Version Control, Git, and GitHub

The problem…

term_paper.doc

term_paper2.doc

term_paper2_updated.doc

termp_paper2_updated2.doc

Another problem…

x Google

Alice

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

Bob

lib.py

```
def div(a, b):
  return a/b
```

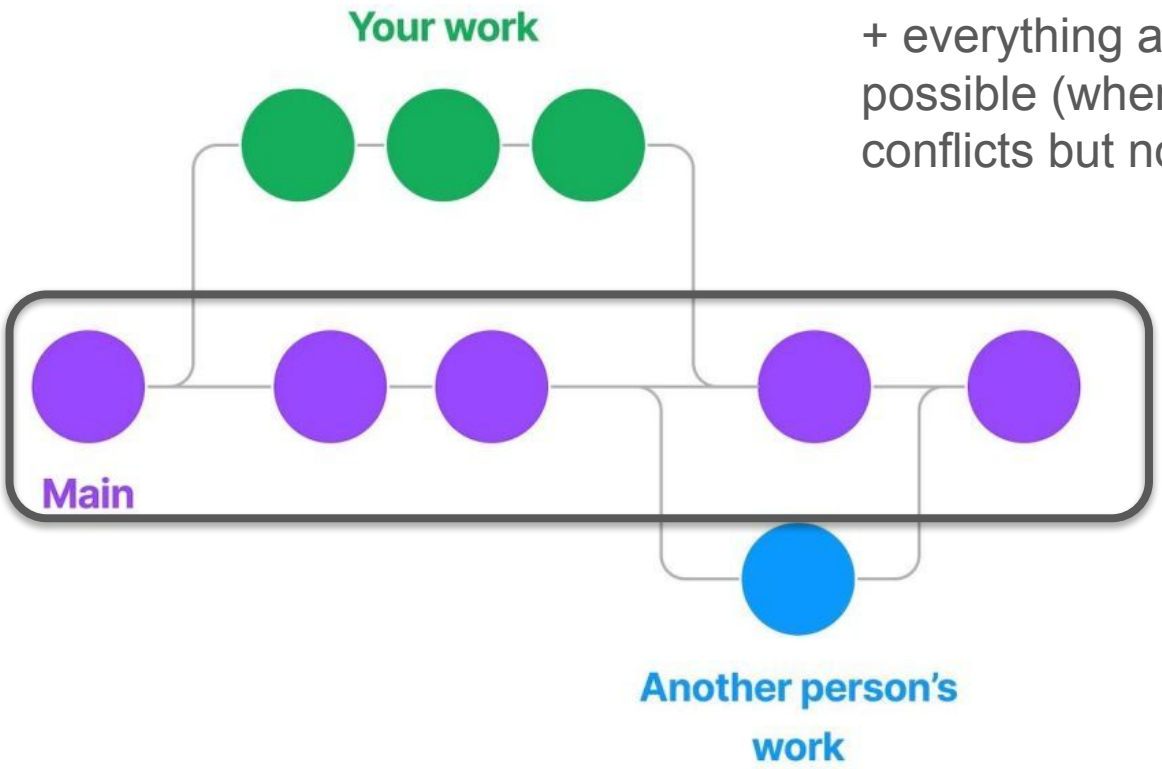
lib.py

```
def div(a, b):
  return a/b

def add(a, b):
  return a + b
```

calc.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None

def add(a, b):
  return a + b
```

What we'd like…



+ everything as fast, memory-light, and smart as possible (where smart=good at resolving merge conflicts but not at the expense of losing code)

Centralized repository

Version control software keeps track of changes to files in a project

```
def div(a, b):
  return a/b
```

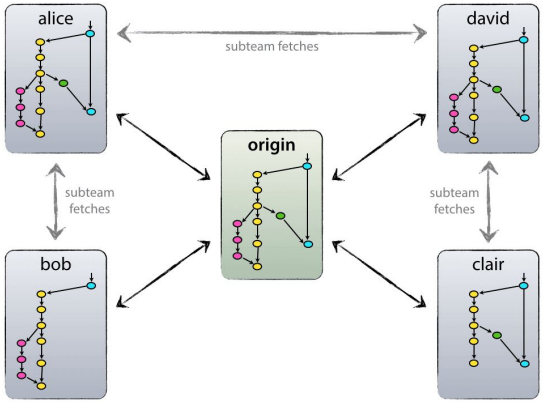
→

```
def div(a, b):
  if b != 0:
    return a/b
  else:
    return None
```

→

```
def div(a, b):
  if b != 0:
    return a/b
  else:
    return Inf
```

- allows reverting back to an old version
- allowing developers to test changes without losing the original
- synchronizing code between developers and users

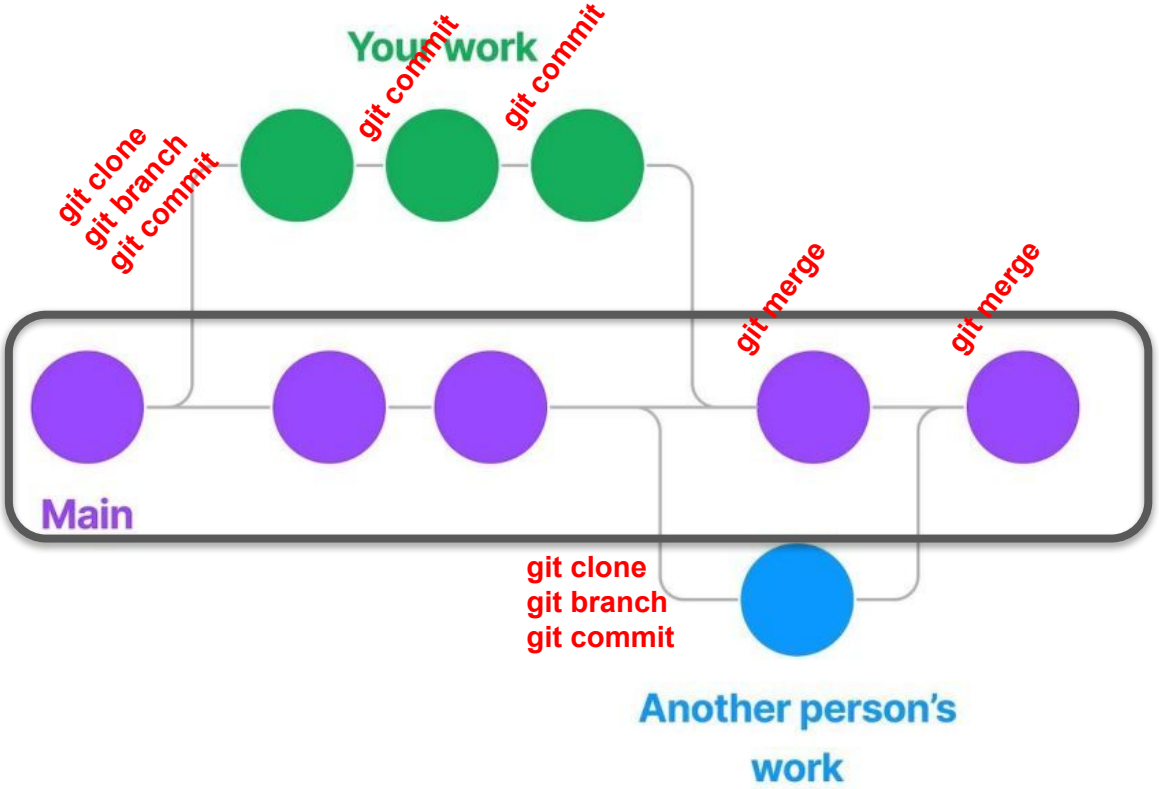


- tagging specific states of the codebase – marking them as important (e.g., the code that was released as version v1.2.3)

https://nvie.com/posts/a-successful-git-branching-model/

Version control software

Software repository hosting company

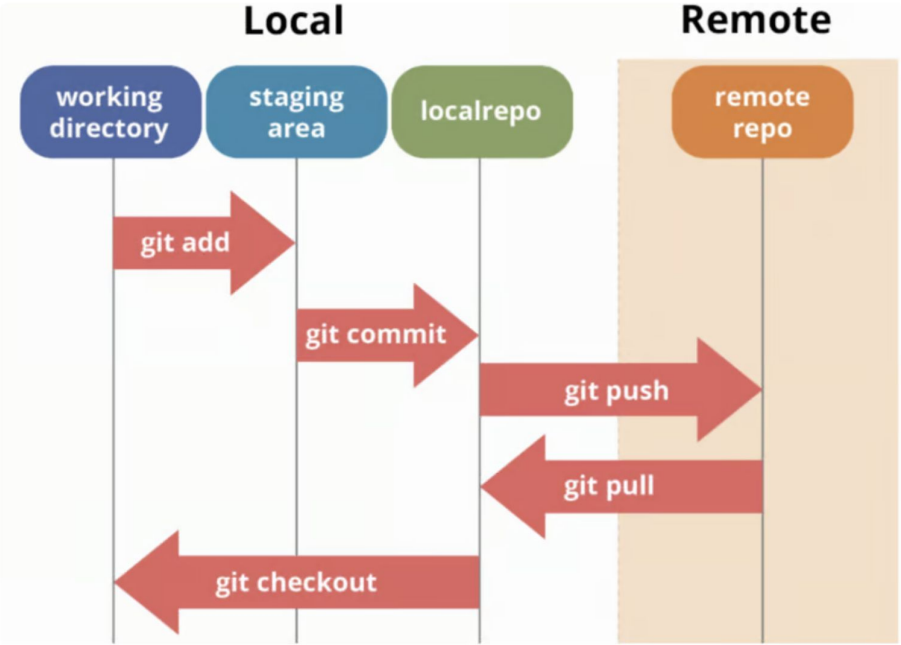


Centralized repository

GitHub

A history of version control

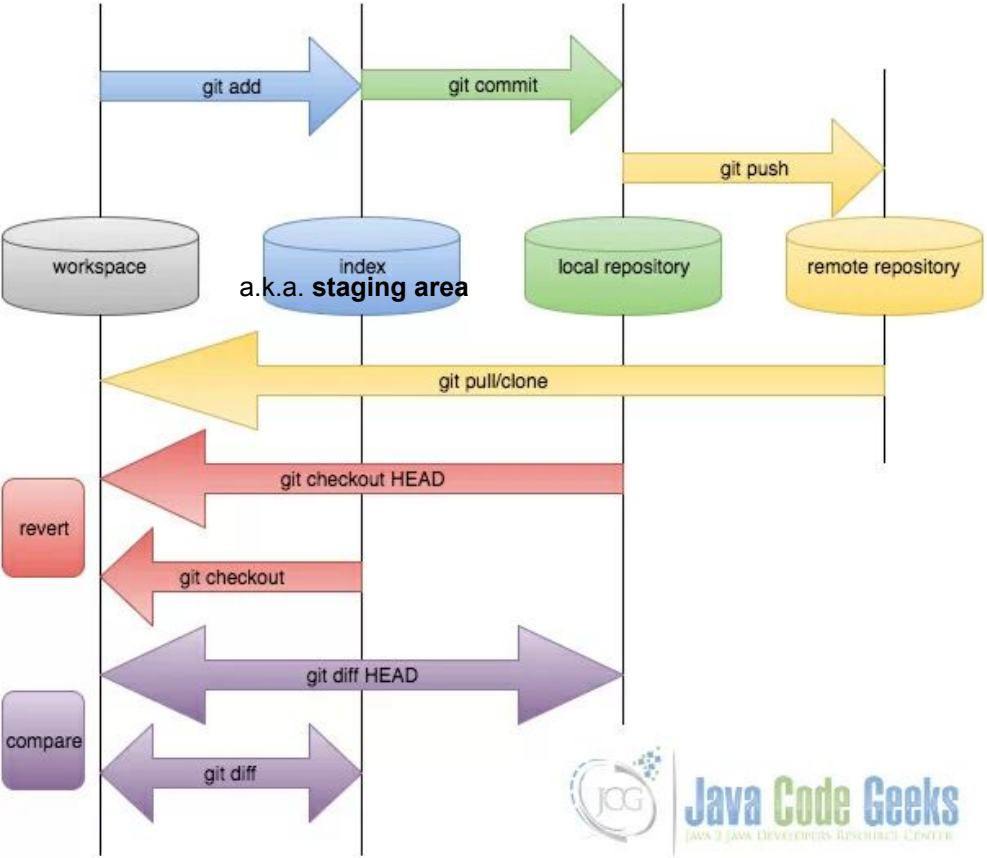
git workflow

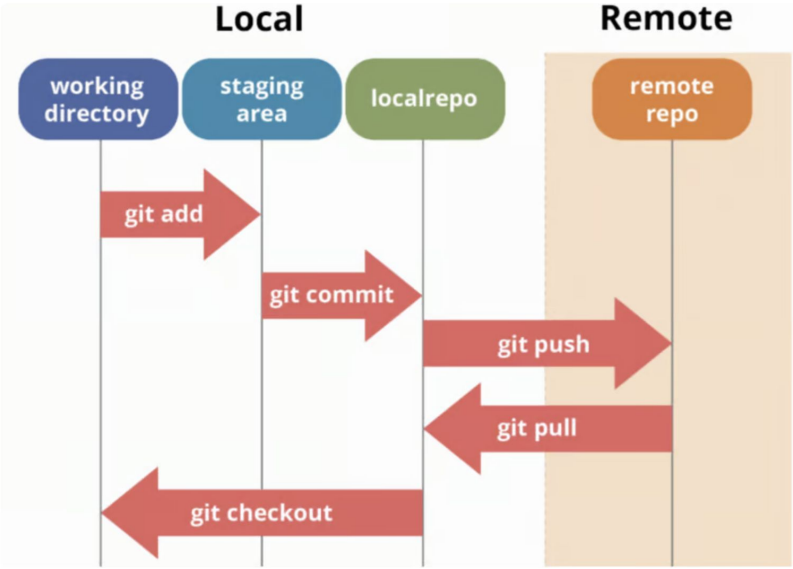


Why the staging area (a.k.a. “index”)? So you can include multiple changed files in the next commit for finer control of which codebase snapshots(=commits) are saved as checkpoints

Metaphor: Think of Git like photography

- **Working directory** = your messy photoshoot session (all the shots you’ve taken).
- **Staging area** = your "lightroom" where you select and polish the shots to include.
- **Commit** = pictures you add to the next page of your photo album


git add
git commit
git push
workspace
index
a.k.a. **staging area**
local repository
remote repository
git pull/clone
git checkout HEAD
revert
git checkout
git diff HEAD
compare
git diff
Java Code Geeks


Local
Remote
working directory
staging area
localrepo
remote repo
git add
git commit
git push
git pull
git checkout

git workflow example (without branching)

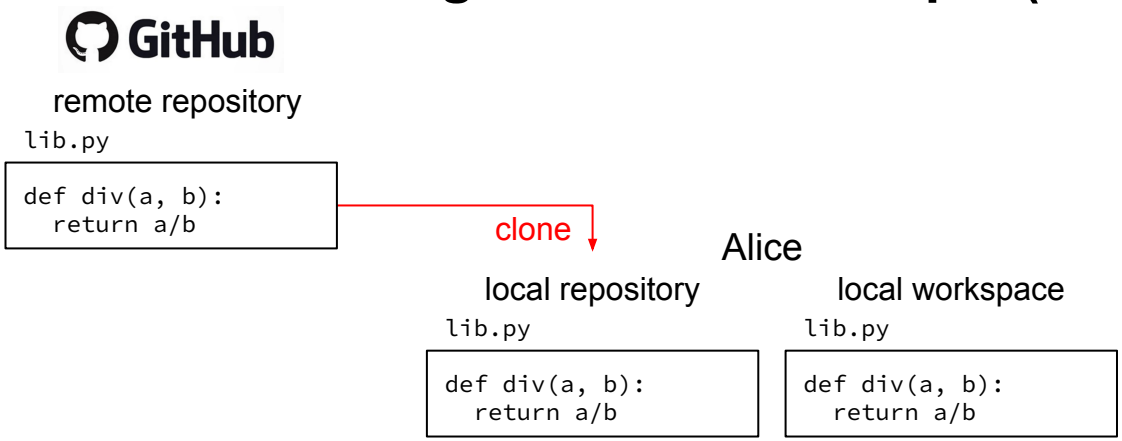
GitHub

remote repository

lib.py

```
def div(a, b):
  return a/b
```

git workflow example (without branching)



git workflow example (without branching)

GitHub

remote repository

lib.py

```
def div(a, b):
  return a/b
```

clone

Alice

local repository

lib.py

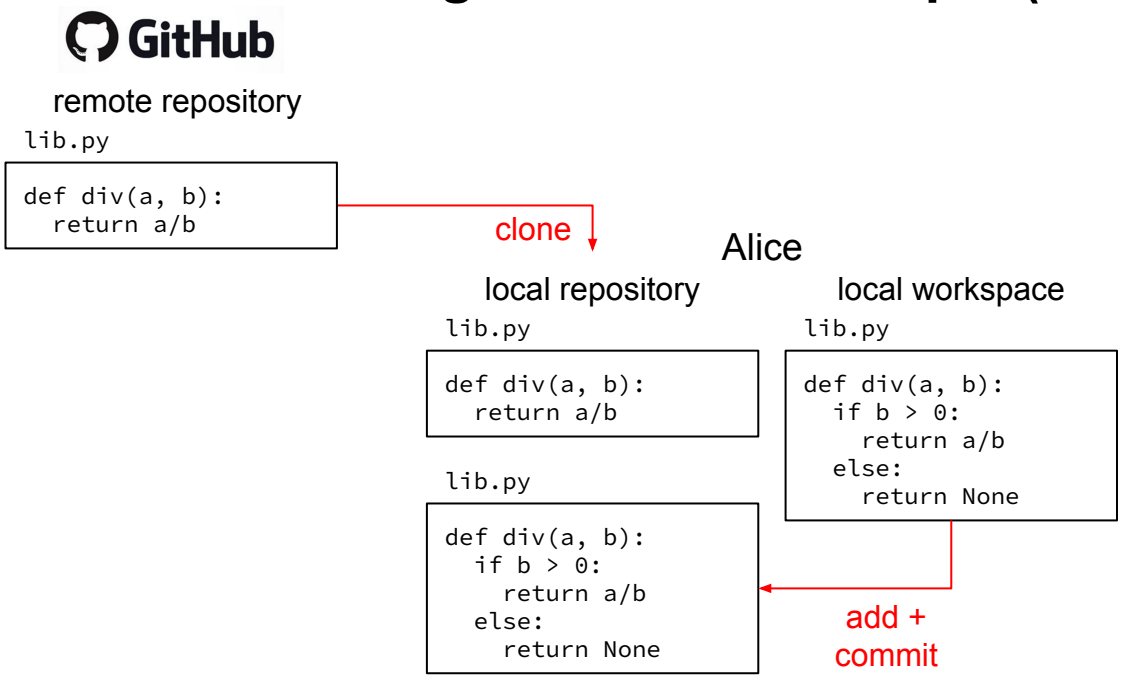
```
def div(a, b):
  return a/b
```

local workspace

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

git workflow example (without branching)



git workflow example (without branching)

GitHub

remote repository

lib.py

```
def div(a, b):
  return a/b
```

clone

Alice

local repository

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

local workspace

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

Bob

local repository

lib.py

```
def div(a, b):
  return a/b
```

local workspace

lib.py

```
def div(a, b):
  return a/b
```

git workflow example (without branching)

GitHub

remote repository

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

push

Alice

local repository

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

local workspace

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

Bob

local repository

lib.py

```
def div(a, b):
  return a/b
```

local workspace

lib.py

```
def div(a, b):
  return a/b
```

git workflow example (without branching)

GitHub

remote repository

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

push

Alice

local repository

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

local workspace

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

Bob

local repository

lib.py

```
def div(a, b):
  return a/b
```

local workspace

lib.py

```
def div(a, b):
  return a/b
```

git workflow example (without branching)

GitHub

remote repository

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

Alice

local repository

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

local workspace

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

Bob

local repository

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  return a/b

def add(a, b):
  return a + b
```

local workspace

lib.py

```
def div(a, b):
  return a/b

def add(a, b):
  return a + b
```

add + commit

git workflow example (without branching)

GitHub

remote repository

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

Alice

local repository

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

local workspace

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

Bob

local repository

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  return a/b

def add(a, b):
  return a + b
```

local workspace

lib.py

```
def div(a, b):
  return a/b

def add(a, b):
  return a + b
```

add + commit

git workflow example (without branching)

GitHub

remote repository

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

Alice

local repository

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

local workspace

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

Bob

local repository

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  return a/b

def add(a, b):
  return a + b
```

local workspace

lib.py

```
def div(a, b):
  return a/b

def add(a, b):
  return a + b
```

push

git workflow example (without branching)

GitHub

remote repository

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

Alice

local repository

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

local workspace

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

Bob

local repository

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  return a/b

def add(a, b):
  return a + b
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None

def add(a, b):
  return a + b
```

local workspace

lib.py

```
def div(a, b):
  return a/b

def add(a, b):
  return a + b
```

pull + fix merge conflicts

git workflow example (without branching)

GitHub

remote repository

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None

def add(a, b):
  return a + b
```

push

Alice

local repository

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

local workspace

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

Bob

local repository

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  return a/b

def add(a, b):
  return a + b
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None

def add(a, b):
  return a + b
```

local workspace

lib.py

```
def div(a, b):
  return a/b

def add(a, b):
  return a + b
```

	remote repository (usually GitHub)	local repository	local workspace
clone			
add			
commit			
push			
pull			

GitHub

remote repository

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None

def add(a, b):
  return a + b
```

`lib.py` commit history

remote repository

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None

def add(a, b):
  return a + b
```

setup.py

viz.py

run.py

repo commit history

GitHub

remote repository

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None

def add(a, b):
  return a + b
```

setup.py

viz.py

run.py

`$ git clone <repo>` grabs the version of each (tracked) file from the most recent commit

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None

def add(a, b):
  return a + b
```

setup.py

viz.py

run.py

remote repository

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None

def add(a, b):
  return a + b
```

setup.py

viz.py

run.py

Commit causes stability issues!

remote repository

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None

def add(a, b):
  return a + b
```

setup.py

viz.py

run.py

Commit causes stability issues!

GitHub

remote repository

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None

def add(a, b):
  return a + b
```

setup.py

viz.py

run.py

v1.0.0

Tag a commit to explicitly mark an important place in development

Commit causes stability issues!

GitHub

remote repository

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None
```

lib.py

```
def div(a, b):
  if b > 0:
    return a/b
  else:
    return None

def add(a, b):
  return a + b
```

setup.py

viz.py

run.py

v1.0.0

Tag a commit to explicitly mark an important place in development

Commit causes stability issues!

`$ git clone repo --branch v1.0.0` grabs the state of the repo that the tagged commit refers to

<u>Exercise</u>: for each of the git commands below, describe what the command does in terms of (1) the remote repository, (2) your local repository, and (3) your local workspace

- git clone
- git add
- git commit
- git push
- git pull

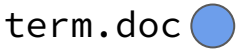
git branching

A **branch** is an independent line of development

(new file, changes to files)

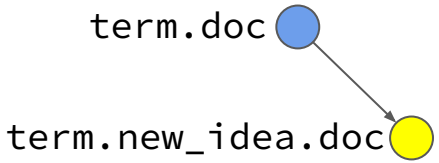
A **branch** is an independent line of development

(new file, changes to files)



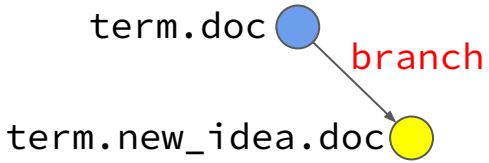
A **branch** is an independent line of development

(new file, changes to files)



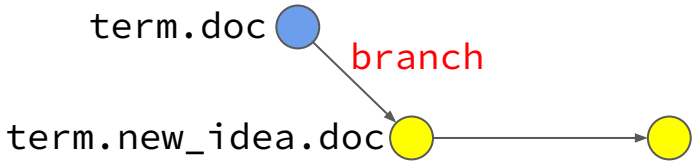
A **branch** is an independent line of development

(new file, changes to files)



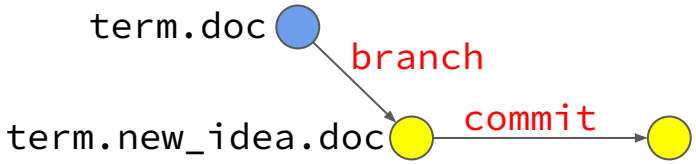
A **branch** is an independent line of development

(new file, changes to files)



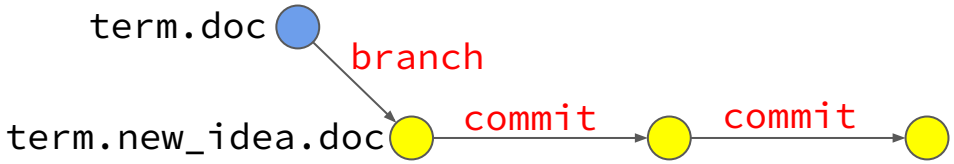
A **branch** is an independent line of development

(new file, changes to files)



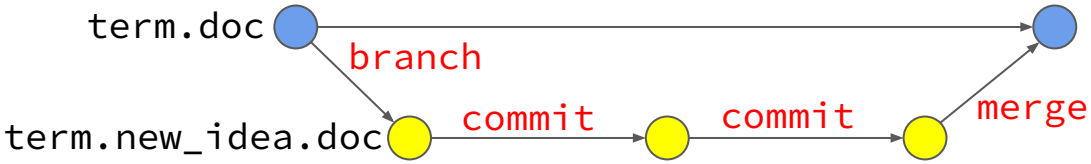
A **branch** is an independent line of development

(new file, changes to files)



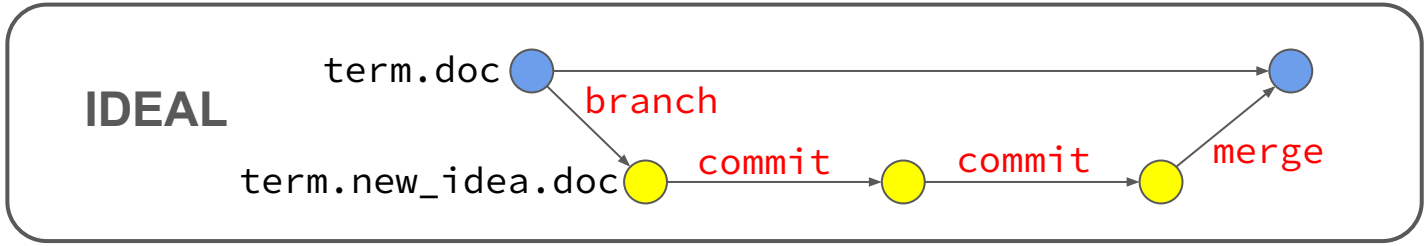
A **branch** is an independent line of development

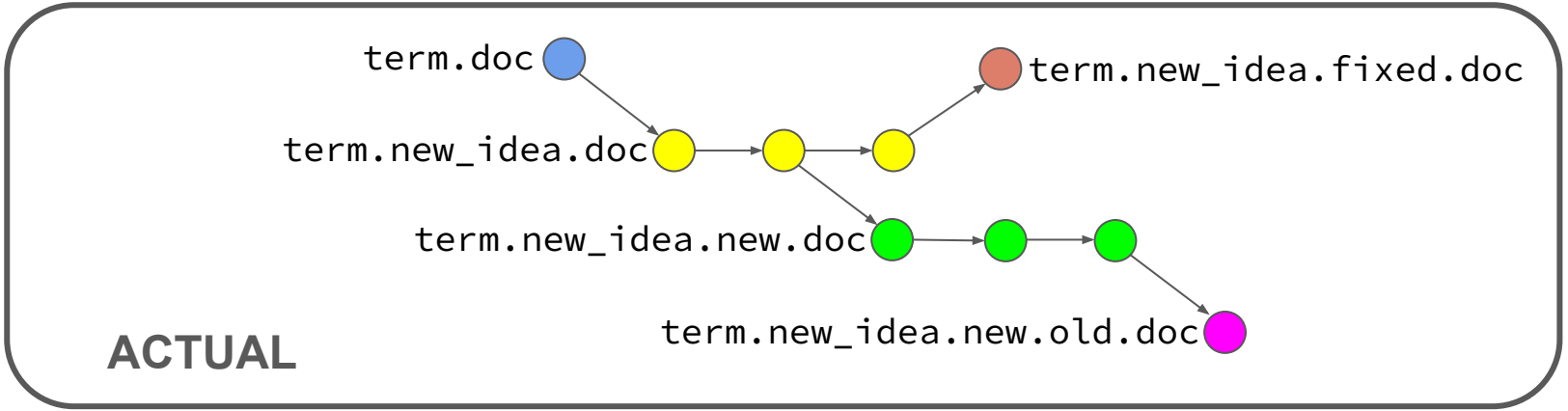
(new file, changes to files)



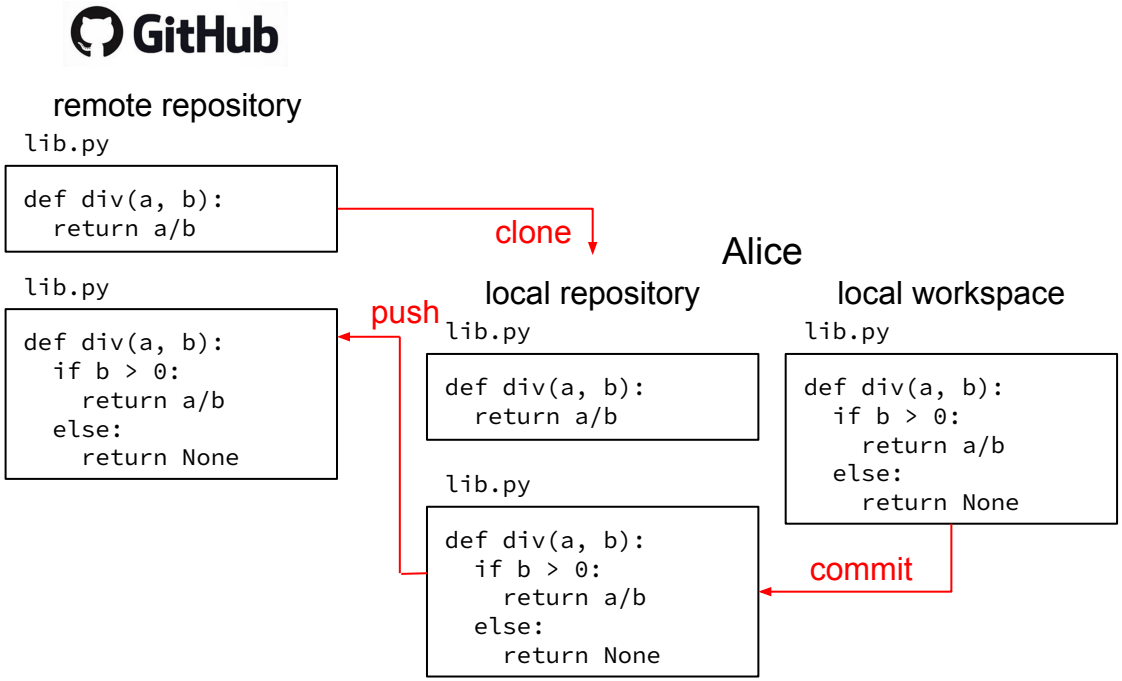
A **branch** is an independent line of development

(new file, changes to files)





git workflow example (without branching)



If everybody is working on branch main,

- prone to conflicts
- messy commit history in main

Example of recommended git workflow example with branching (without pull requests)

main

lib.py

```
def div(a, b):
  return a/b
```

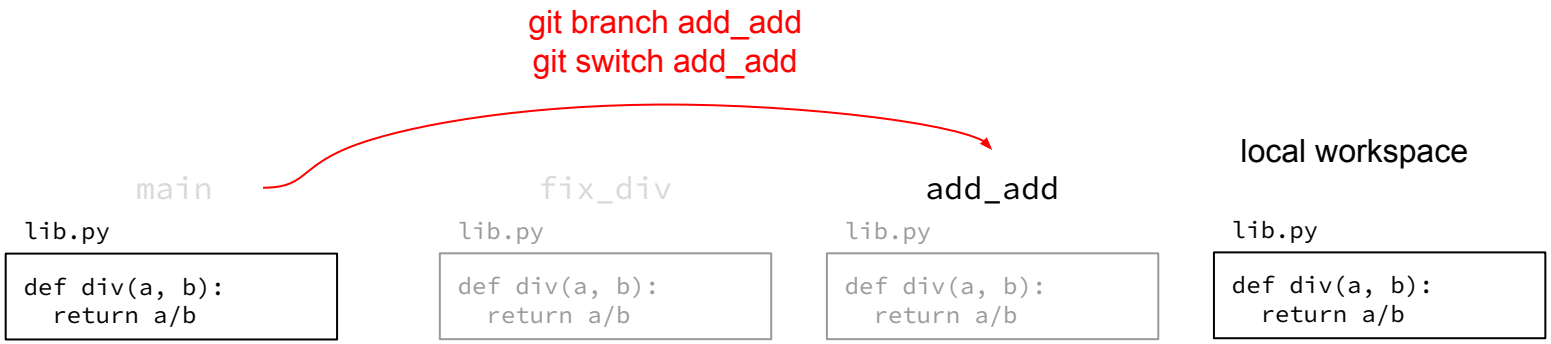
local workspace

lib.py

```
def div(a, b):
  return a/b
```

Example of recommended git workflow example with branching (without pull requests)

Example of recommended git workflow example with branching (without pull requests)



Example of recommended git workflow example with branching (without pull requests)

main

lib.py

```
def div(a, b):
  return a/b
```

fix_div

lib.py

```
def div(a, b):
  return a/b
```

add_add

lib.py

```
def div(a, b):
  return a/b
```

local workspace

lib.py

```
def div(a, b):
  return a/b

def add(a, b):
  return a + a
```

main

lib.py

```
def div(a, b):
  return a/b
```

fix_div

lib.py

```
def div(a, b):
  return a/b
```

add_add

lib.py

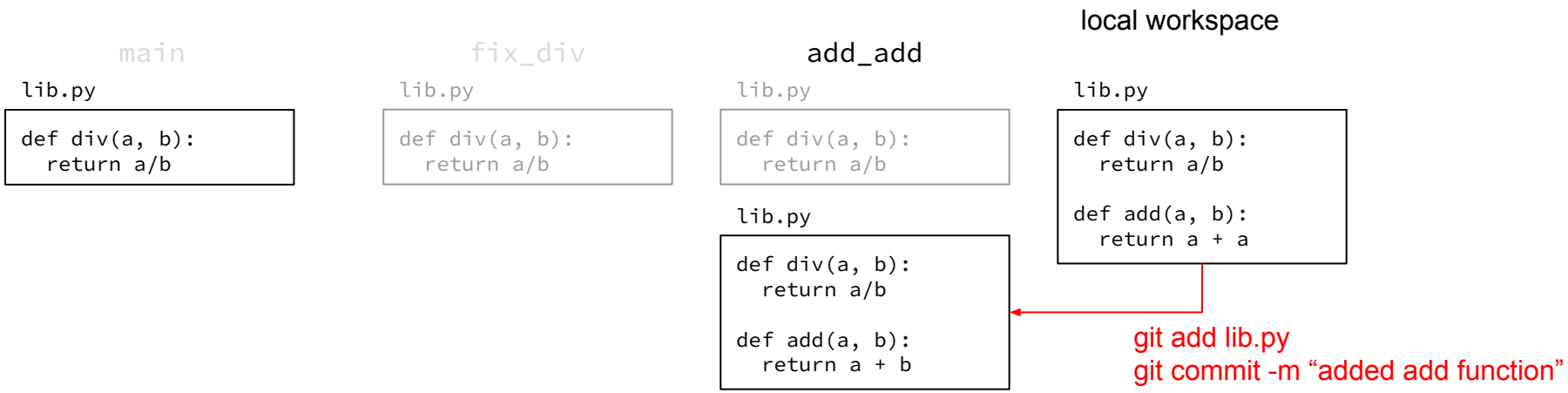
```
def div(a, b):
  return a/b
```

local workspace

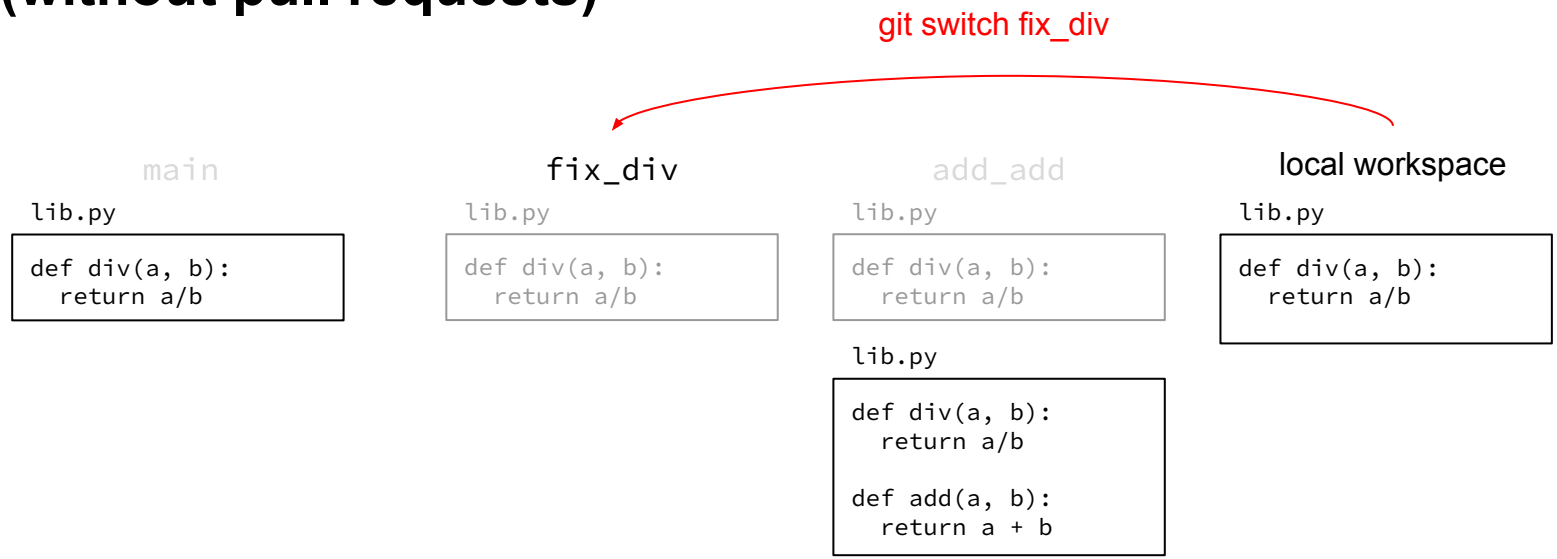
lib.py

```
def div(a, b):
  return a/b

def add(a, b):
  return a + a
```

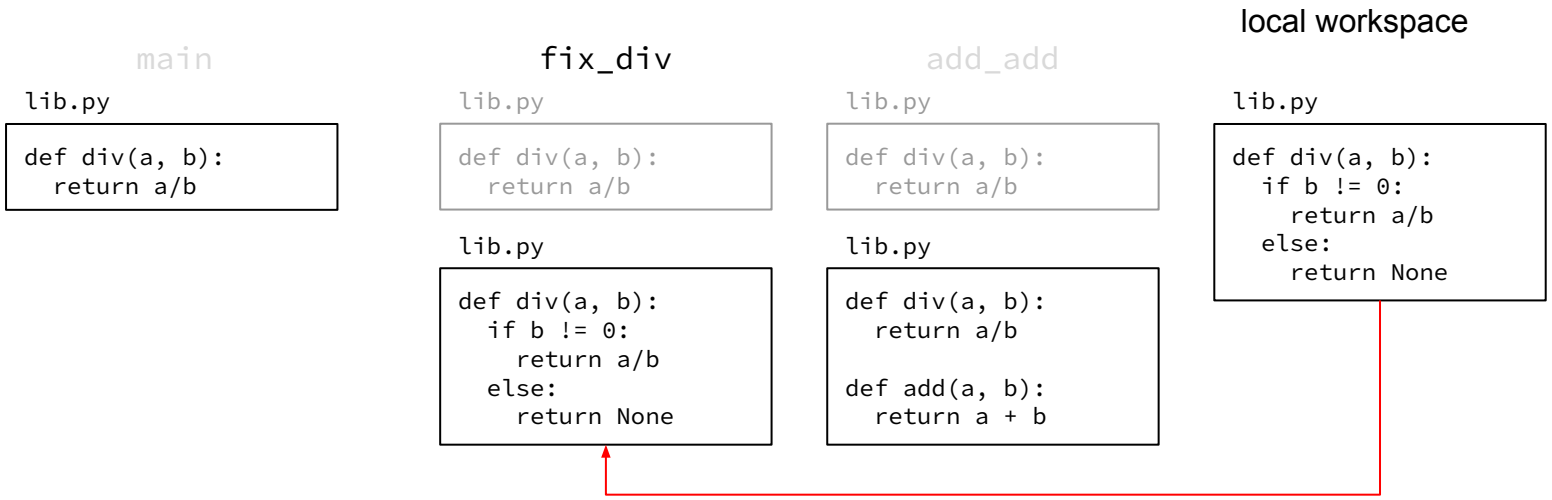
Example of recommended git workflow example with branching (without pull requests)



Example of recommended git workflow example with branching (without pull requests)

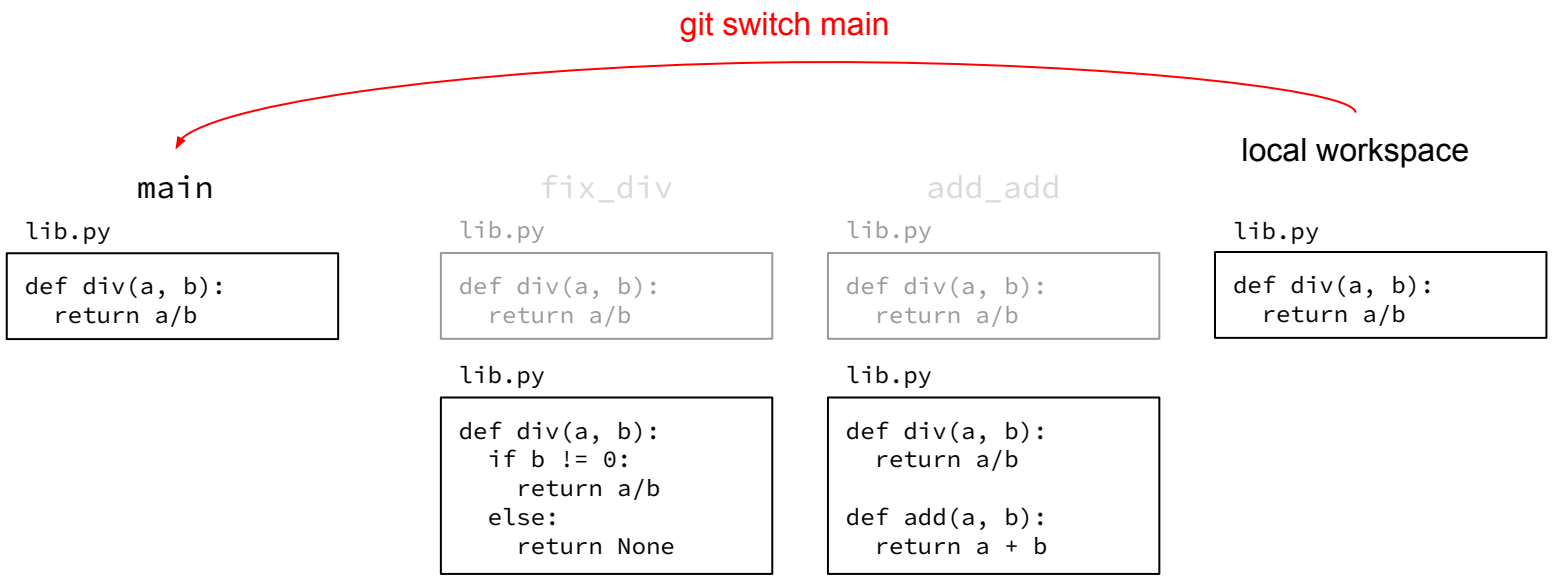


Example of recommended git workflow example with branching (without pull requests)

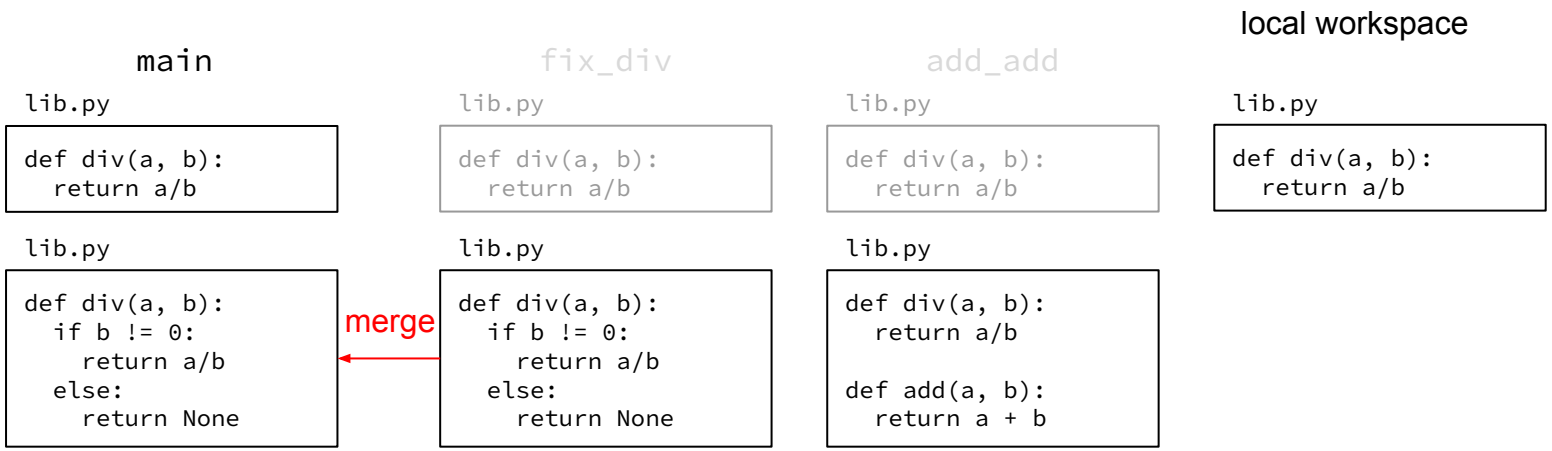


git add lib.py
git commit -m “fixed div function–support denominator being 0”

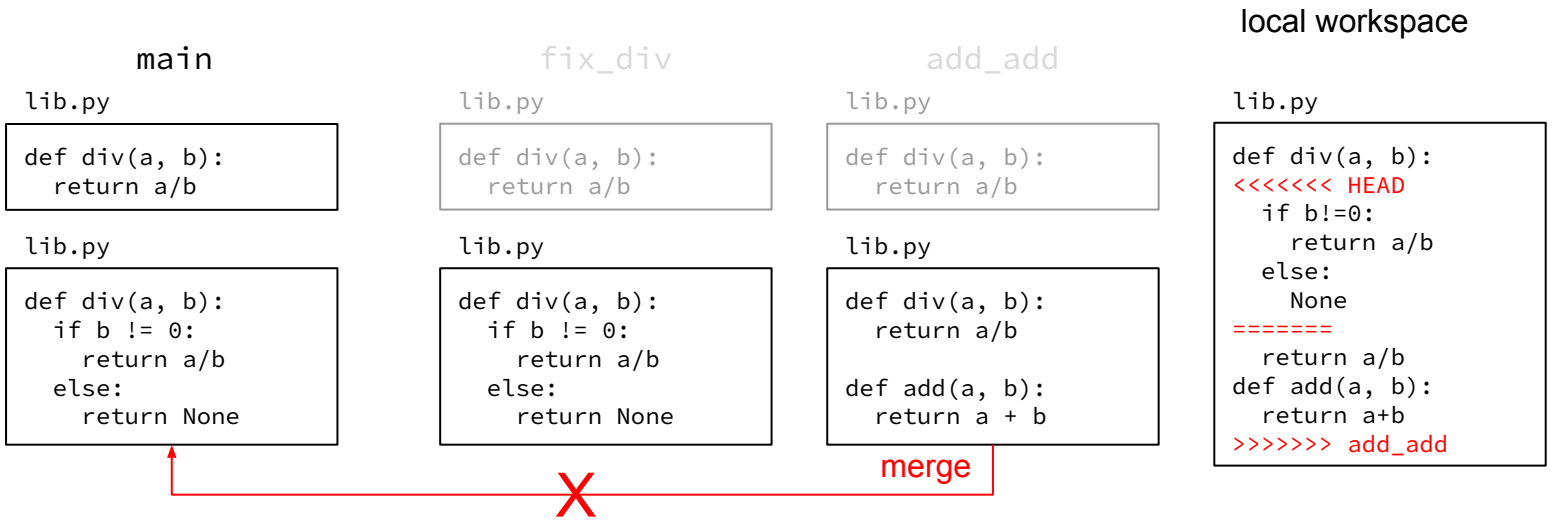
Example of recommended git workflow example with branching (without pull requests)



Example of recommended git workflow example with branching (without pull requests)



Example of recommended git workflow example with branching (without pull requests)



Example of recommended git workflow example with branching (without pull requests)

main

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  if b != 0:
    return a/b
  else:
    return None
```

lib.py

```
def div(a, b):
  if b!=0:
    return a/b
  else:
    None

def add(a, b):
  return a+b
```

fix_div

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  if b != 0:
    return a/b
  else:
    return None
```

add_add

lib.py

```
def div(a, b):
  return a/b
```

lib.py

```
def div(a, b):
  return a/b

def add(a, b):
  return a + b
```

local workspace

lib.py

```
def div(a, b):
<<<<<<< HEAD
  if b!=0:
    return a/b
  else:
    None
=======
  return a/b
def add(a, b):
  return a+b
>>>>>>> add_add
```

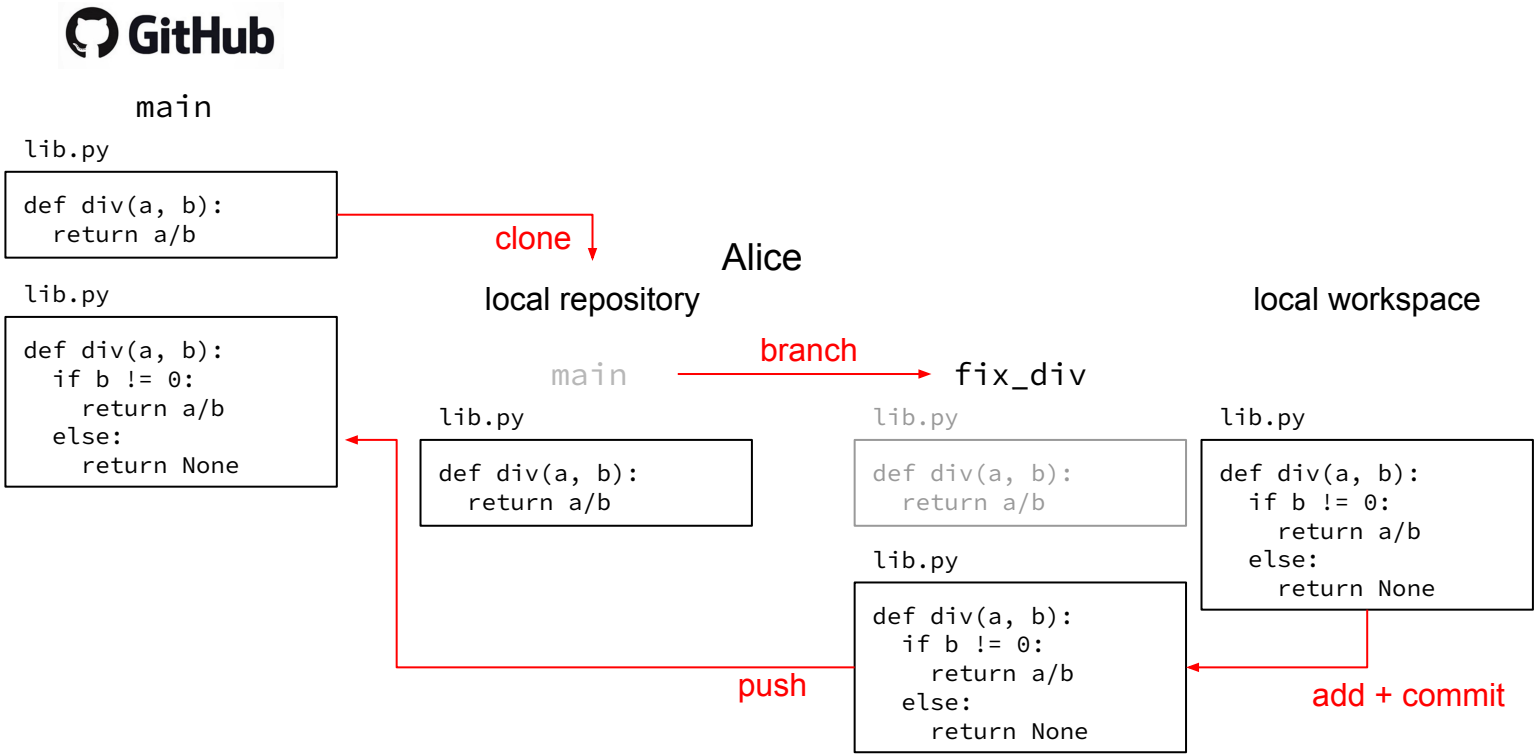
lib.py

```
def div(a, b):
  if b!=0:
    return a/b
  else:
    None

def add(a, b):
  return a+b
```

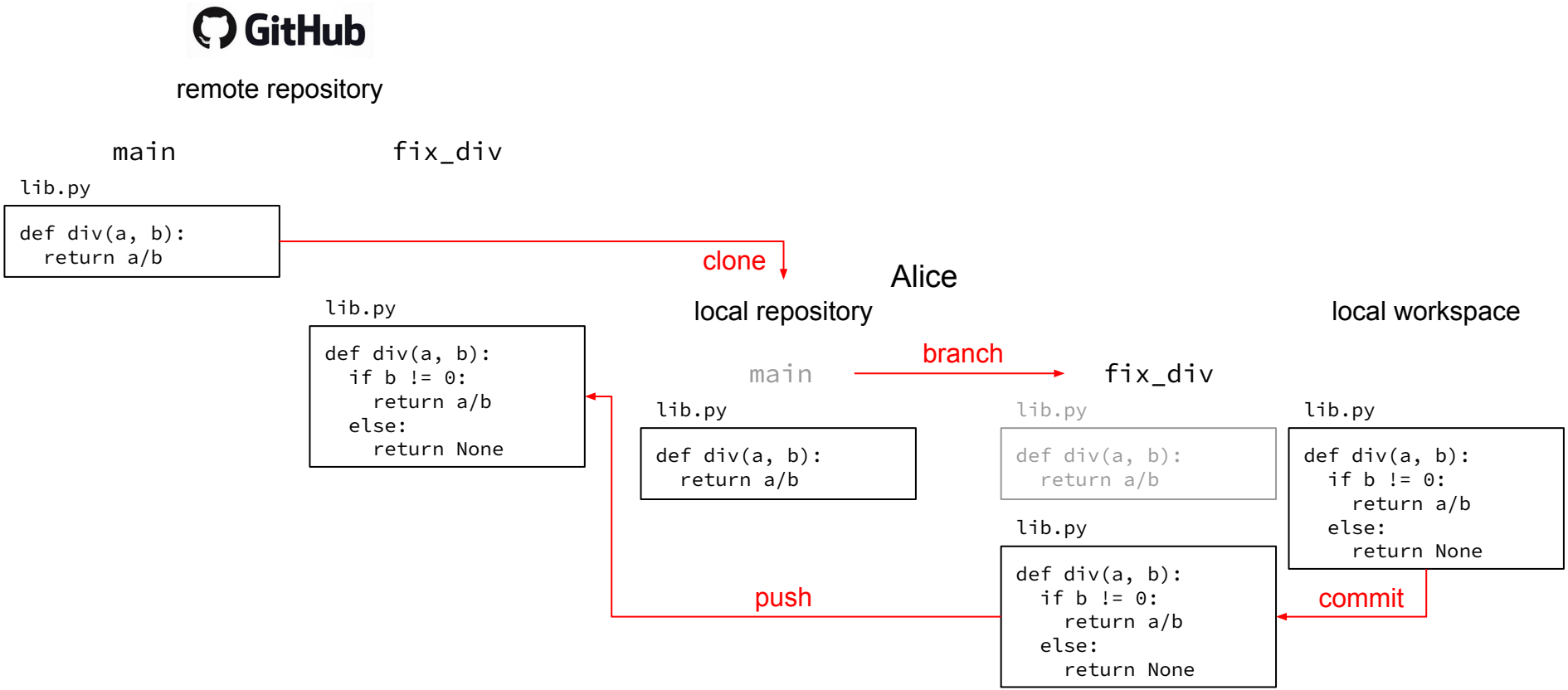
commit

Pull requests

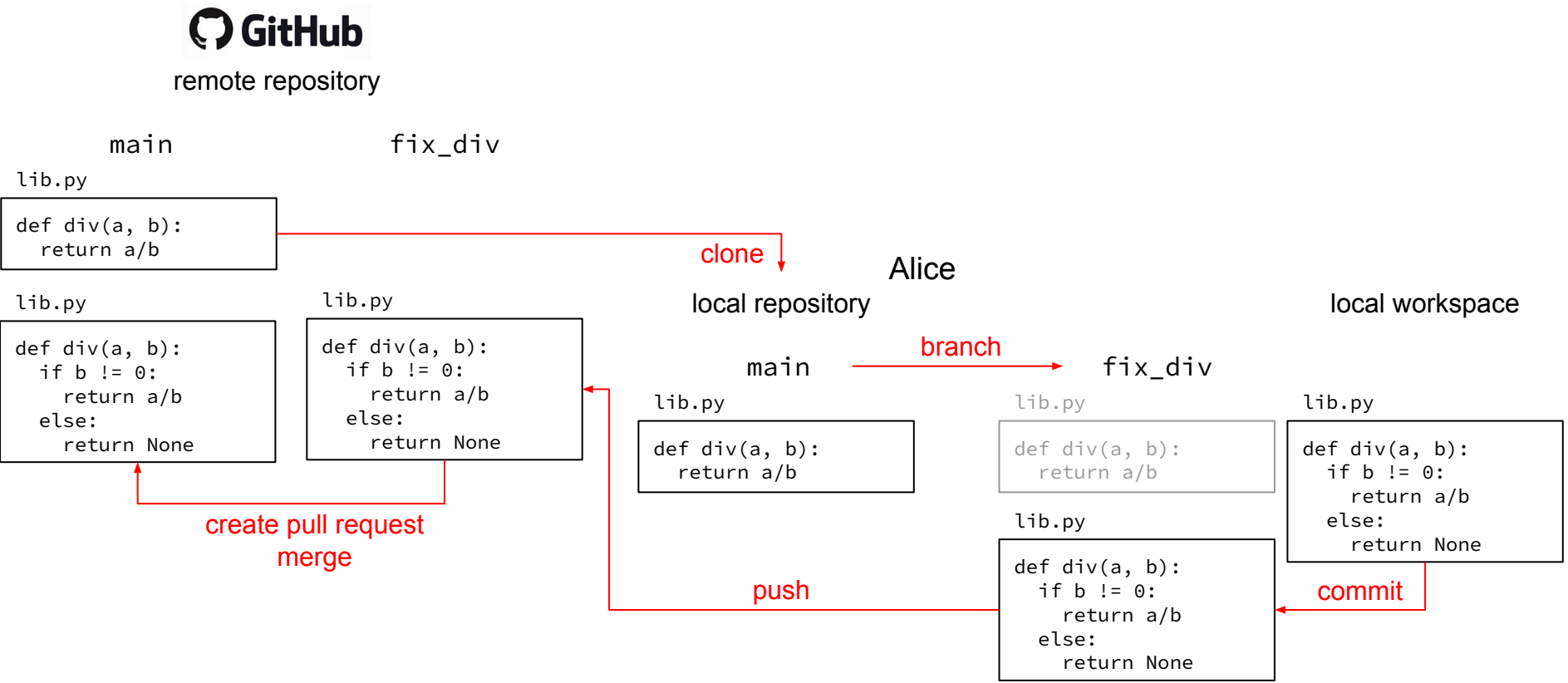
Example of recommended git workflow example with branching (without pull requests)



Example of recommended git workflow example with branching and with pull requests



Example of recommended git workflow example with branching and with pull requests



Some common & useful git commands

- git clone
- git add
- git commit
- git push
- git pull
- git fetch
- git log (--oneline --graph)

Related stuff you can find on the course webpage

- Notes on authenticating with GitHub via SSH keys: “Using SSH Keys with GitHub”
- Similar material as these slides but in write-up/notes form: “Version Control, Git, and GitHub”

The End