Git

COS 333 Tutorial

Problem Statement

```python
def run_application():
    num1 = input()
    num2 = input()
    print(f'You entered {num1} and {num2}')

```

David's work

```python
def run_application():
    num1 = input()
    num2 = input()
    print(f'You entered {num1} and {num2}')
    print(f'Their sum is {sum_two(num1, num2)}')

def sum_two(num1: int, num2: int):
    return num1+ num2

```

Bob's work

```python
def run_application():
    num1 = input()
    num2 = input()
    print(f'You entered {num1} and {num2}')
    print(f'Their product is{multiply_two(num1, num2)}')

def multiply_two(num1: int, num2: int):
    return num1 * num2

```

Problem Statement

David's work

file1.py

```python
def run_application():
    num1 = input()
    num2 = input()
    print(f'You entered {num1} and {num2}')
    print(f'Their sum is {sum_two(num1, num2)}')

def sum_two(num1: int, num2: int):
    return num1+ num2
```

file1.py

```python
def run_application():
    num1 = input()
    num2 = input()
    print(f'You entered {num1} and {num2}')
    print(f'Their product is{multiply_two(num1, num2)}')

def multiply_two(num1: int, num2: int):
    return num1 * num2
```

Bob's work

MERGE

file1.py

```python
def run_application():
    num1 = input()
    num2 = input()
    print(f'You entered {num1} and {num2}')
    print(f'Their sum is {sum_two(num1, num2)}')
    print(f'Their product is{multiply_two(num1, num2)}')

def multiply_two(num1: int, num2: int):
    return num1 * num2

def sum_two(num1: int, num2: int):
    return num1 + num2
```

Problems with merging like this?

Problems with merging like this?

01

It takes way too much time

If the codebase was indeed dozens of files, merging work of up to five people working on different files would not scale.

Problems with merging like this?

01

It takes way too much time

If the codebase was indeed dozens of files, merging work of up to five people working on different files would not scale.

02

Our brains don’t remember everything

Collaborators might delete piece of code, change small things, or overwrite each other’s work. A lot can be forgotten during merge.

Problems with merging like this?

01

It takes way too much time

If the codebase was indeed dozens of files, merging work of up to five people working on different files would not scale.

02

Our brains don't remember everything

Collaborators might delete piece of code, change small things, or overwrite each other's work. A lot can be forgotten during merge.

03

Features might depend on each other

On a large team, student A may need to work on something that Student B is making. However, student A's work may also change something that students C, D, and E's code depends on. Resolving manually becomes a nightmare.

Solution: git

Solution: git

01 Tracks file changes automatically

Solution: git

01 Tracks file changes automatically

02 Allows you to view full codebase history

Solution: git

01 Tracks file changes automatically

02 Allows you to view full codebase history

03 Merges files automatically, only asking for manual input if there are conflicts

So what is git?

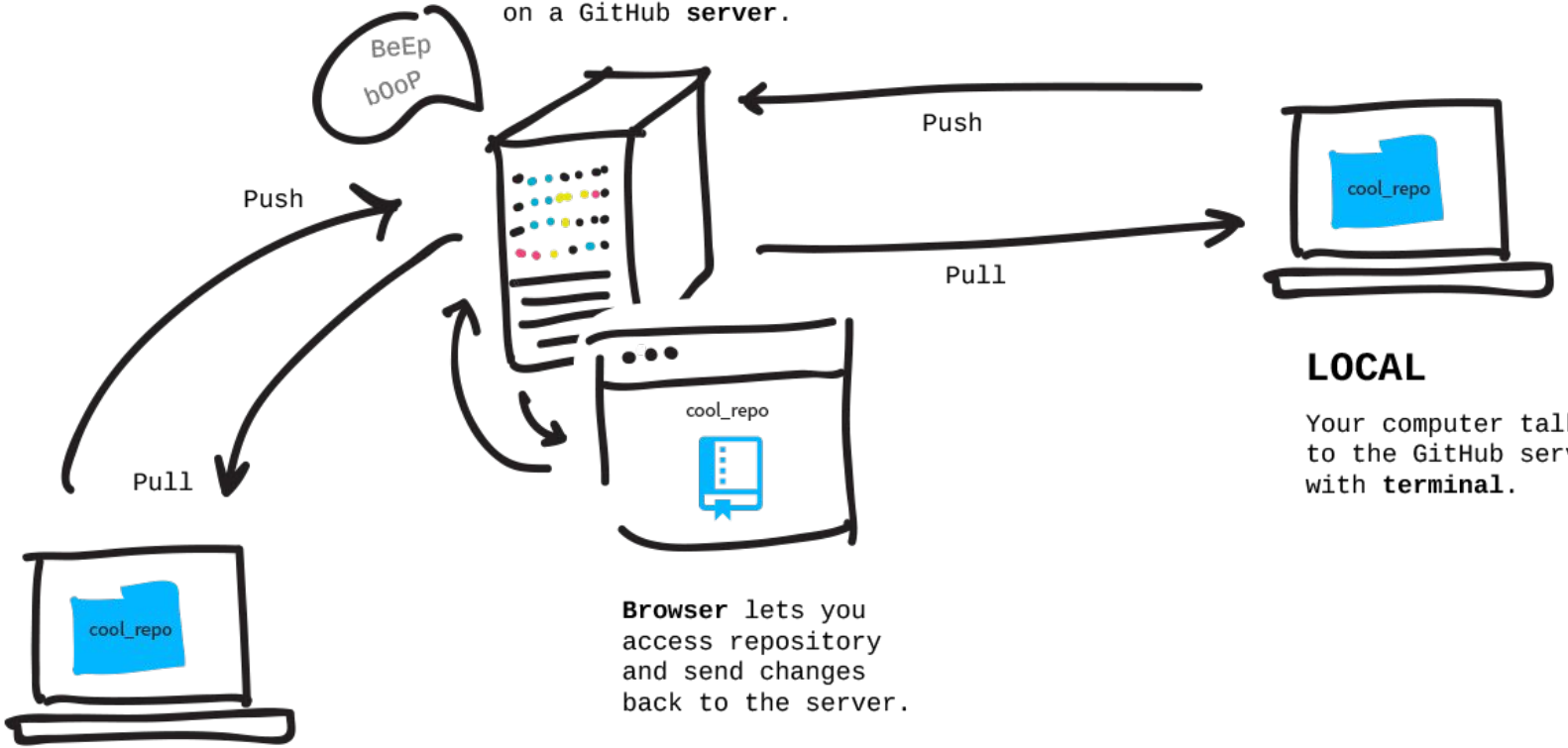
git is a distributed version control system, a piece of software that tracks all file changes within a specified directory. It likewise enables file version management through branches, history, and merging.

So what is GitHub?

GitHub is simply a cloud service that stores a git repository for you. From there, you can work on the repository collaboratively with other people, enjoying the benefits of automated file change tracking, branching, automated merging, and other more advanced features.

REMOTE

Repositories live
on a GitHub **server**.



LOCAL

Your computer talks
to the GitHub server
with **terminal**.

Browser lets you
access repository
and send changes
back to the server.

LOCAL

Someone else’s computer
talks to the GitHub server.

Live Exercise

- Installing and configuring git
- Creating a GitHub account
- Creating and initializing a git repository with a README on GitHub
- Cloning the repository onto your local machine
- Adding, committing, and pushing Assignment 0 files to remote
- Deleting files
- Branching and merging

Live Exercise

- Installing and configuring git (see git/github primer)

Live Exercise

- Creating a GitHub account

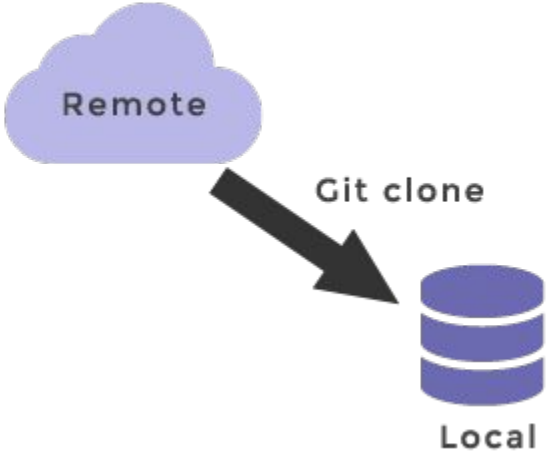
Live Exercise

- Creating and initializing a git repository with a README on GitHub

Pair up with another student and pick whose repository you will be using for the rest of the exercise. Then on the chosen repository, add your partner as a collaborator.

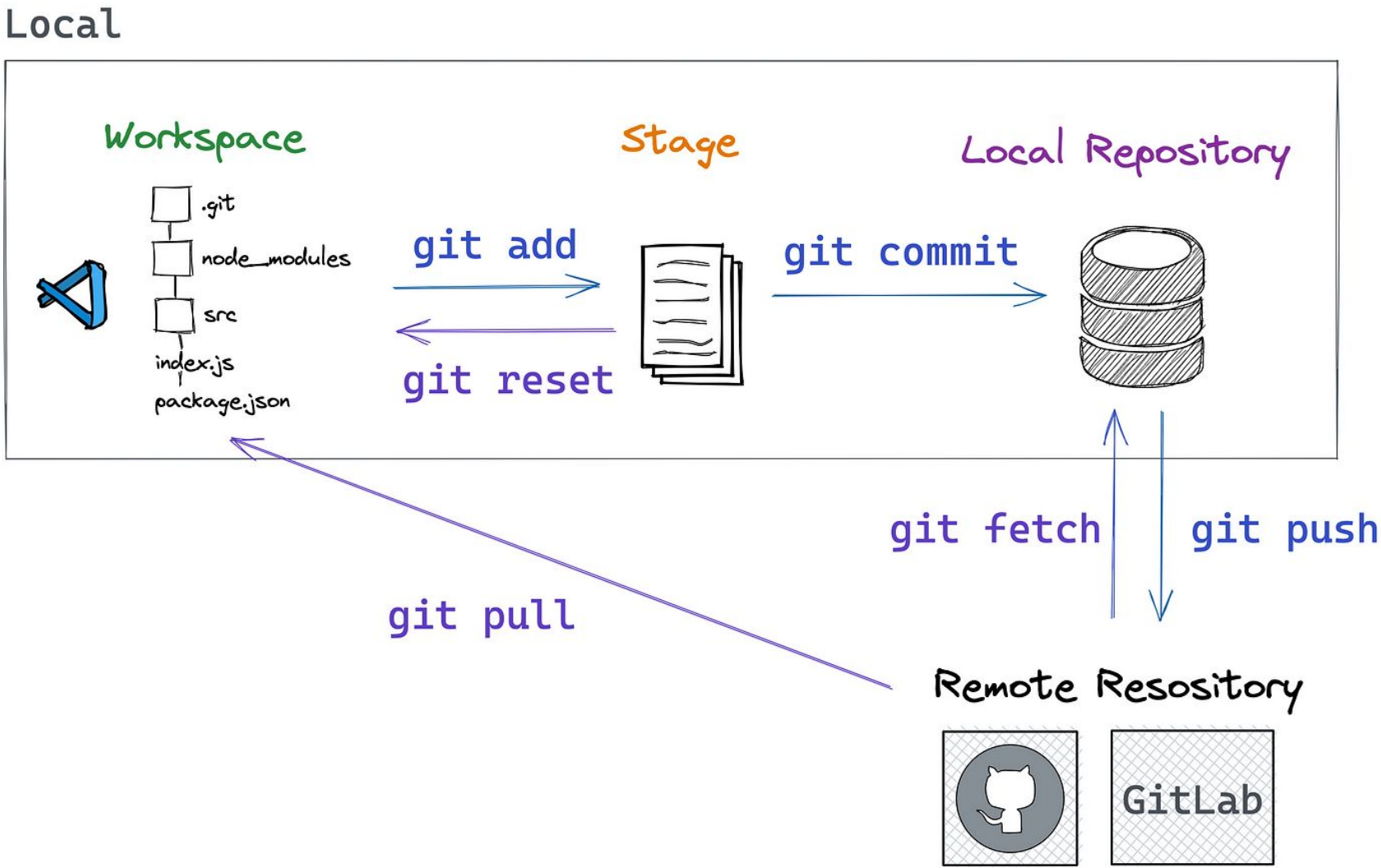
Live Exercise

- Cloning the repository onto your local machine



Live Exercise

- Adding, committing, and pushing Assignment 0 files to remote


Local
Workspace
.git
node_modules
src
index.js
package.json
Stage
Local Repository
git add
git reset
git commit
git fetch
git push
git pull
Remote Resosiory
GitLab

Live Exercise

- Adding, committing, and pushing Assignment 0 files to remote

1) Student 1 creates a file “file1.py” and pushes it up to remote
2) Student 2 creates a file “file2.py” and pushes it up to remote
    a) You might see an error that informs you that changes have been made to the remote repository. To update your local repository with those changes, use “git pull”

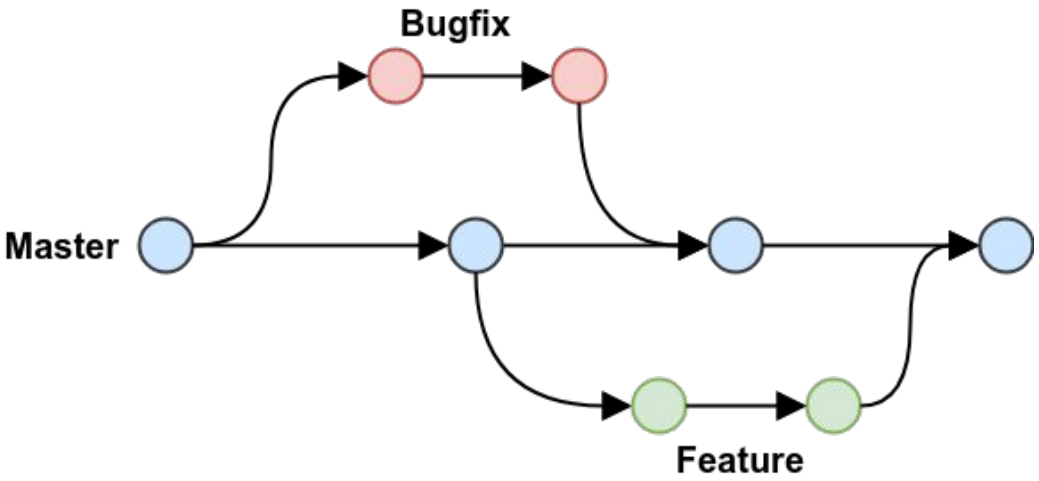
Live Exercise

- Deleting files

1) Student 1 deletes “file2.py” from the repository
2) Student 2 runs “git pull” to reflect those changes locally

Live Exercise

- Branching and merging



Live Exercise

- Branching and merging

1) Both students “git checkout -b <feature-name>” and edit the remaining “file1.py” locally, then push the changes up to their respective branches.

file1.py

```python
def run_application():
    num1 = input()
    num2 = input()
    print(f'You entered {num1} and {num2}')
    print(f'Their sum is {sum_two(num1, num2)}')

def sum_two(num1: int, num2: int):
    return num1+ num2
```

file1.py

```python
def run_application():
    num1 = input()
    num2 = input()
    print(f'You entered {num1} and {num2}')
    print(f'Their product is{multiply_two(num1, num2)}')

def multiply_two(num1: int, num2: int):
    return num1 * num2
```

Live Exercise

- Branching and merging

1) Both students “git checkout -b <feature-name>” and edit the remaining “file1.py” locally, then push the changes up to their respective branches.
2) Run “git fetch” to make new branches visible locally

Live Exercise

- Branching and merging

1) Both students “git checkout -b <feature-name>” and edit the remaining “file1.py” locally, then push the changes up to their respective branches.
2) Run “git fetch” to make new branches visible locally
3) Go on GitHub and create PRs from the respective branches. Attempt to merge into master.

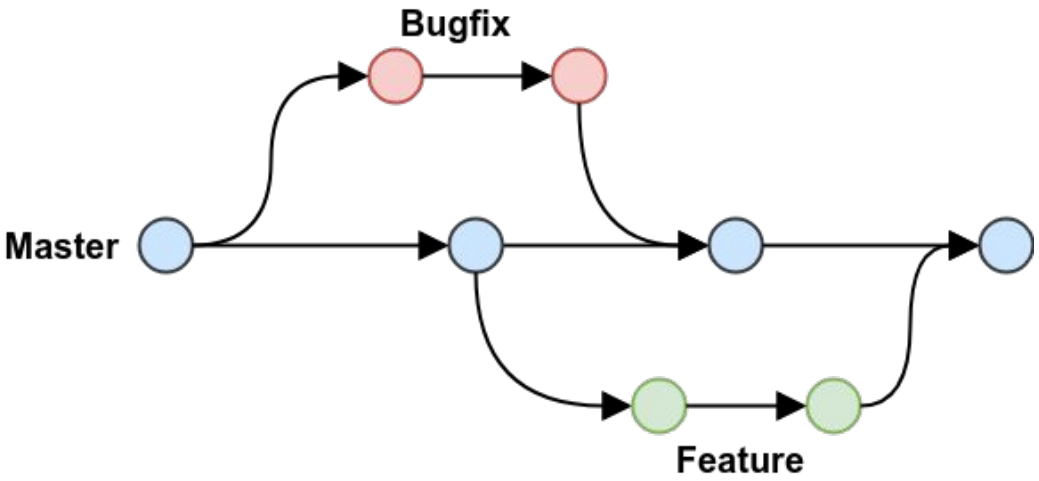
Live Exercise

- Branching and merging

1) Both students “git checkout -b <feature-name>” and edit the remaining “file1.py” locally, then push the changes up to their respective branches.
2) Run “git fetch” to make new branches visible locally
3) Go on GitHub and create PRs from the respective branches. Attempt to merge into master.
4) Delete your feature branch since the feature is now complete.

Live Exercise

- Merging master into feature branch



Live Exercise

- Merging master into feature branch

1) Student 2 creates a new branch `feature-3`

Live Exercise

- Merging master into feature branch

1) Student 2 creates a new branch `feature-3`
2) Student 1 makes a change on master and pushes it up to remote

Live Exercise

- Merging master into feature branch

1) Student 2 creates a new branch `feature-3`
2) Student 1 makes a change on master and pushes it up to remote
3) Student 2 wants to have the change on master reflected on his feature branch

Live Exercise

- Merging master into feature branch

1) Student 2 creates a new branch `feature-3`
2) Student 1 makes a change on master and pushes it up to remote
3) Student 2 wants to have the change on master reflected on his feature branch

git fetch origin master OR git checkout master && git pull && git checkout feature-3
git merge master

Resources

- The Online Git Book
- Bob’s Git / GitHub Reference
- Documentation
- Atlassian Git Tutorial
- OhShitGit