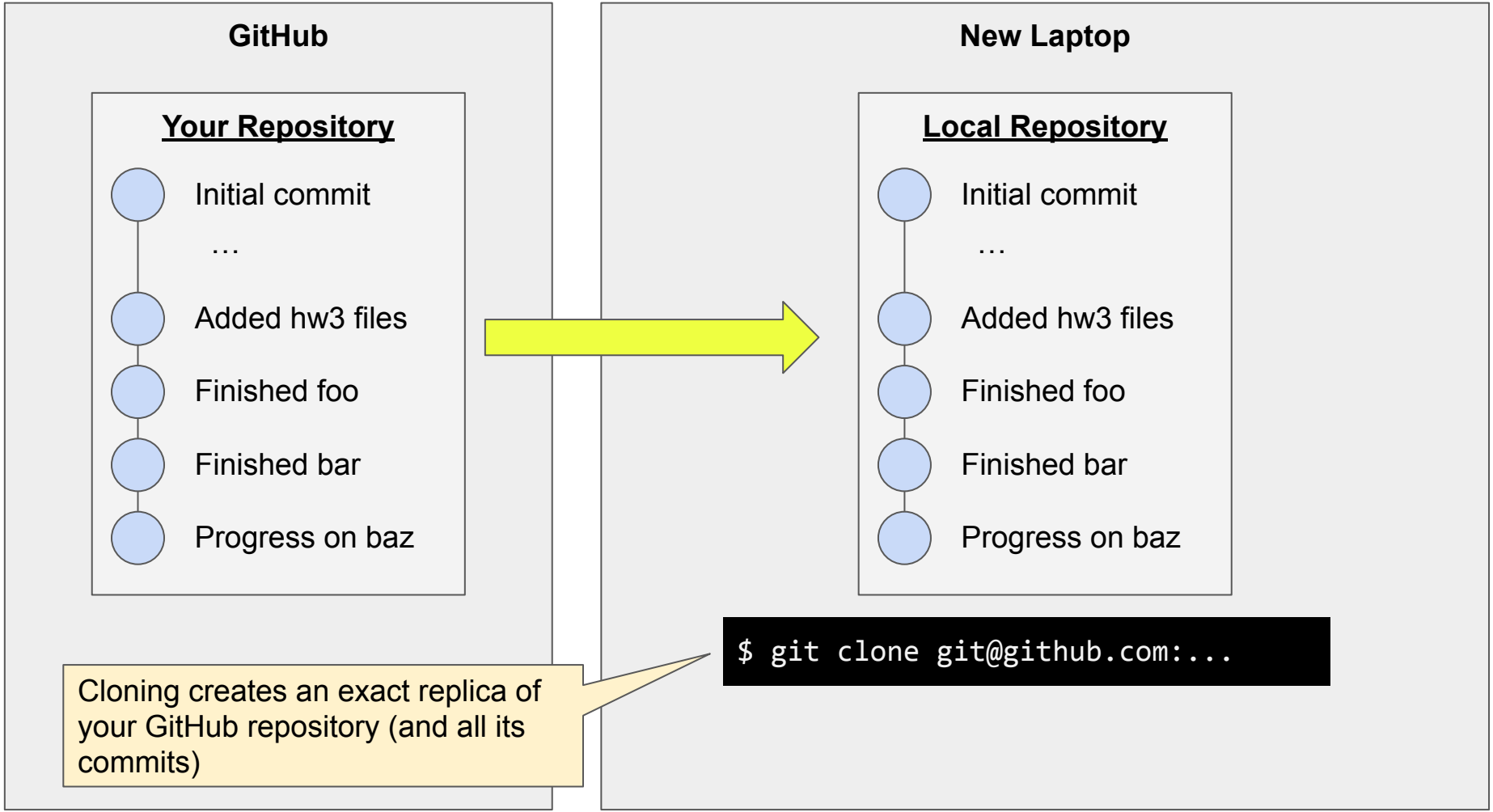

Your Computer
hw3.py
def foo():
# 3 hou work
def bar():
# 2 hours of wo
def baz():
# 2 hours of work
# Not finished yet
Local Repository
commit
Added hw3 files
Finished foo
Finished bar
ress on baz
GitHub
Your Repository
Initial commit
…
Added hw3 files
Finished foo
Finished bar
Progress on baz
Fortunately, your work is backed up on the GitHub servers!


GitHub
Your Repository
Initial commit
…
Added hw3 files
Finished foo
Finished bar
Progress on baz
The CS Linux Servers
Local Repository
Initial commit
…
Finished hw1
Let’s say you set up your repository on the CS Linux Servers, worked on Homework #1 there, and then switched to do all your work on your laptop after that.

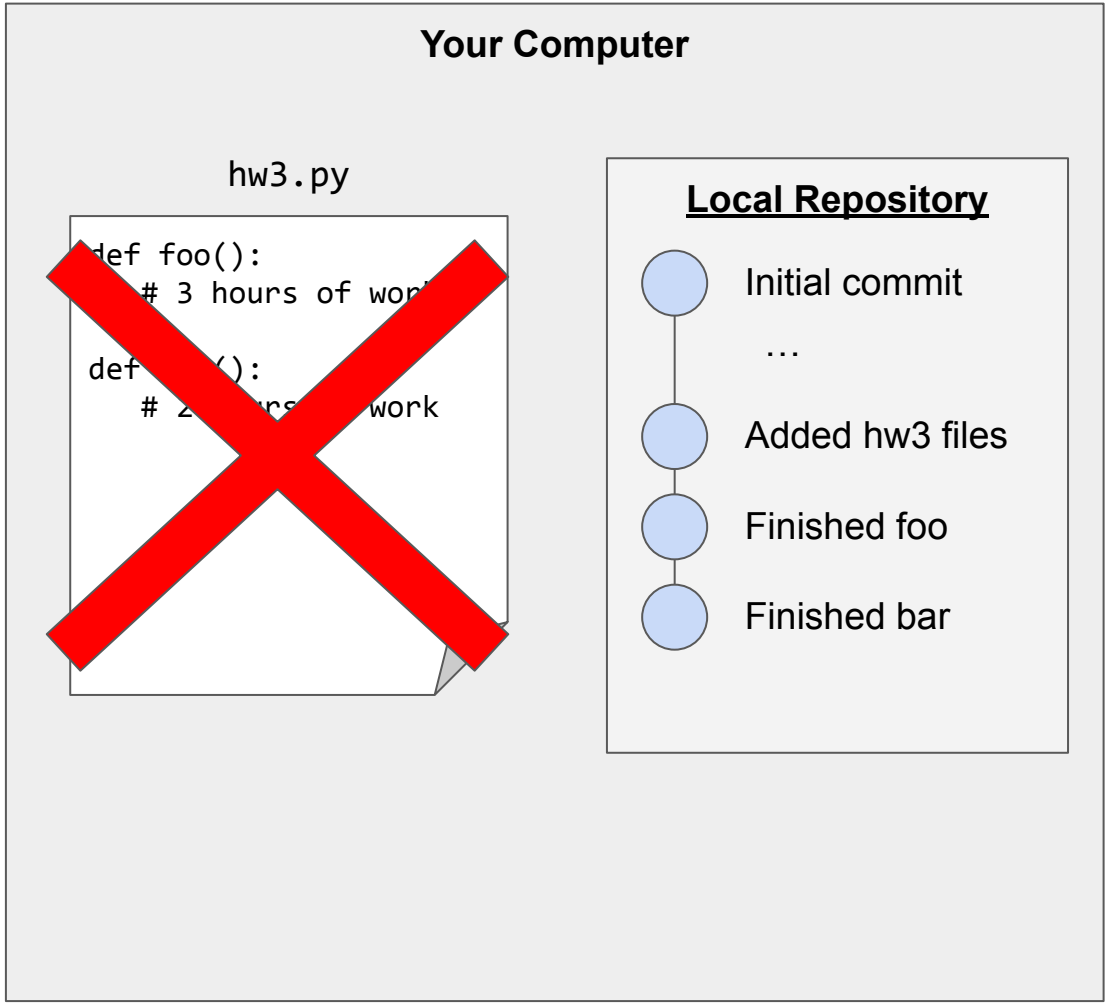

GitHub
Your Repository
Initial commit
…
Added hw3
Finished foo
Finished bar
Progress on baz
Git can pull in all the commits you created since the “Finished hw1” commit. This is known as “fast-forwarding”
The CS Linux Servers
Local Repository
Initial commit
…
Finished hw1
…
Added hw3 files
Finished foo
Finished bar
Progress on baz
$ git pull

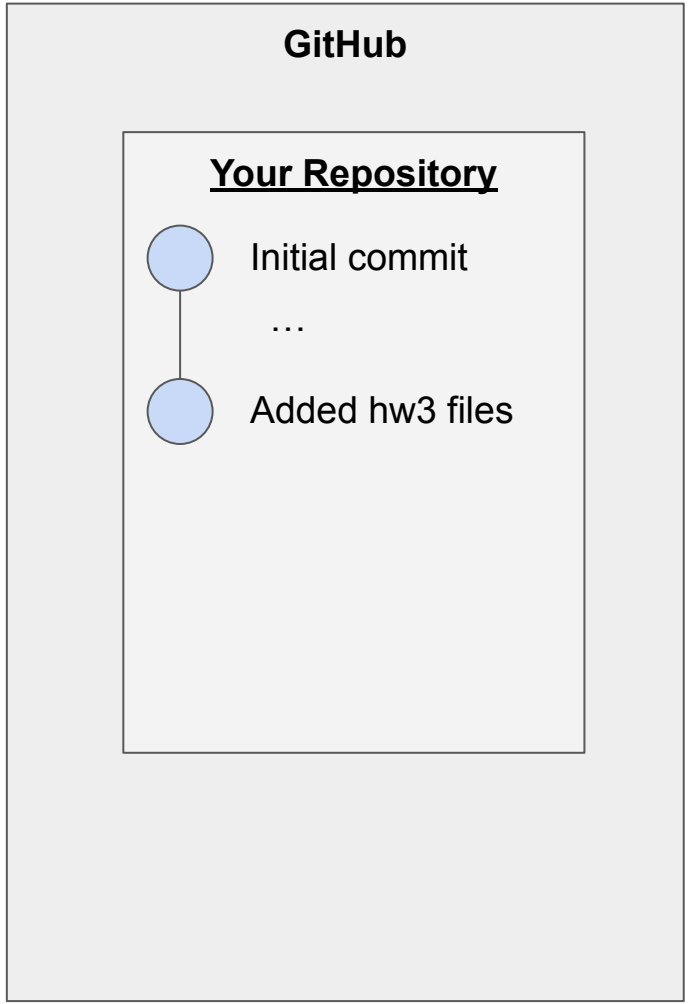

GitHub
Your Repository
Initial commit
…
Added hw3 files
Finished foo
Finished bar
Progress on baz

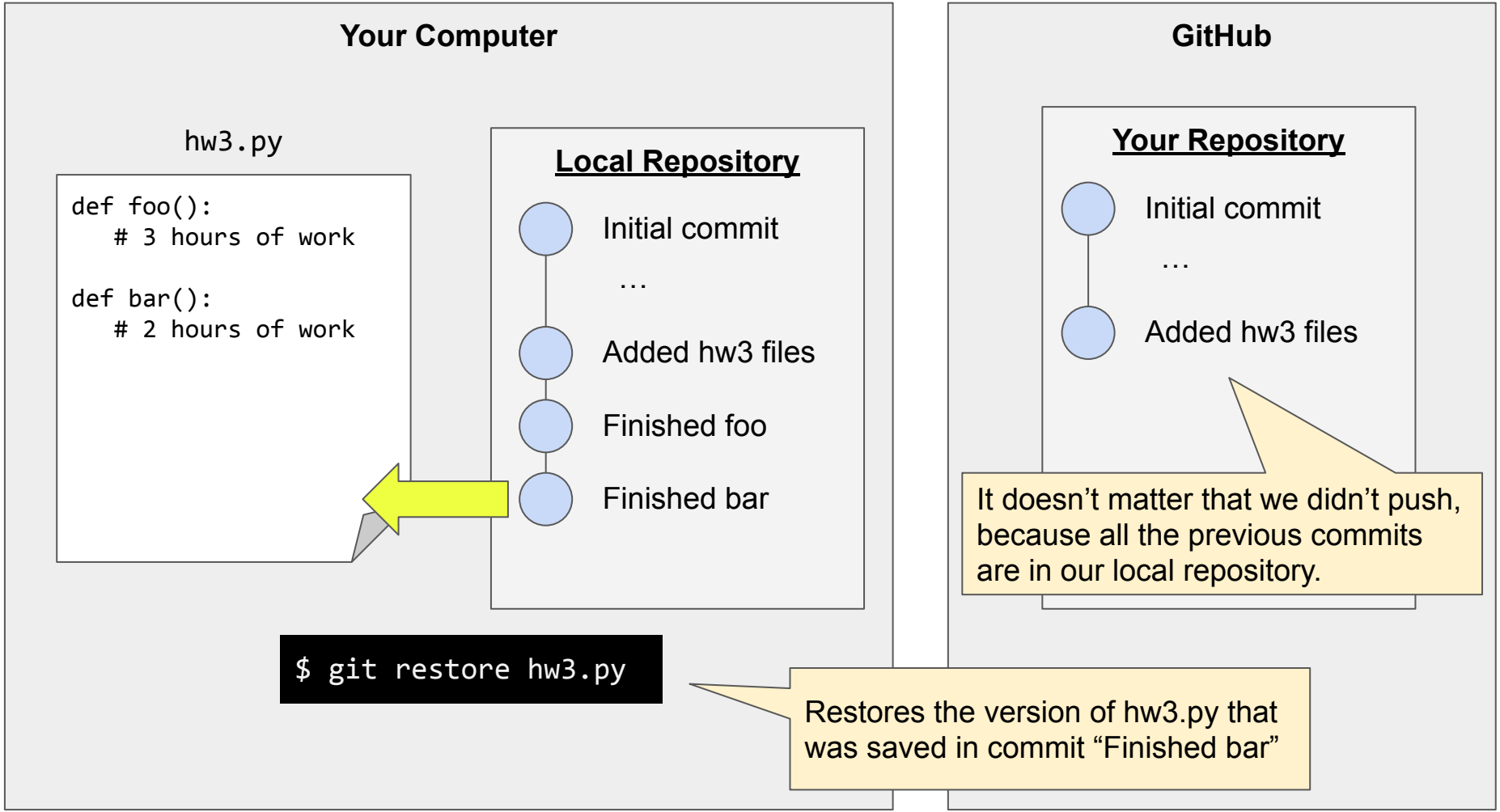

New Laptop
What if you get a new laptop?


GitHub
Your Repository
Initial commit
…
Added hw3 files
Finished foo
Finished bar
Progress on baz
New Laptop
Local Repository
Initial commit
…
Added hw3 files
Finished foo
Finished bar
Progress on baz
$ git clone git@github.com:...
Cloning creates an exact replica of your GitHub repository (and all its commits)

You delete `hw3.py` by mistake


Your Computer
hw3.py
def foo():
# 3 hours of wor
def
work
Local Repository
Initial commit
…
Added hw3 files
Finished foo
Finished bar


GitHub
Your Repository
Initial commit
…
Added hw3 files


Your Computer
hw3.py
def foo():
# 3 hours of work
def bar():
# 2 hours of work
Local Repository
Initial commit
…
Added hw3 files
Finished foo
Finished bar
$ git restore hw3.py
Restores the version of hw3.py that was saved in commit “Finished bar”
GitHub
Your Repository
Initial commit
…
Added hw3 files
It doesn’t matter that we didn’t push, because all the previous commits are in our local repository.

You want to go back to a previous version of your code

You can use `git restore` to revert all the changes to a file since your last commit (careful: this *cannot* be undone!)

While there are mechanisms to “rewind” your repository back to a previous commit, you will often just want to see what the code looked like at a given commit. This can be easily done on GitHub’s web interface.

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insi

main | 1 branch | 0 tags | Go to file | Add file | Code

borjasotomayor Added link to documentation in README ✓ e4feef8 last month 6 commits

.github/workflows	Minor update to GitHub Actions workflow	last month
docs	Initial commit	last month
example-clients/connectm	Initial commit	last month
src/chimera	Initial commit	last month
tests	Initial commit	last month
.gitignore	Initial commit	last month
.readthedocs.yml	Added .readthedocs.yml	last month
LICENSE	Initial commit	last month
README.md	Added link to documentation in README	last month
pyproject.toml	Initial commit	last month
pytest.ini	Initial commit	last month

Click here to access the commit history for a repository

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | ···

main

Commits on Feb 26, 2023

Added link to documentation in README
borjasotomayor committed last month ✓ — e4feef8

Added badges to README
borjasotomayor committed last month ✓ — 747d08f

Minor update to GitHub Actions workflow
borjasotomayor committed last month ✓ — 7b40f72

Browse the repository at this point in the history

Added .readthedocs.yml
borjasotomayor committed last month ✓ — 569f19a

Initial commit
borjasotomayor committed last month ✓ — 1814bb6

Commits on Feb 9, 2023

First commit
borjasotomayor committed on Feb 9 — 143f3f6

Click here to see what the files in the repository looked like at a specific commit.

Later in the quarter…

We'll see how a single repository can be shared by multiple developers.

While having personal/individual repositories is common, where Git really shines is when multiple developers have to work on the same code.

Git is pretty good at reconciling conflicting versions of the code (what if two developers modify the same piece of code, and come up with different versions?)