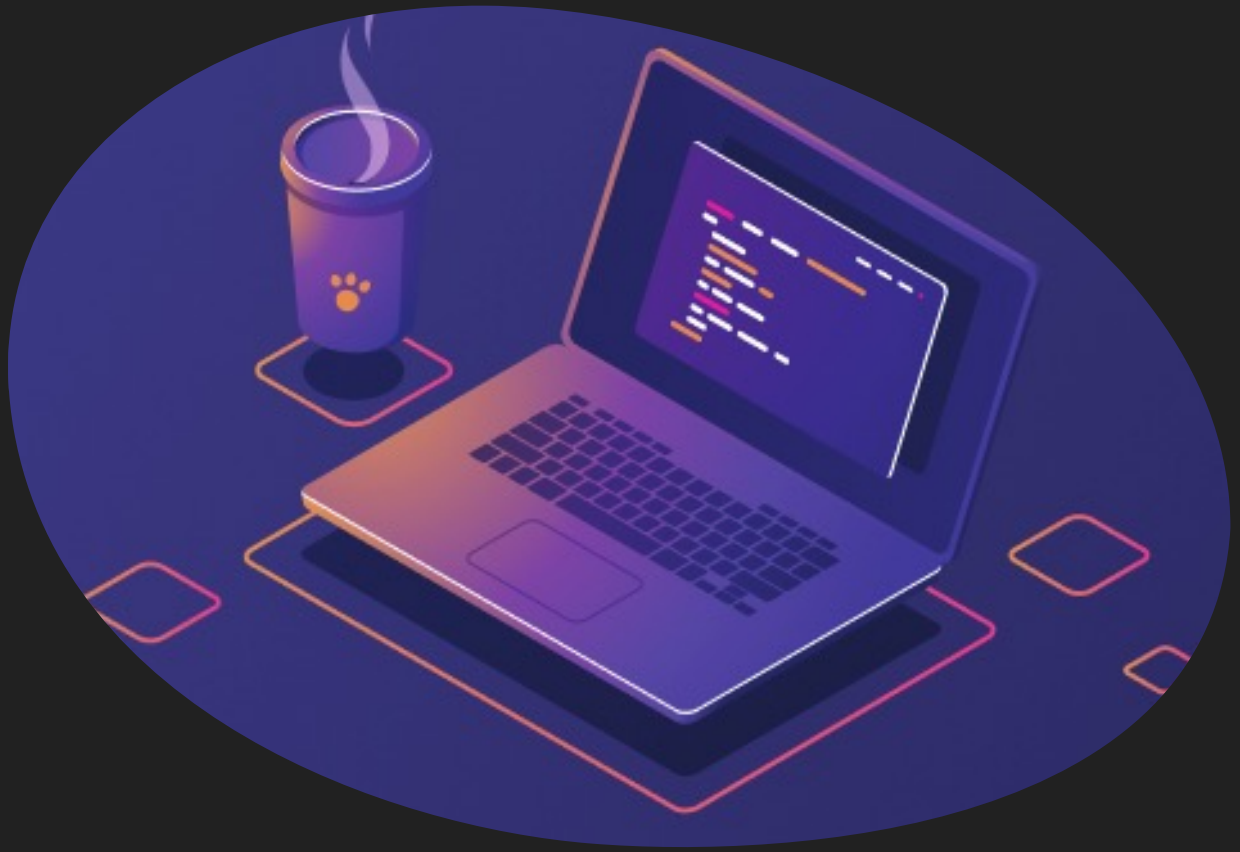


Week 2 : SOFTWARE DEVELOPMENT TOOLS AND ENVIRONMENTS

Week2 Getting Started with Git

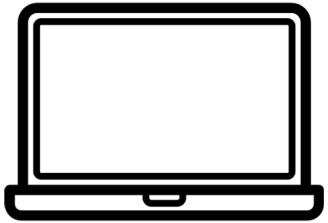
- **Week2 Commands:**
  - Changing code in a Repository
    - **git add**
  - Committing these changes
    - **git commit**
  - Pushing or Pulling Changes
    - **git push** and **git pull**
  - Checking Status, Logs, and Changes
    - **git status**, **git log**, **git diff**

Week 2
Basic Git Usage

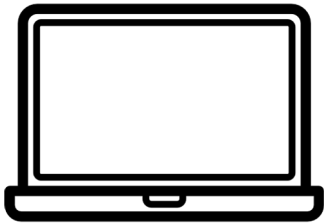
Week 2 Getting Started with Git

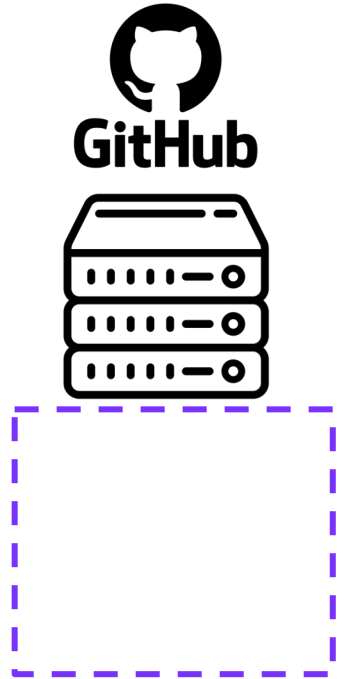
- **Basic Git Usage**
  - Let's cover the basic cycle of a workflow of using Git and GitHub.
  - This particular basic example will assume just a solo developer and everything working on the same branch.
  - We'll cover branches and working with others on Week 3.

Week 2 Getting Started with Git

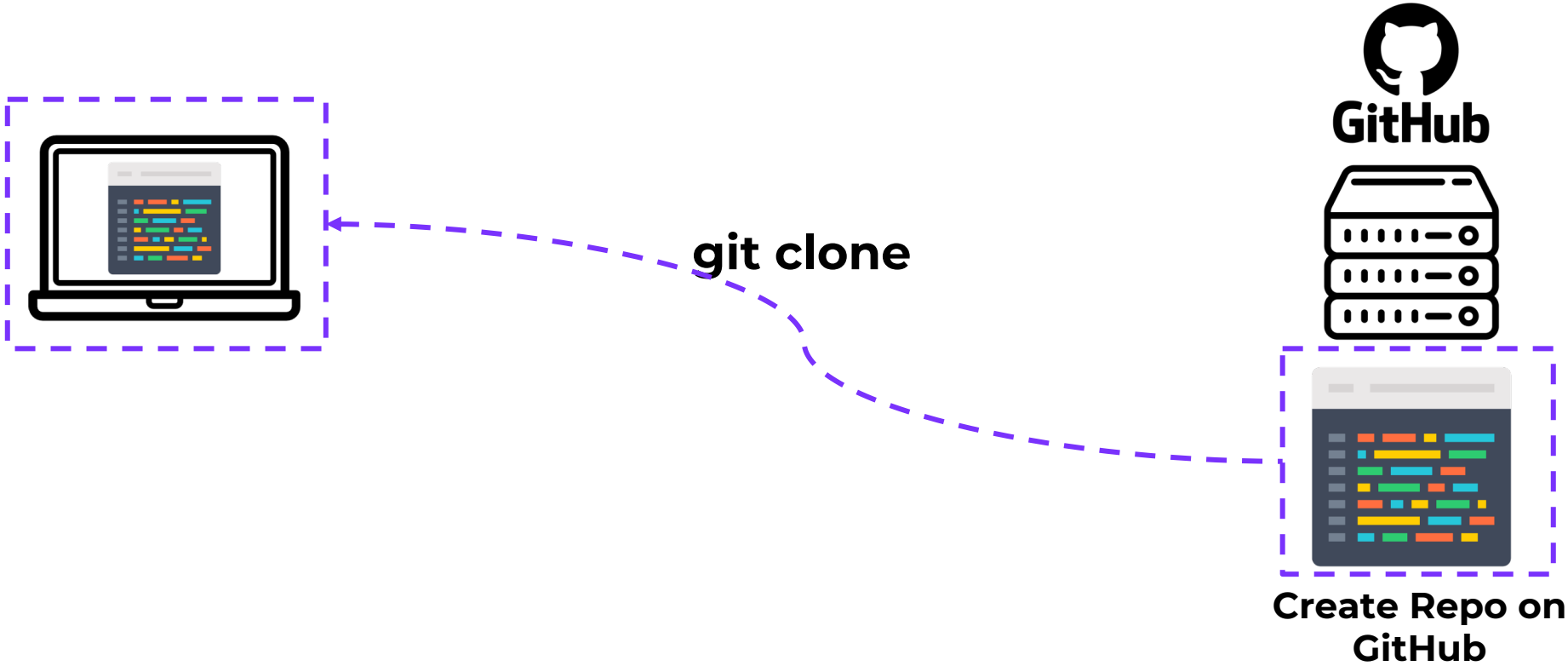


Week 2 Getting Started with Git

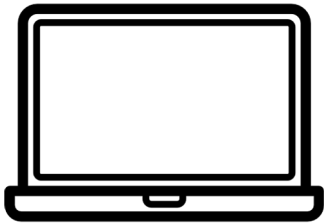




Week 2 Getting Started with Git



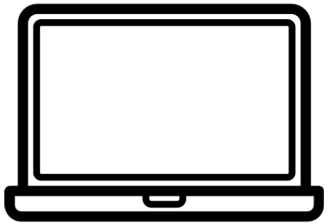
Week 2 Getting Started with Git



Week 2 Getting Started with Git

Week 2 Getting Started with Git

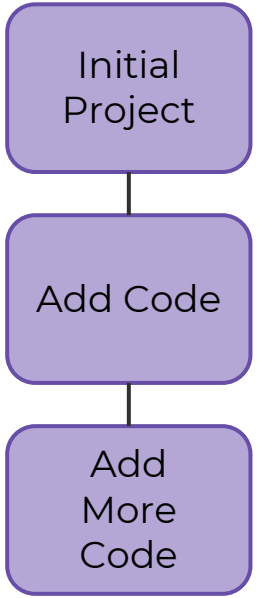
Week 2 Getting Started with Git



What we need to learn toWeek:

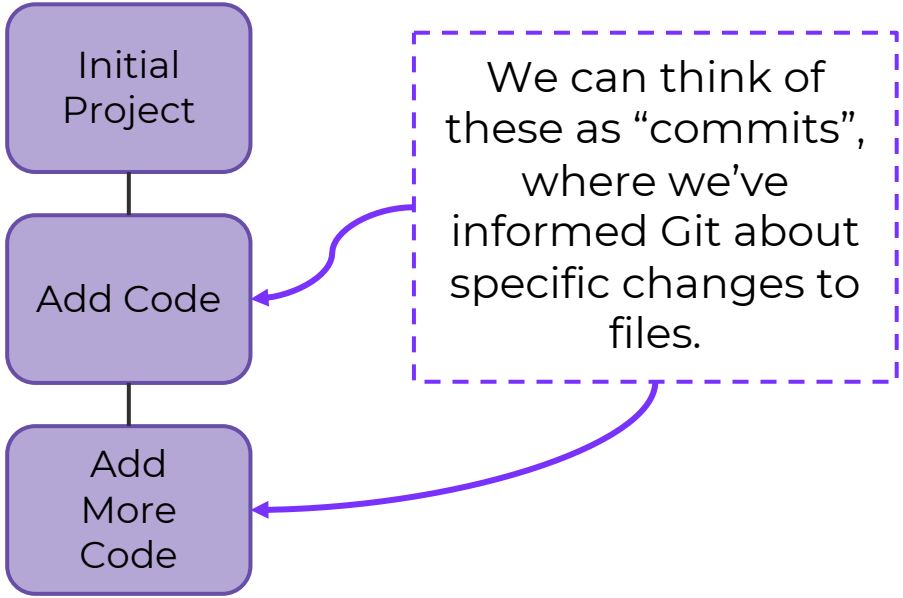
- Git Workflow
- How to tell Git about changes to our code
- How to push changes to GitHub
- How to pull changes from GitHub

Week 2 Getting Started with Git



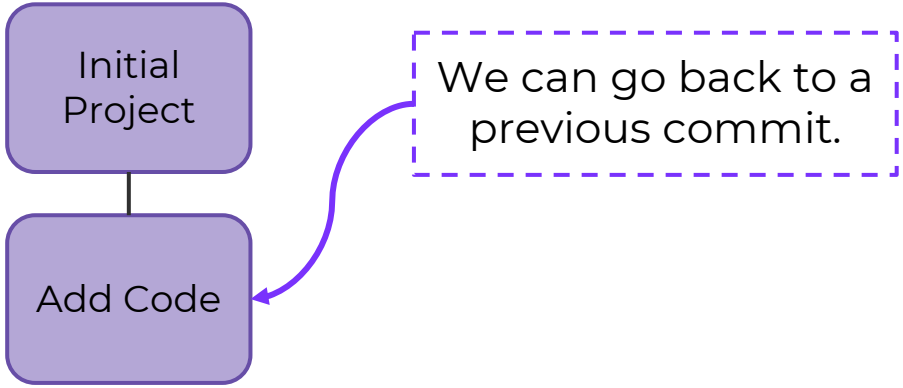

```
Add Code

More Code
```

Week 2 Getting Started with Git



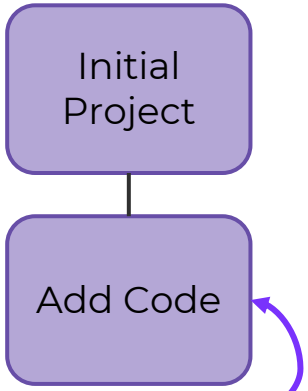

```
Add  Code

More  Code
```

Week 2 Getting Started with Git

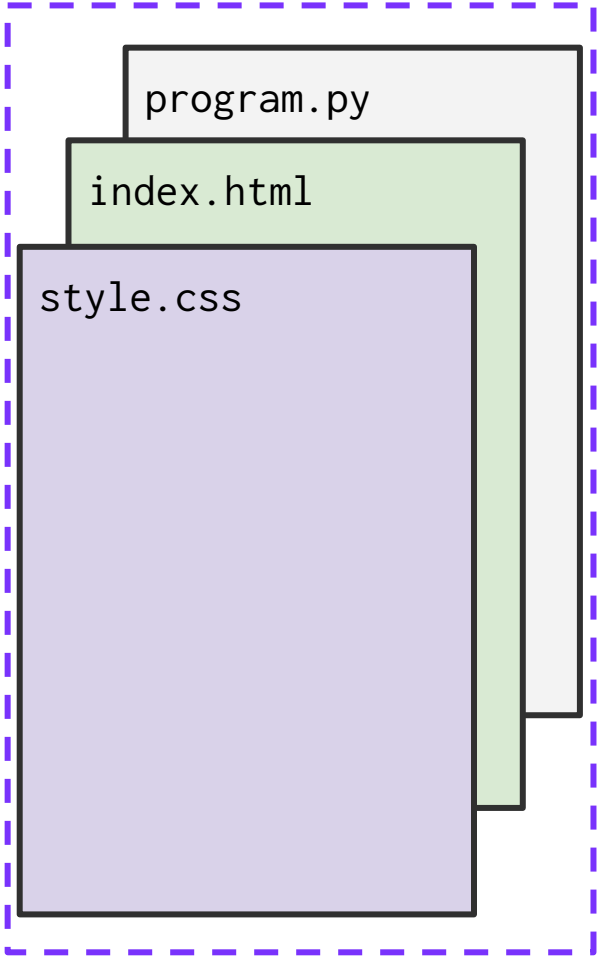


Add Code

Week 2 Getting Started with Git



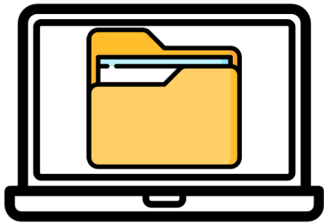
A Git commit doesn't just pertain to a saving changes in a single file. It can constitute specific changes across an entire **working directory**.



Week 2 Getting Started with Git

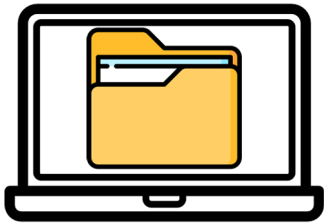
Week 2 Getting Started with Git

Working Directory



Week 2 Getting Started with Git

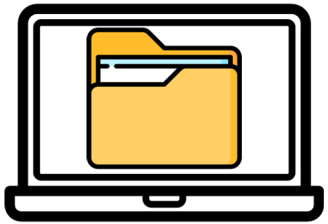
Working Directory



`program.py`

Week 2 Getting Started with Git

Working Directory



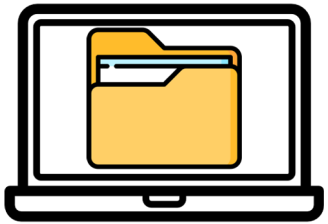
program.py

index.html

style.css

Week 2 Getting Started with Git

Working Directory



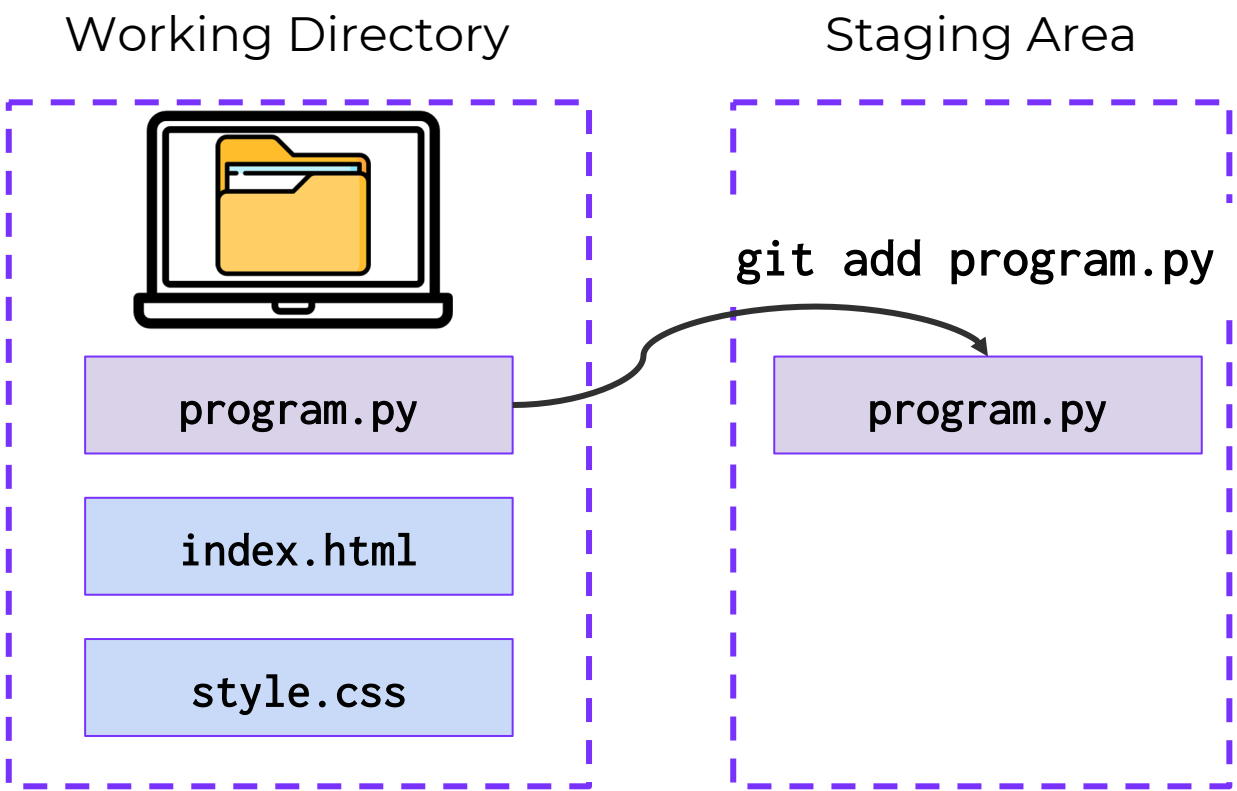
program.py

index.html

style.css

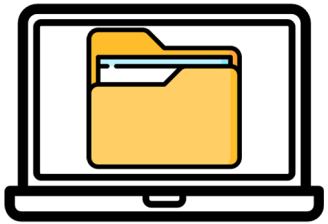
Staging Area

Week 2 Getting Started with Git



Week 2 Getting Started with Git

Working Directory



program.py

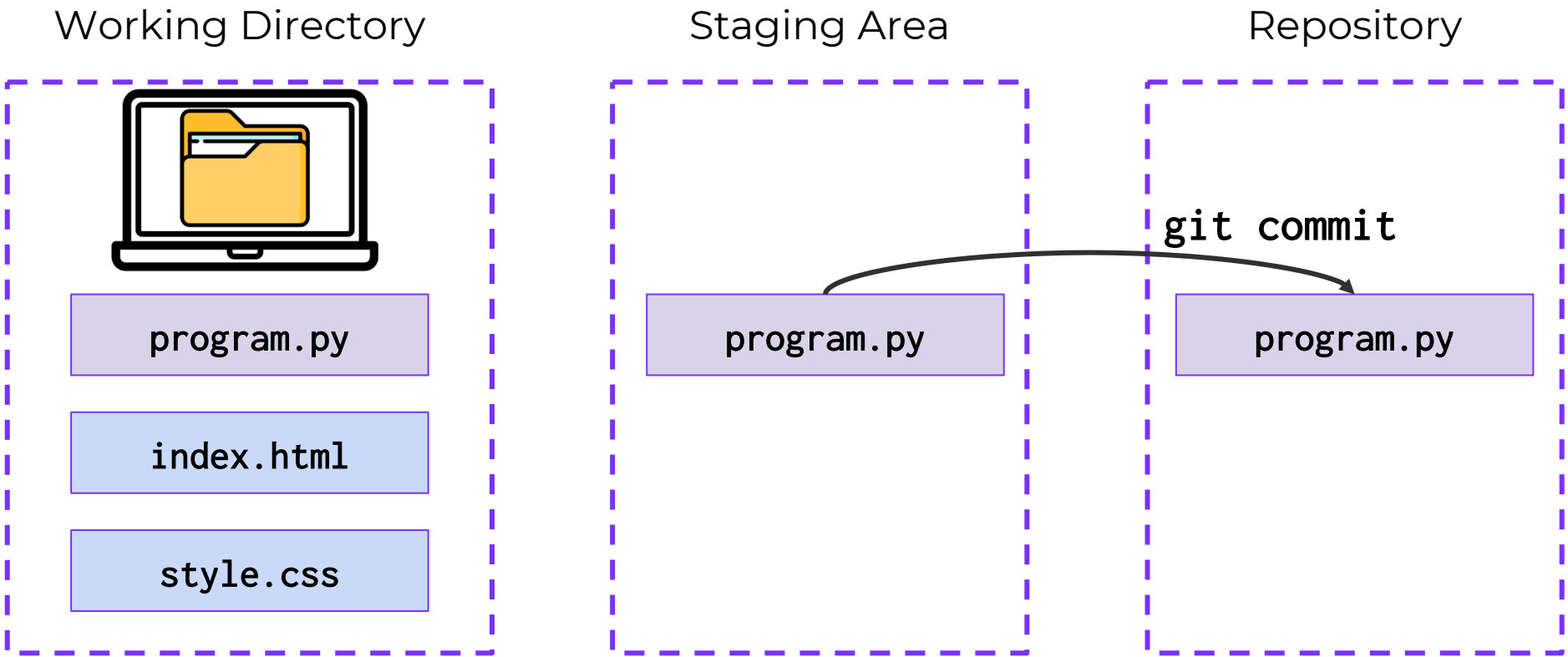
index.html

style.css

Staging Area

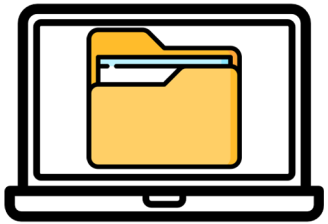
program.py

Week 2 Getting Started with Git



Week 2 Getting Started with Git

Working Directory



program.py

index.html

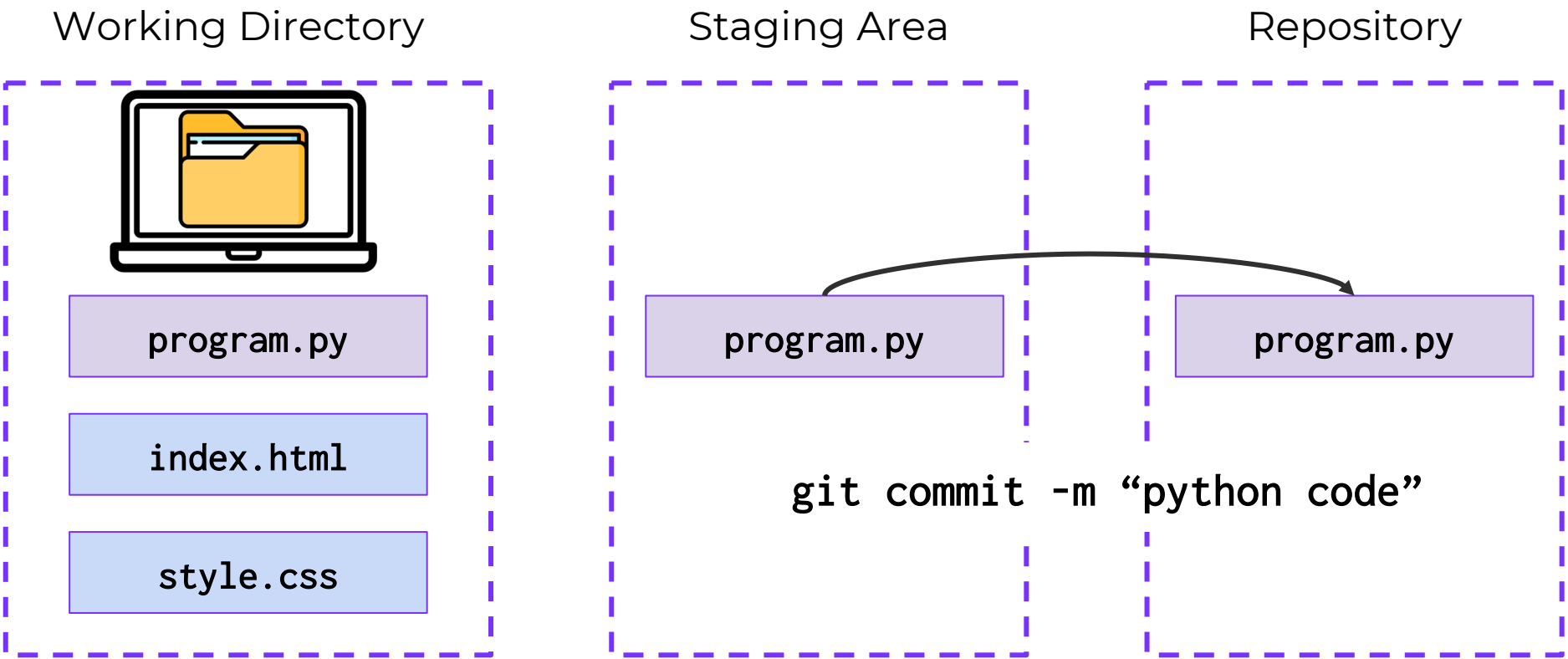
style.css

Staging Area

Repository

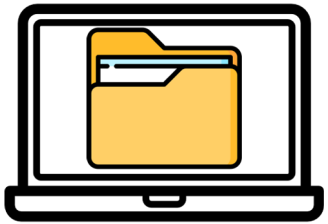
program.py

Week 2 Getting Started with Git



Week 2 Getting Started with Git

Working Directory



program.py

index.html

style.css

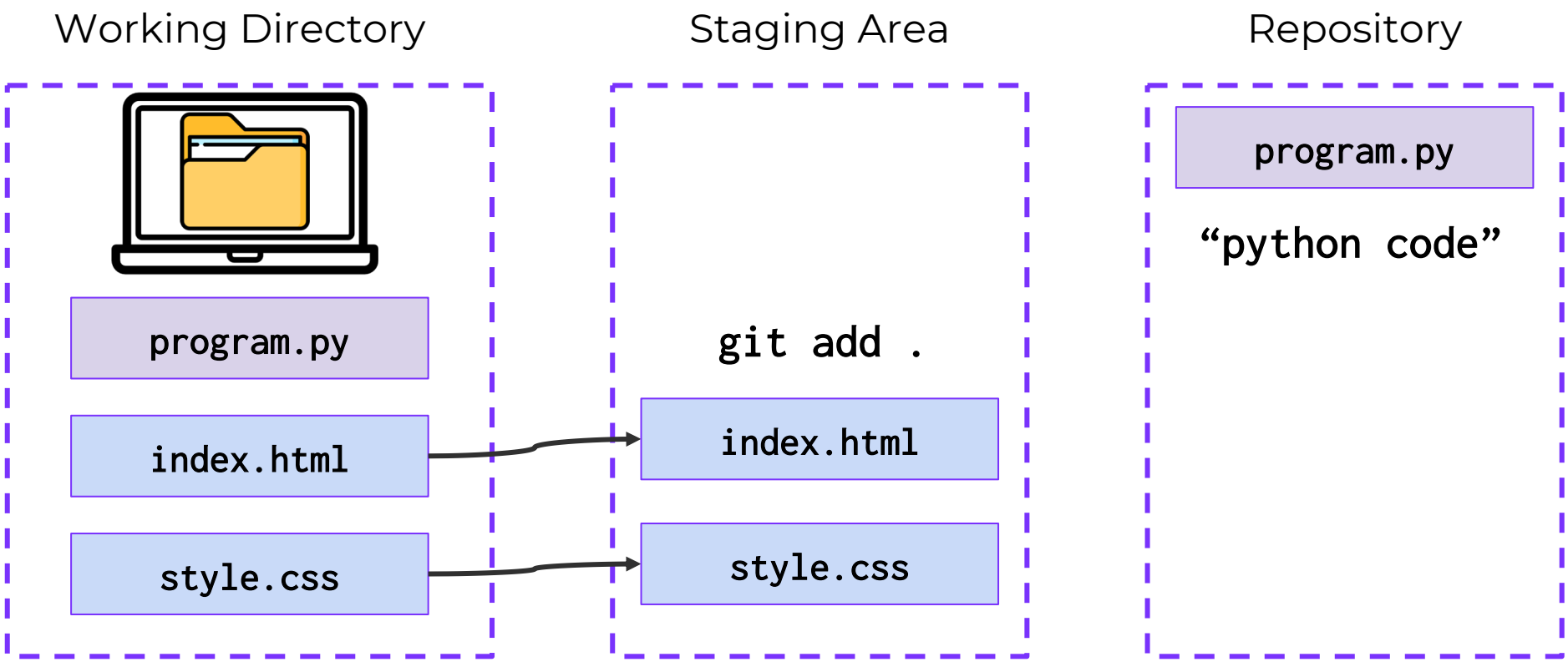
Staging Area

Repository

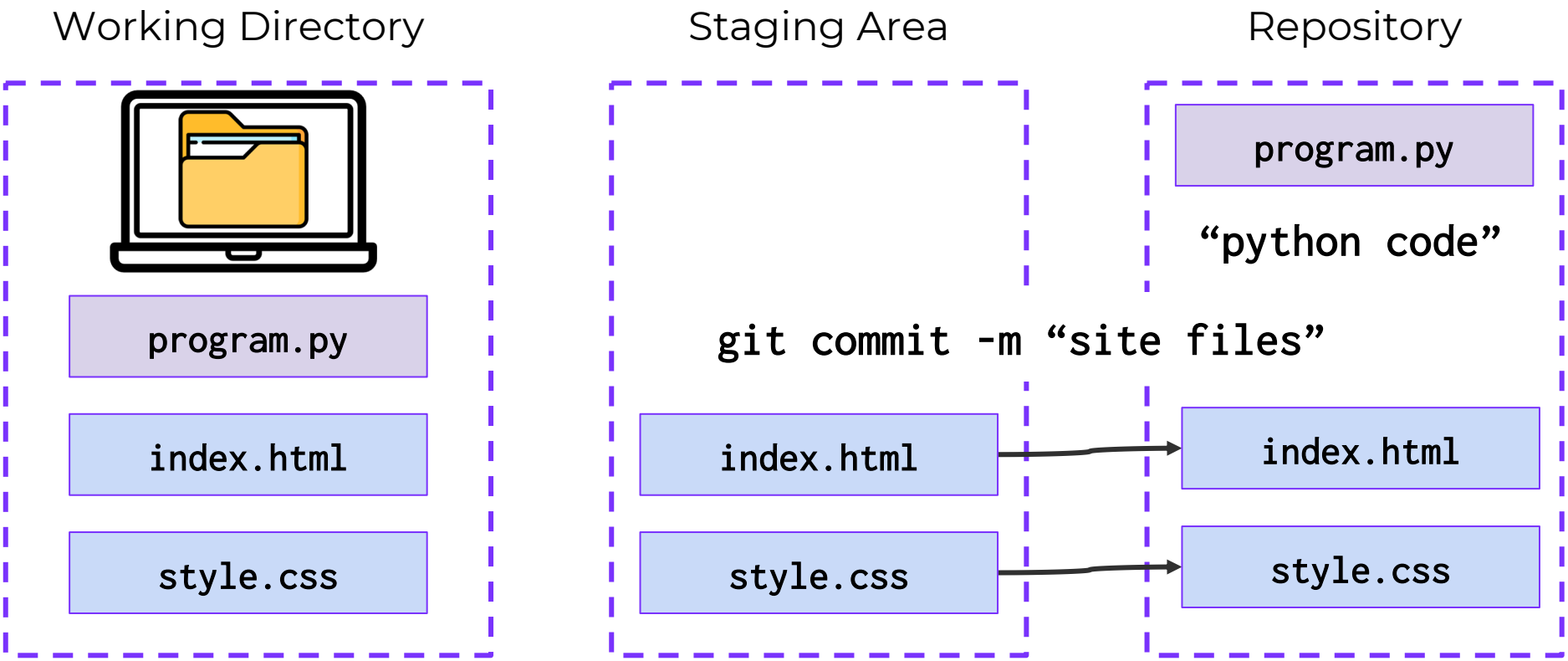
program.py

“python code”

Week 2 Getting Started with Git

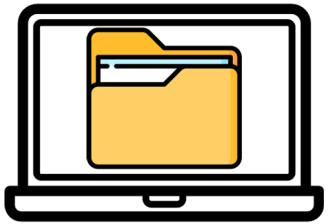


Week 2 Getting Started with Git



Week 2 Getting Started with Git

Working Directory



program.py

index.html

style.css

Staging Area

Repository

program.py

"python code"

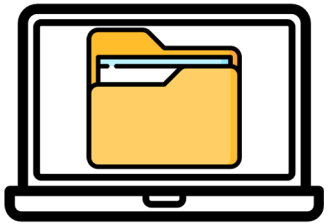
index.html

style.css

"site files"

Week 2 Getting Started with Git

Working Directory



program.py

index.html

style.css

Repository

program.py

“python code”

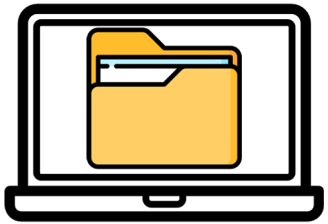
index.html

style.css

“site files”

Week 2 Getting Started with Git

Working Directory



program.py

index.html

style.css

Repository

program.py

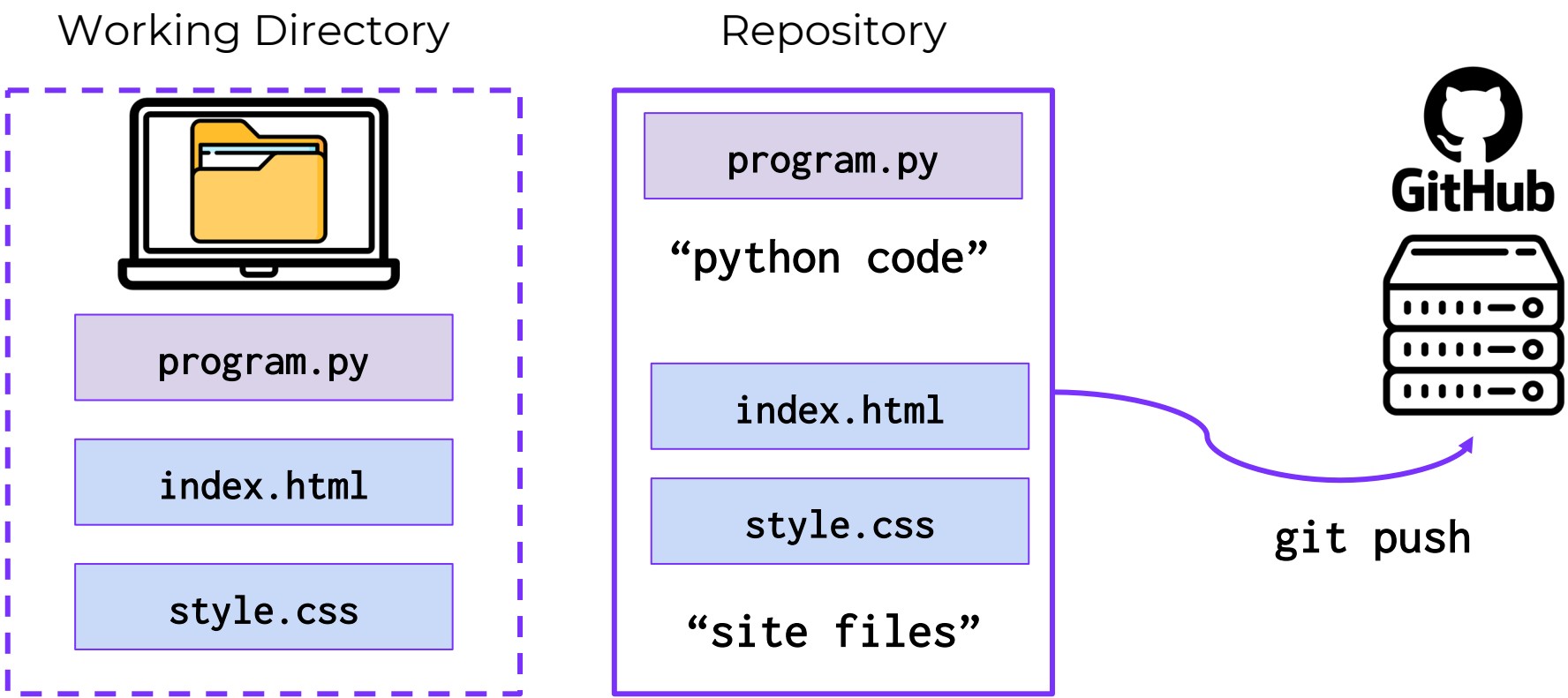
"python code"

index.html

style.css

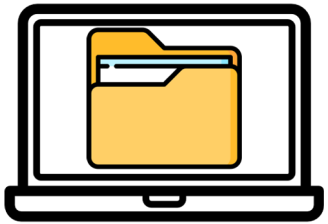
"site files"

Week 2 Getting Started with Git



Week 2 Getting Started with Git

Working Directory



program.py

Repository

program.py

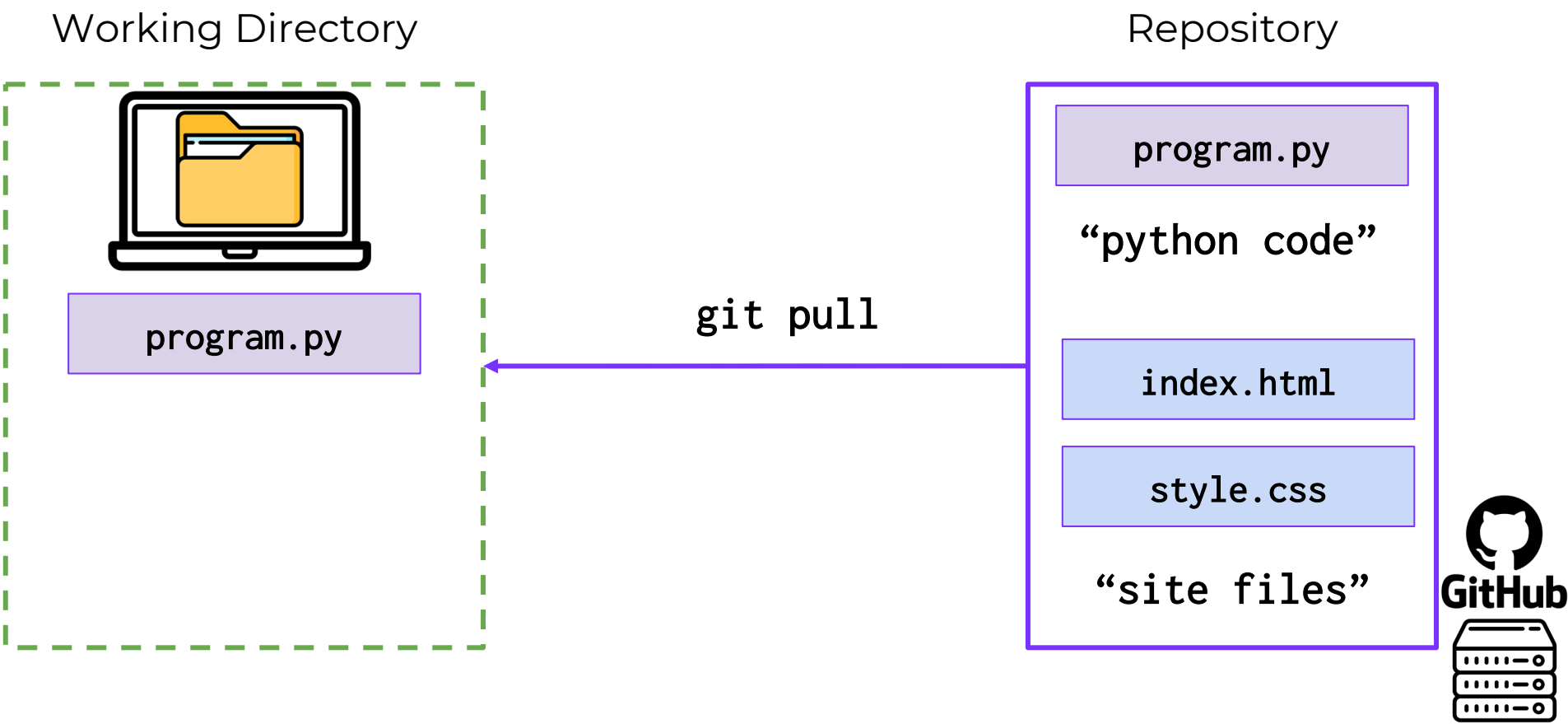
“python code”

index.html

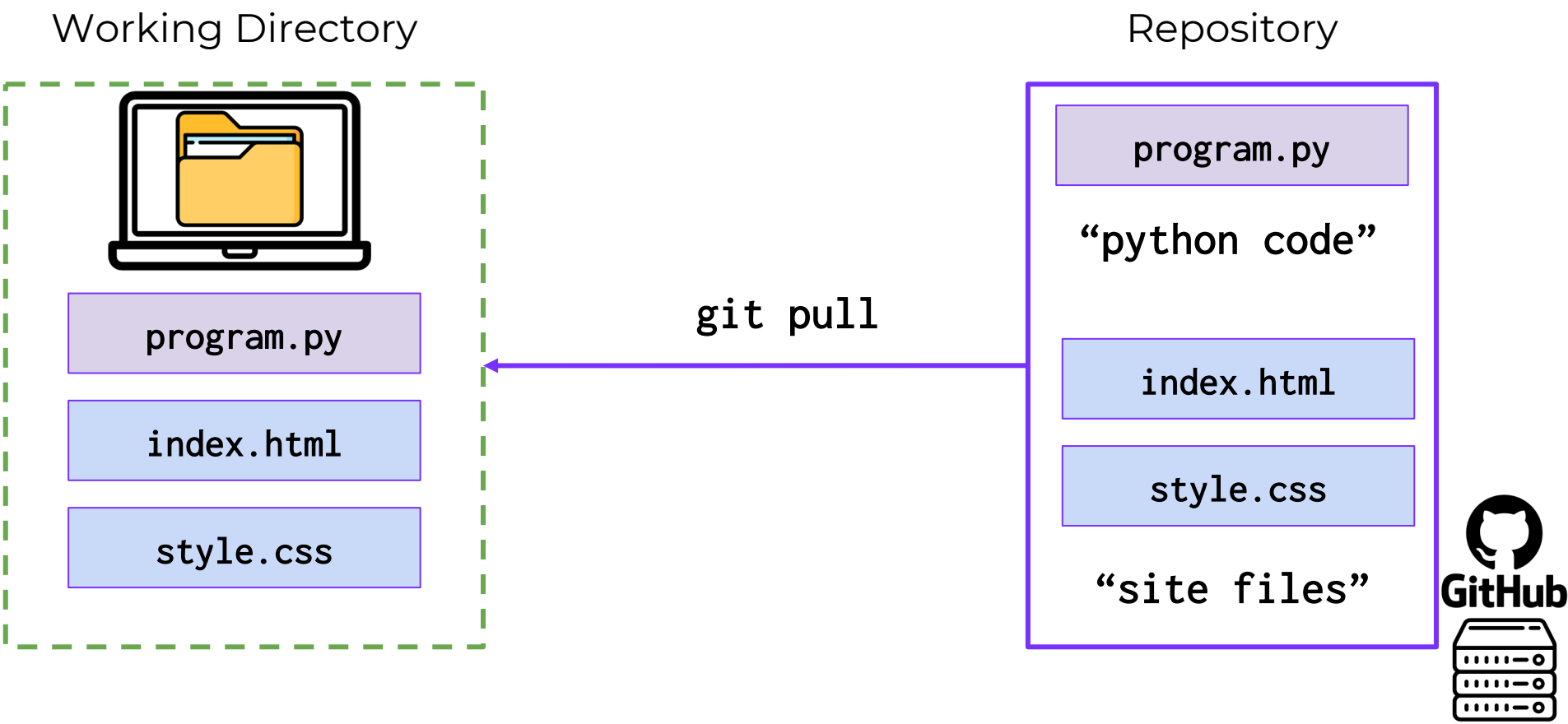
style.css

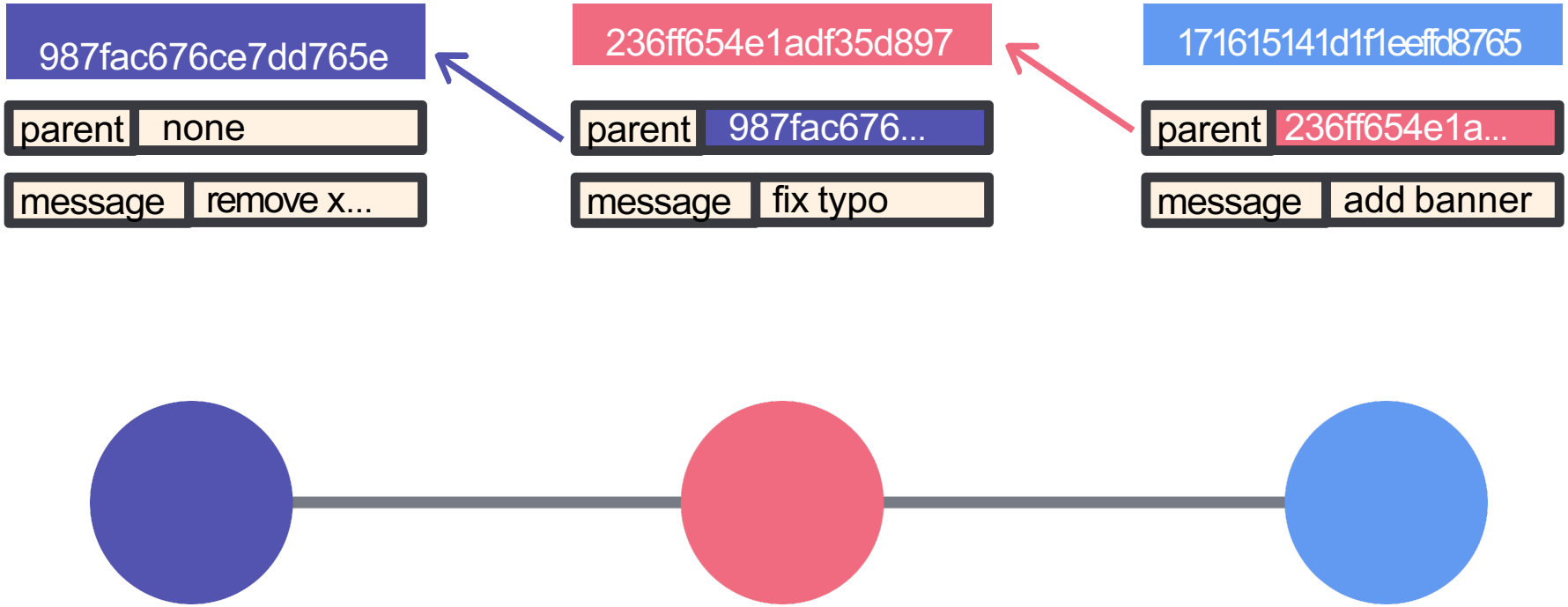
“site files”

Week 2 Getting Started with Git



Week 2 Getting Started with Git




987fac676ce7dd765e
parent
none
message
remove x...
236ff654e1adf35d897
parent
987fac676...
message
fix typo
171615141d1f1eeffd8765
parent
236ff654e1a...
message
add banner

Week 2 Getting Started with Git

- This is the main workflow we're covering toWeek.
- However as we continue, try to keep in mind what would happen if we had multiple people working on different parts of the code at different times.
  - *How would we deal with differences or conflicting code? What about branches?*

Week 2
Add and Commit

Week 2 Getting Started with Git

- Let's now go through the process of creating a repository, creating new files inside the working directory, adding them to the staging area, and then committing them to the repository.
- You will need a text editor, we recommend VS Code, download it here:
  - **https://code.visualstudio.com/**

Week 2
Push and
Remote Branches

Week 2 Getting Started with Git

- We've learned how to create repositories locally and add changes to the staging area and then commit them to the main (master) branch we have locally.
- If you are also using GitHub as a hosting service, we can think of this as a **remote branch**.

Week 2 Getting Started with Git

- We're still operating under the assumption that the context is just a single solo developer operating on just a single branch, but later on we'll talk about branches in more detail.
- Let's learn how to **push** local code to a **remote branch** on GitHub.

Week 2 Getting Started with Git

- We can check for remote branches with the command:
  - **git remote -v**
- If you run this command on a cloned repo, you will **view** the URL of the remote branch, like the GitHub URL.
- If there is no connection to a remote branch, then you won't see a URL.

Week 2 Getting Started with Git

- After we've created a repository locally, we need to create the repository on GitHub.
- Once you've created the repository on GitHub, you will actually see the instructions under: ***"...or push an existing repository from the command line"***

Week 2 Getting Started with Git

- We tell git we want to add a remote branch using the git remote command syntax:
  - **git remote add name https://url.git**
- By convention, we call this remote branch the **origin** branch.
  - **git remote add origin https://url.git**
- You then replace the .git URL with the .git URL from the repository you created.

Week 2 Getting Started with Git

- **Important Note:**
  - *When watching along with our video, you'll need to create your own repository, you won't be able to just push to the repository shown in the video (which makes sense that not just anybody could push to any repo with just the URL). Also you will need a PAT.*

Week 2 Getting Started with Git

- **Important Note:**
  - We won’t use these commands in the video, but just in case you need them in the future:
    - **git remote rename <old> <new>**
    - **git remote remove <name>**

Week 2 Getting Started with Git

- Once we've connected to our remote branch on GitHub, we can **push** our code to the remote branch.
- We tell git to push to the remote main/master branch called origin with the command:
  - **git push -u origin main/master**

Week 2 Getting Started with Git

- **Important Note:**
  - GitHub has officially changed the naming convention of its **master** branch to **main** branch.
  - You'll see this reflected in the instructions that GitHub provides:
    - **git branch -M main**

Week 2 Getting Started with Git

- Let’s explore all of this:
  - Create a new repo on GitHub
  - Connect our repo from the previous lecture to this remote branch on GitHub
  - Push our commits to GitHub

Master or Main??!!

Master?
Main?

In 2020, Github renamed the default branch from **master** to **main.** The default Git branch name is still **master**, though the Git team is exploring a potential change.

We will circle back to this shortly.

Couple Years Back..

Branch: master

LOCAL DESKTOP

Branch: master

GitHub

Improper Reference

Branch: master

LOCAL DESKTOP

Branch: main

GitHub

Clone

Branch: main

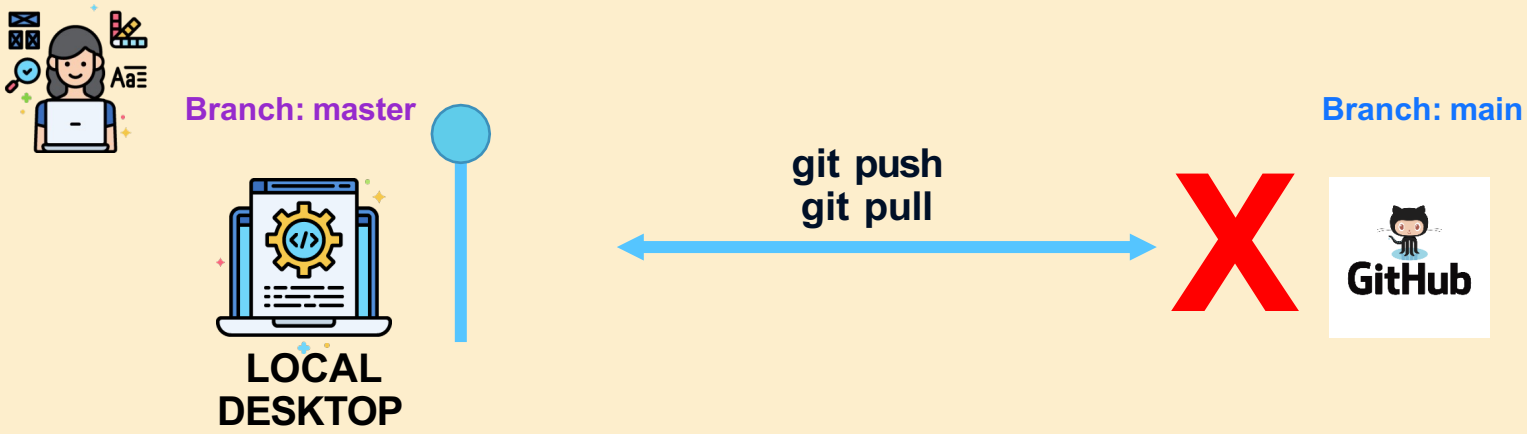
LOCAL DESKTOP

git clone

Branch: main

GitHub

Local Folder to GitHub without Clone



Local Folder to GitHub without Clone

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

Set up in Desktop or HTTPS SSH https://github.com/saha-rajdeep/test77.git

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

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

```
echo "# test77" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/saha-rajdeep/test77.git
git push -u origin main
```

Local Folder to GitHub without Clone

Branch: main

LOCAL DESKTOP

git branch –M main

git push

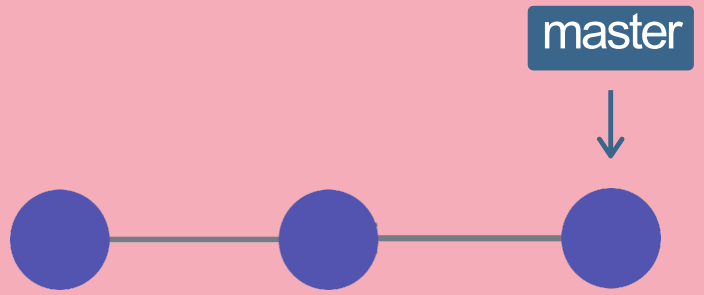
Branch: main

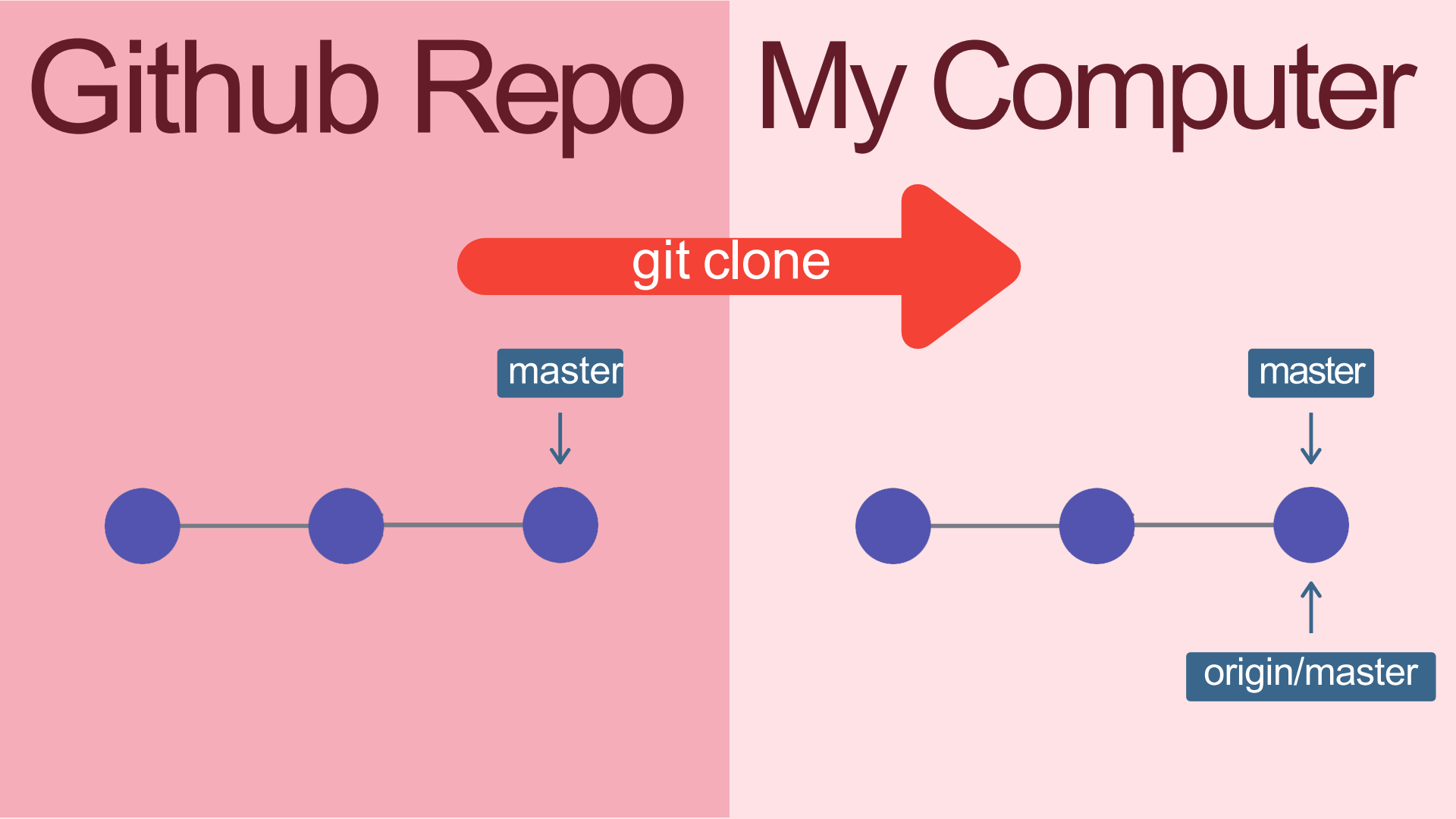
GitHub

A Closer Look At Cloning

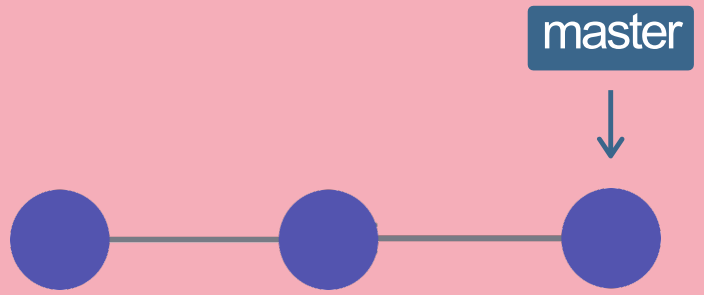
Github Repo

My Computer



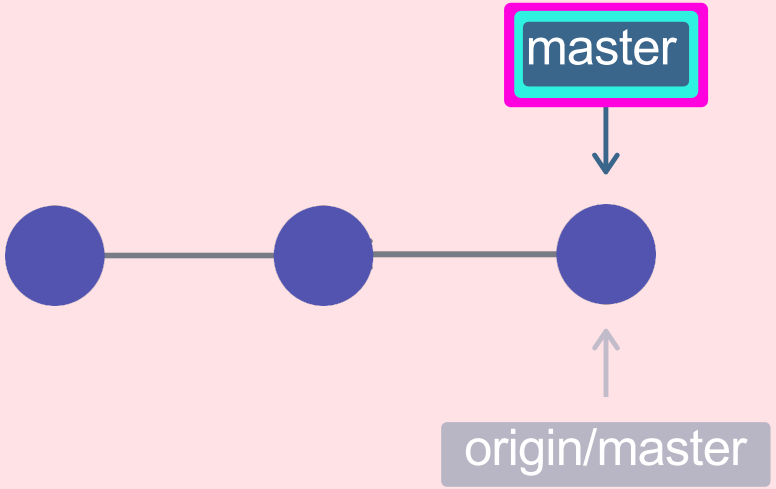

Github Repo
My Computer
git clone
master
master
origin/master

Github Repo



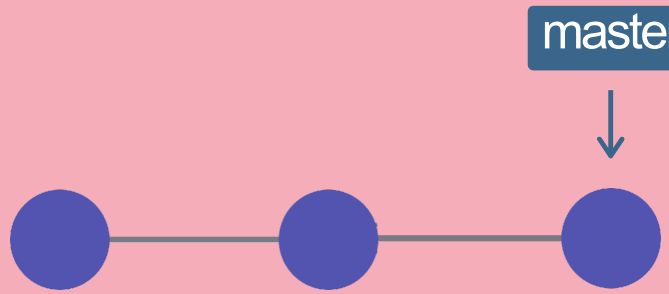
My Computer

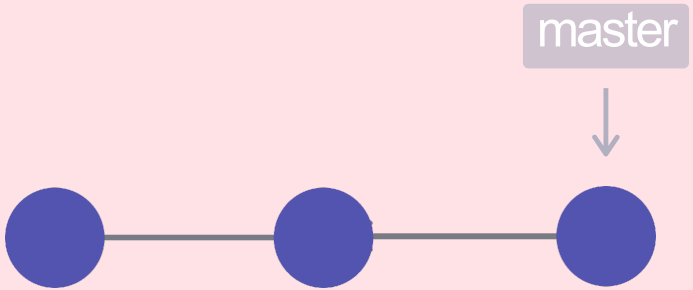
A regular branch reference.
I can move this around myself.



Github Repo

My Computer





This is a "Remote Tracking Branch". It's a reference to the state of the master branch on the remote. I can't move this myself. It's like a bookmark pointing to the last known commit on the master branch on origin

origin/master

Remote Tracking Branches

"At the time you last communicated with this remote repository, here is where x branch was pointing"

They follow this pattern <remote>/<branch>.

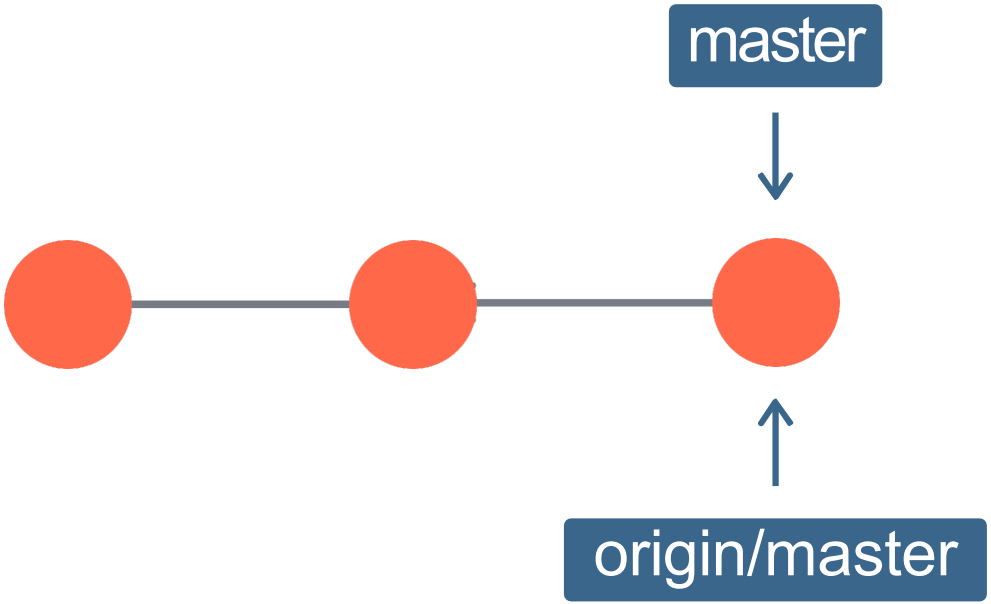
- origin/master references the state of the master branch on the remote repo named origin.
- upstream/logoRedesign references the state of the logoRedesign branch on the remote named upstream (a common remote name)

Remote Branches

Run git branch -r to view the remote branches our local repository knows about.

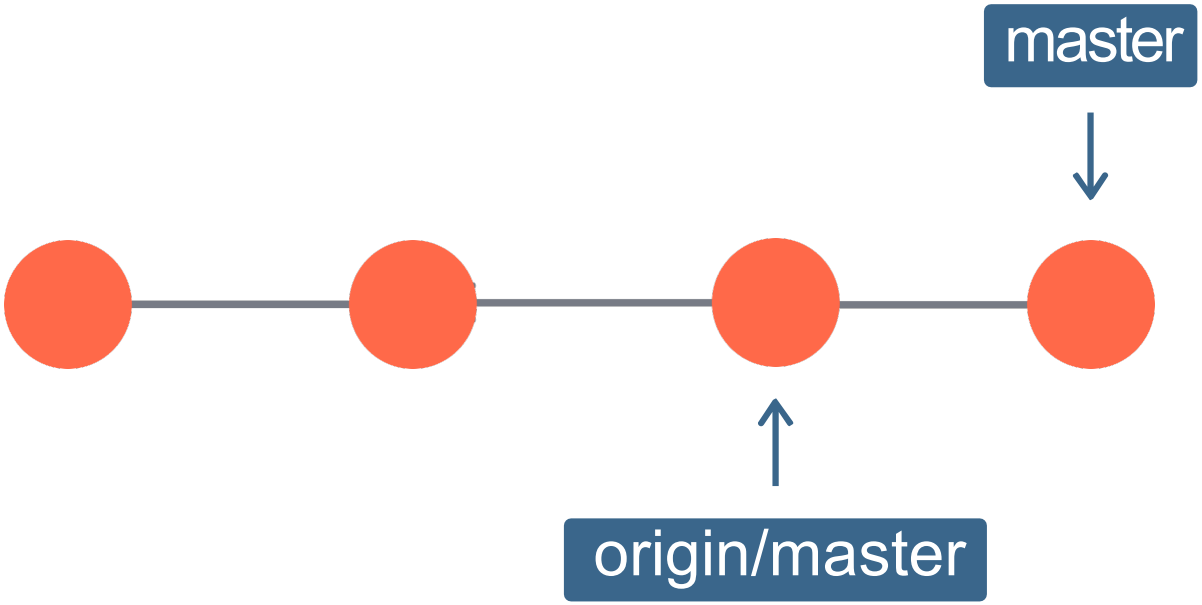
```
> git branch -r
  origin/master
```

My Computer



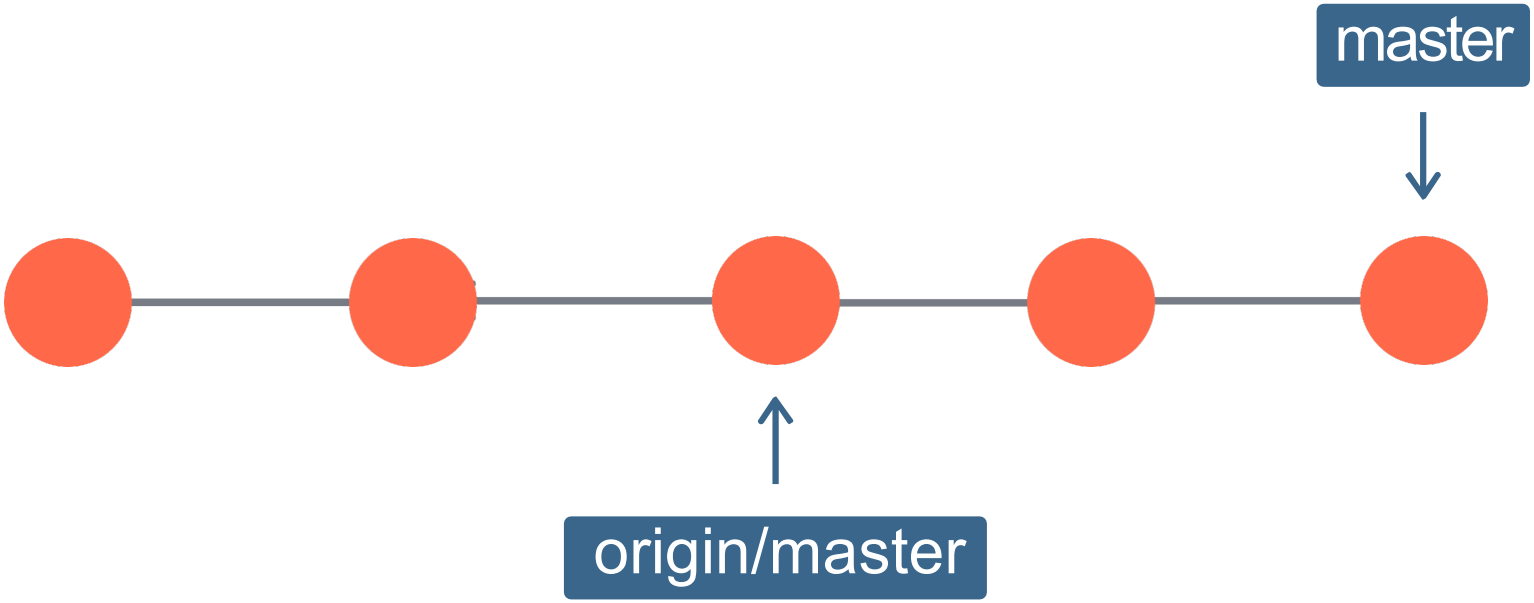
My Computer

I make a new commit locally. My master branch reference updates, like always.



The remote reference stays the same

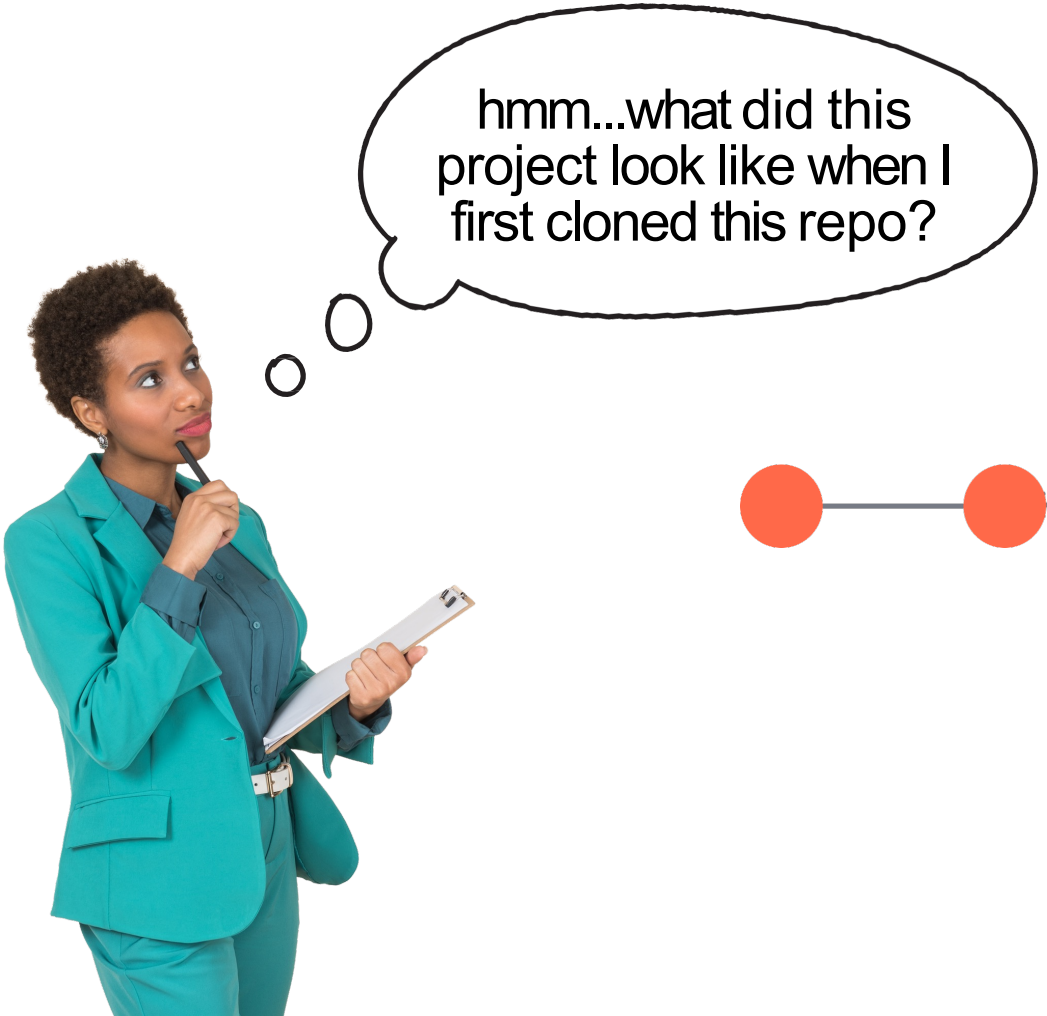
My Computer

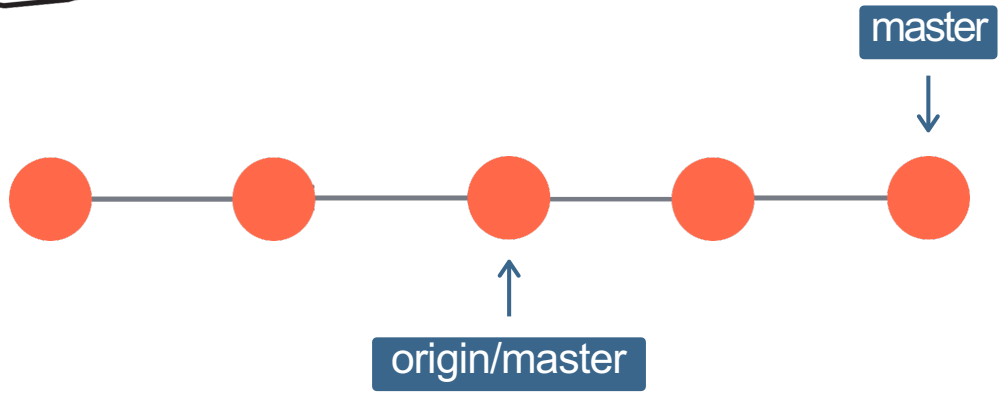


Remote reference doesn't move!

When I run git status

```
❯ git status
  On branch master
  Your branch is ahead of 'origin/master' by 2
  commits.
    (use "git push" to publish your local commits)
```


hmm...what did this project look like when I first cloned this repo?


master
origin/master

You can checkout these remote branch pointers

```
git checkout origin/master

Note: switching to 'origin/master'.
Youarein'detached HEAD'state. You can look
around, make experimental changes and commit
them, and youcan  discard any commits you make
in this blah blah blah blah
```

Detached HEAD! Don't panic. It's fine.

Remote Branches

Once you've cloned a repository, we have all the data and Git history for the project at that moment in time. However, that does not mean it's all in my workspace!
The github repo has a branch called puppies, but when I run git branch I don't see it on my machine! All I see is the master branch. What's going on?

```
❯ git branch
  *master
```

Week 2
Git Log

Week 2 Getting Started with Git

- Before we jump into using git fetch and git pull, let’s quickly show you how to use **git log**.
- The **git log** command will show a list of all the commits made to a repository, including the hash, message, and metadata.
- Think of it as the history of a repo.

Week 2
Fetch and Pull

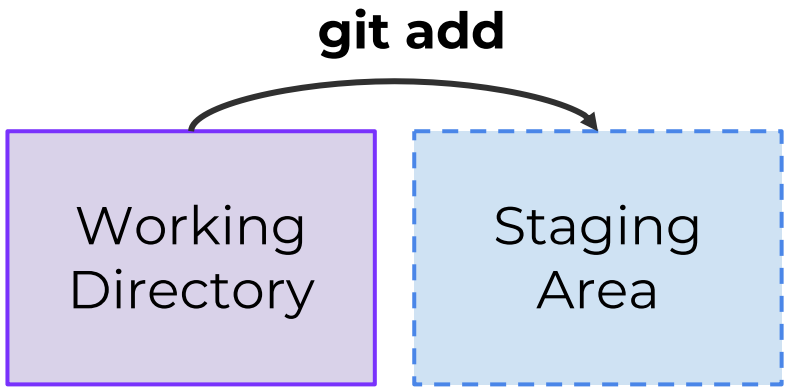
Week 2 Getting Started with Git

- There are two options of getting repository changes from a remote branch (like the remote branch on GitHub).
  - **git pull**
  - **git fetch**

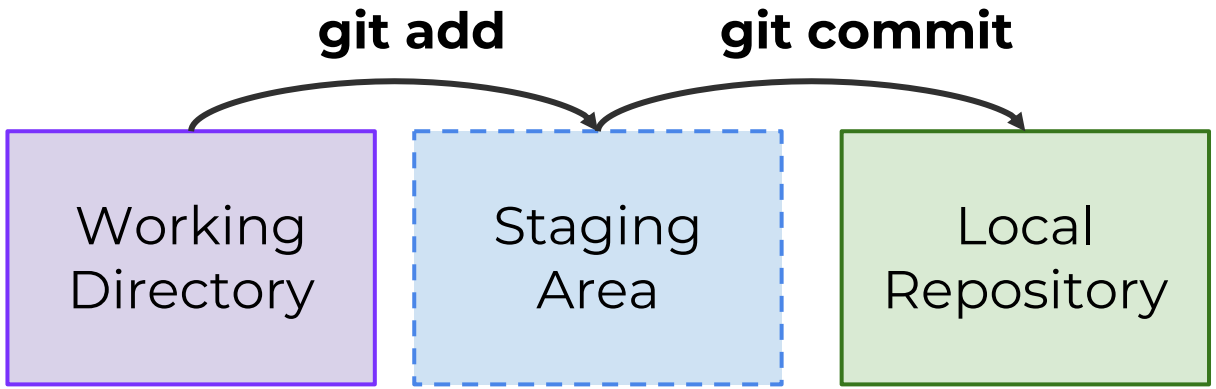
Week 2 Getting Started with Git

Working Directory

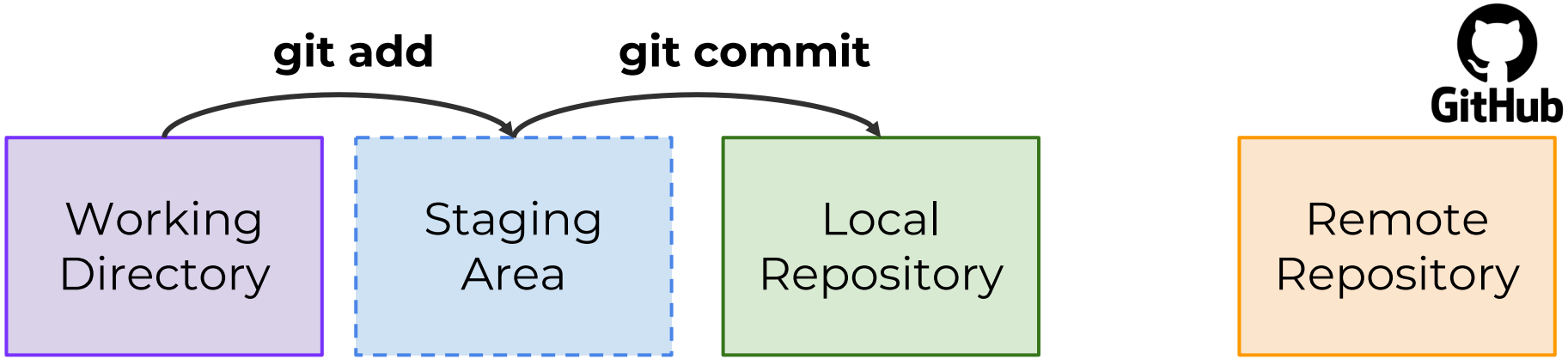
Week 2 Getting Started with Git



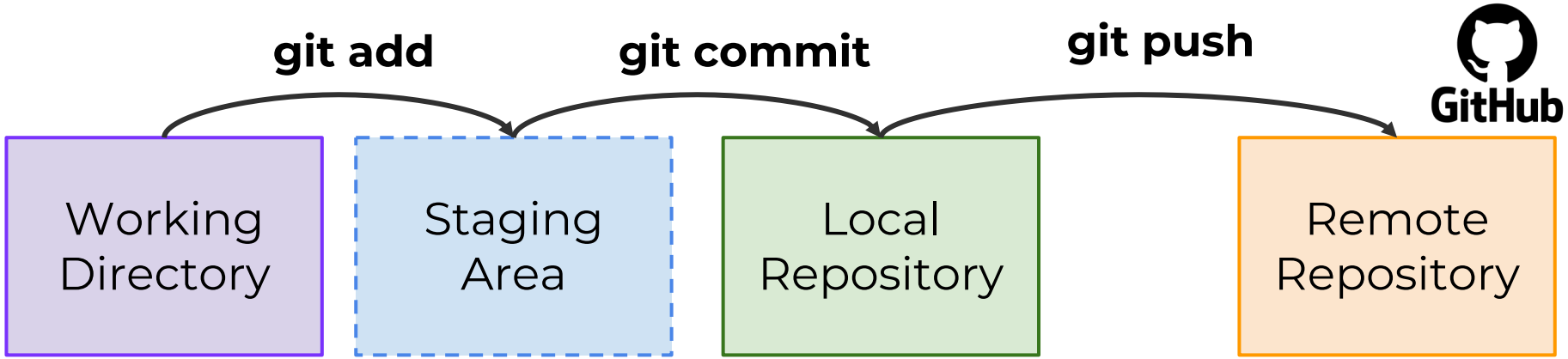
Week 2 Getting Started with Git



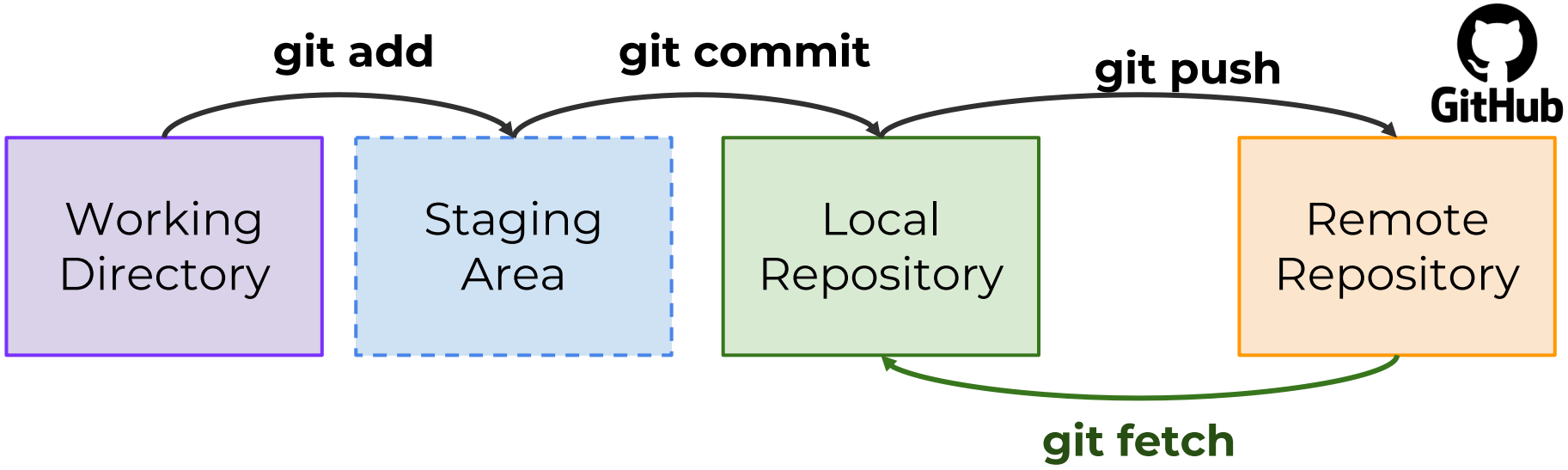
Week 2 Getting Started with Git



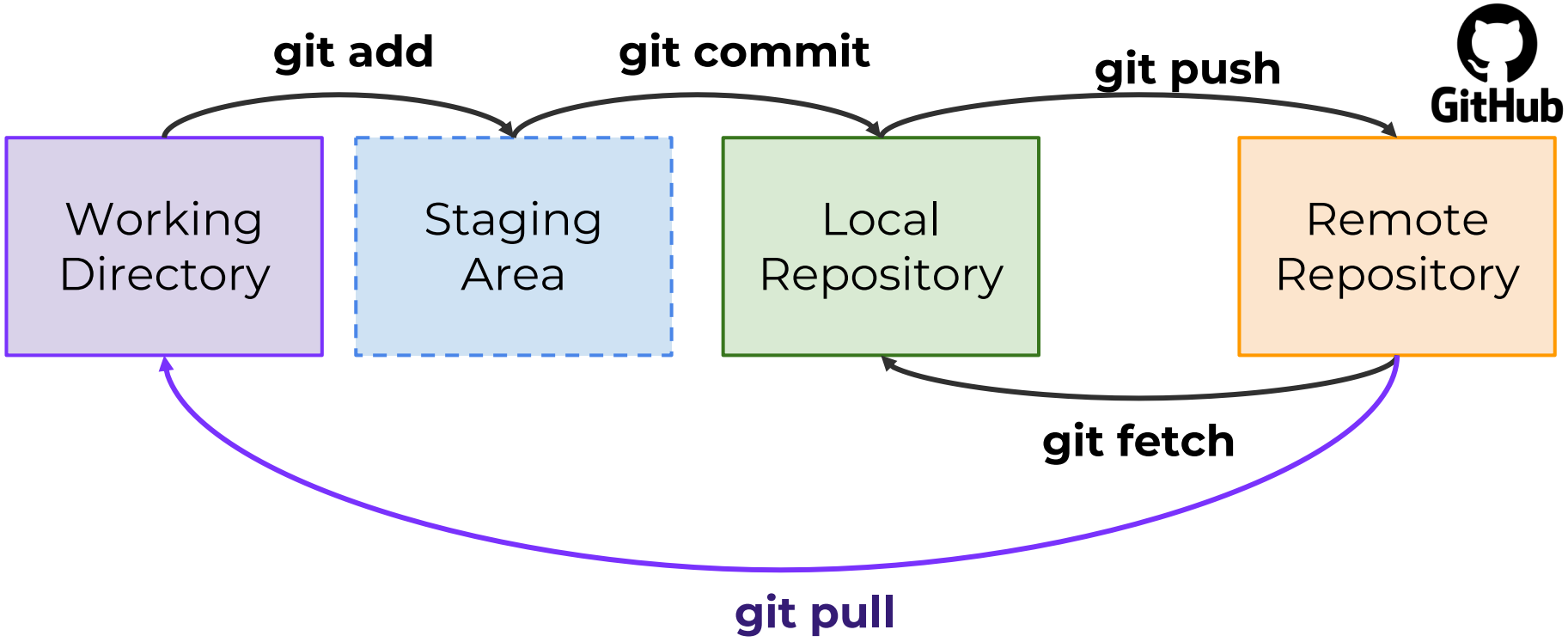
Week 2 Getting Started with Git



Week 2 Getting Started with Git

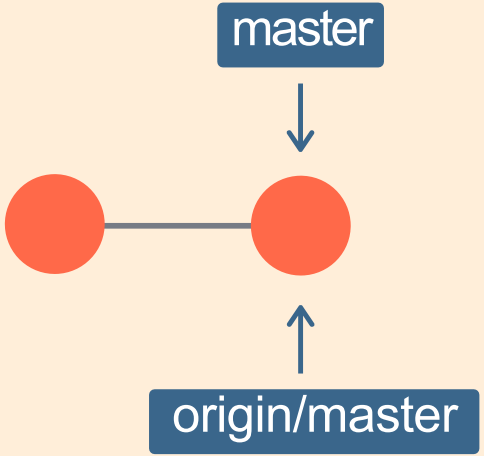


Week 2 Getting Started with Git



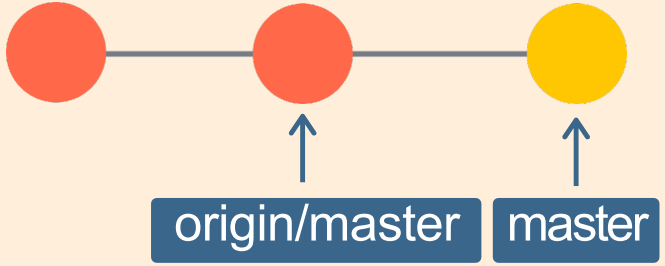
Github

Local



Github

Local

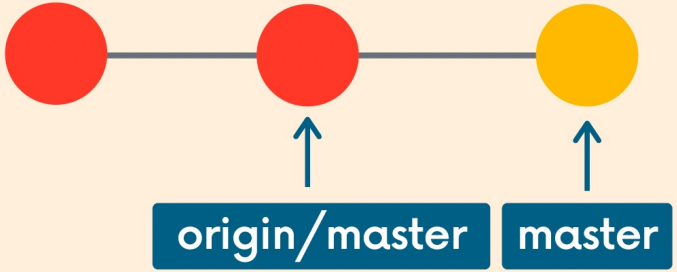


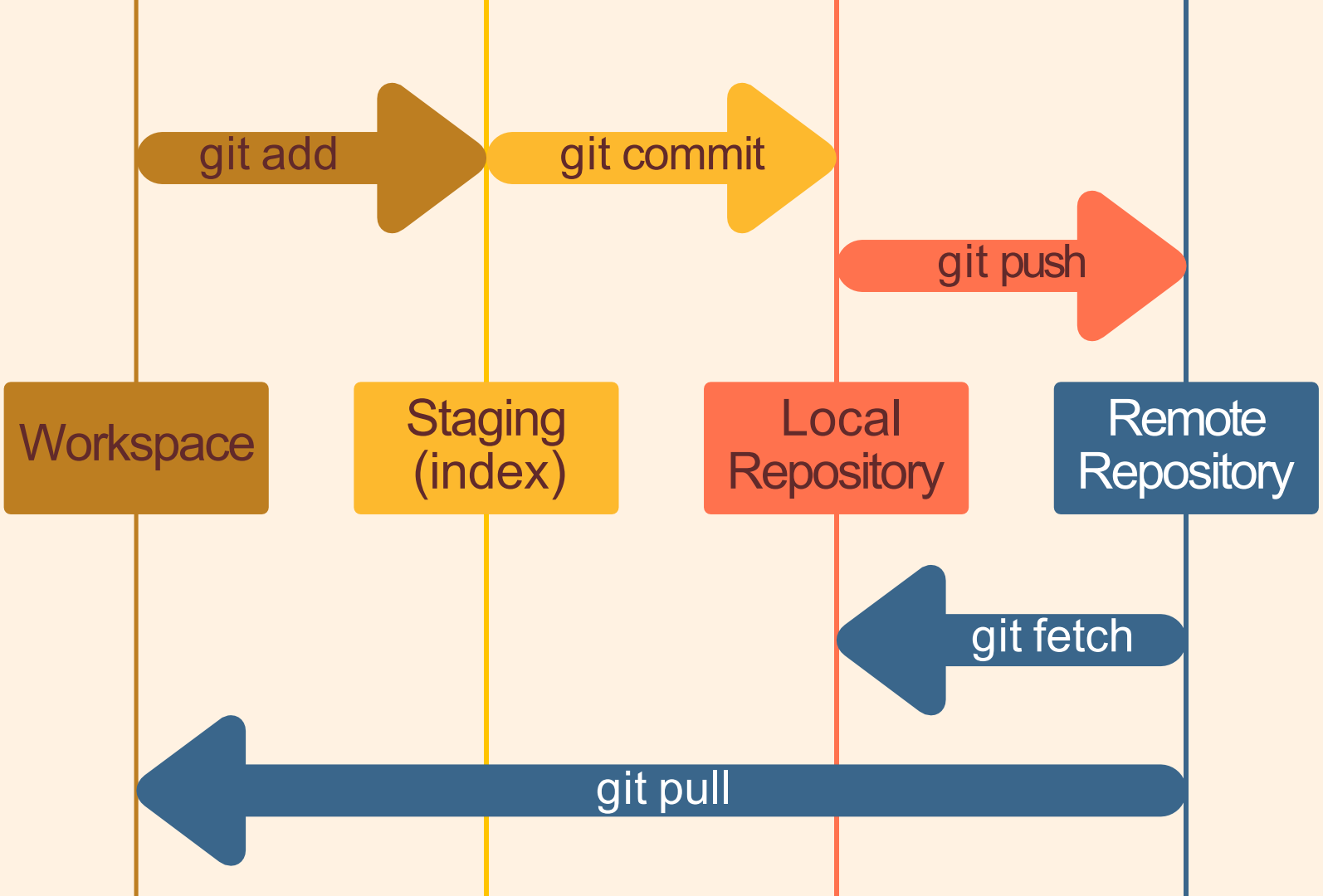
Github

Uh oh! The remote repo has changed! A teammate has pushed up changes to the master branch, but my local repo doesn't know!

How do I get those changes???

Local




git add
git commit
git push
Workspace
Staging
(index)
Local
Repository
Remote
Repository
git fetch
git pull

Fetching

Fetching allows us to download changes from a remote repository, BUT those changes will not be automatically integrated into our working files.

It lets you see what others have been working on, without having to merge those changes into your local repo.

Think of it as "please go and get the latest information from Github, but don't screw up my working directory."

Git Fetch

The git fetch <remote> command fetches branches and history from a specific remote repository. It only updates remote tracking branches.

git fetch origin would fetch all changes from the origin remote repository.

```
git fetch <remote>
```

If not specified, <remote> defaults to origin

Git Fetch

We can also fetch a specific branch from a remote using git fetch <remote> <branch>

For example, git fetch origin master would retrieve the latest information from the master branch on the origin remote repository.

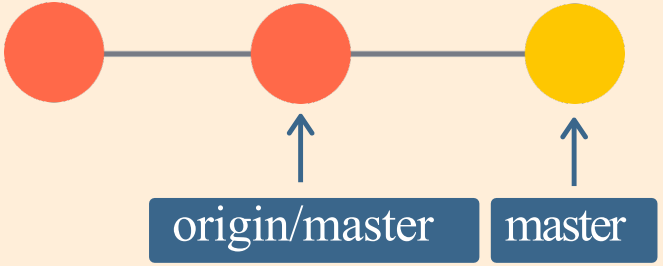
```
git fetch <remote> <branch>
```

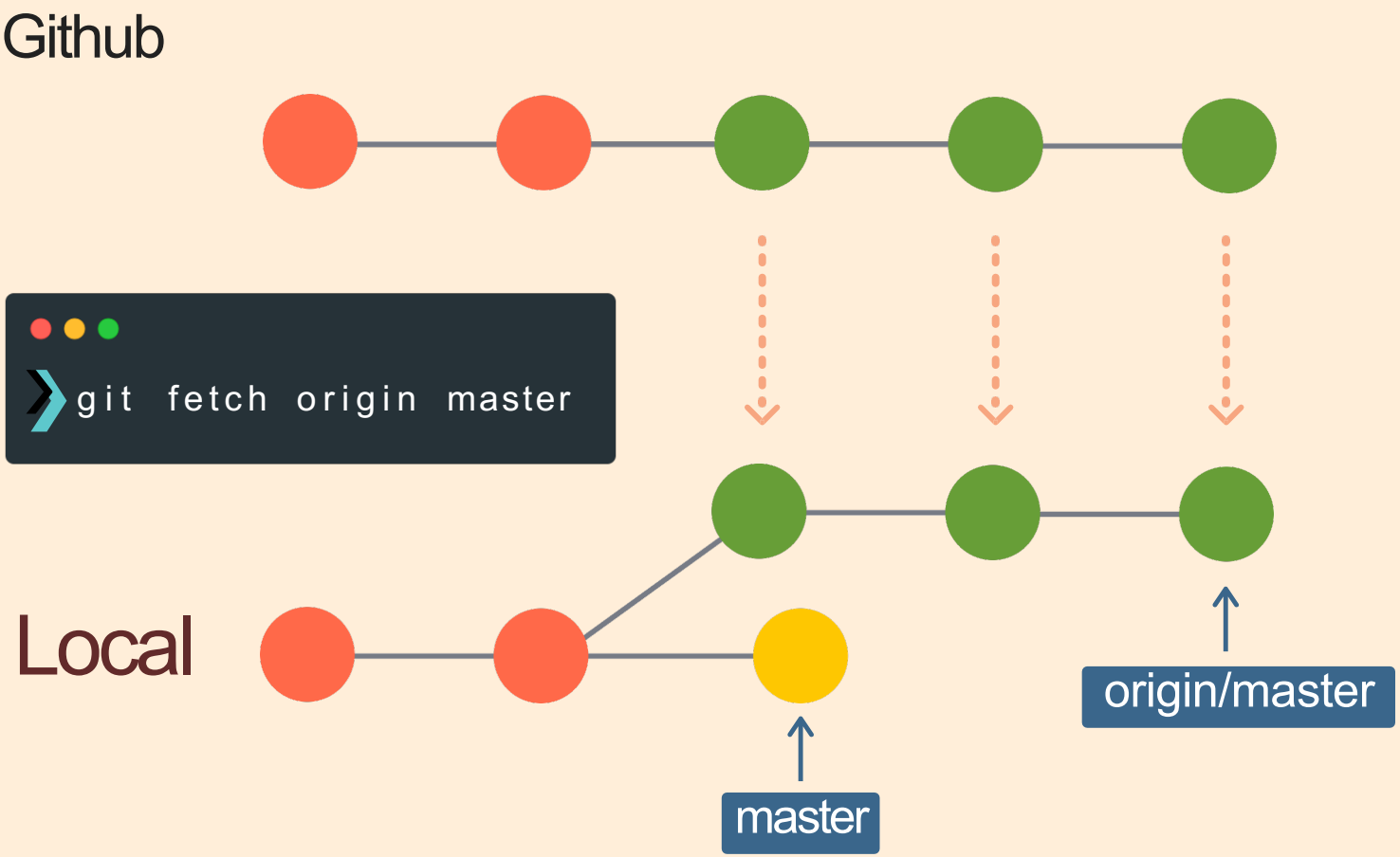
Github

- Uh oh! The remote repo has changed! A teammate has pushed up changes to the master branch, but my local repo doesn't know!

- How do I get those changes???

Local

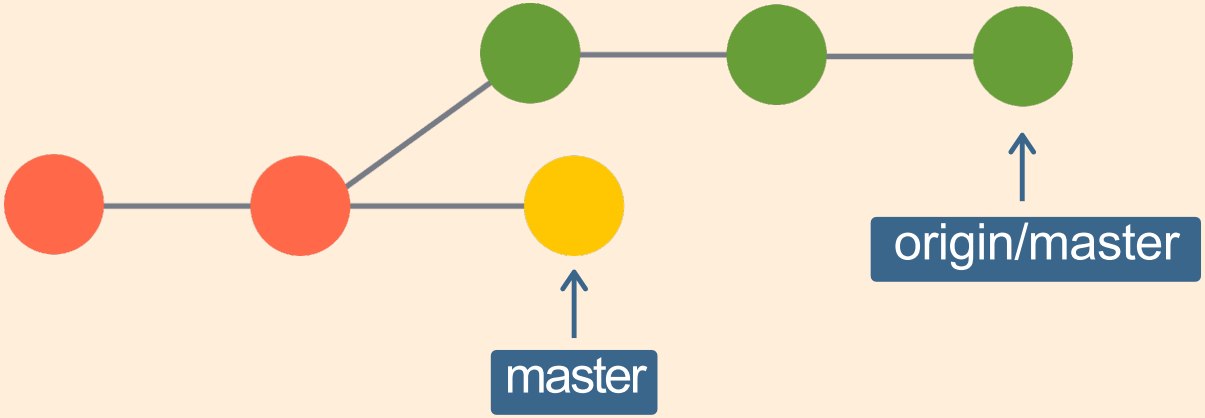


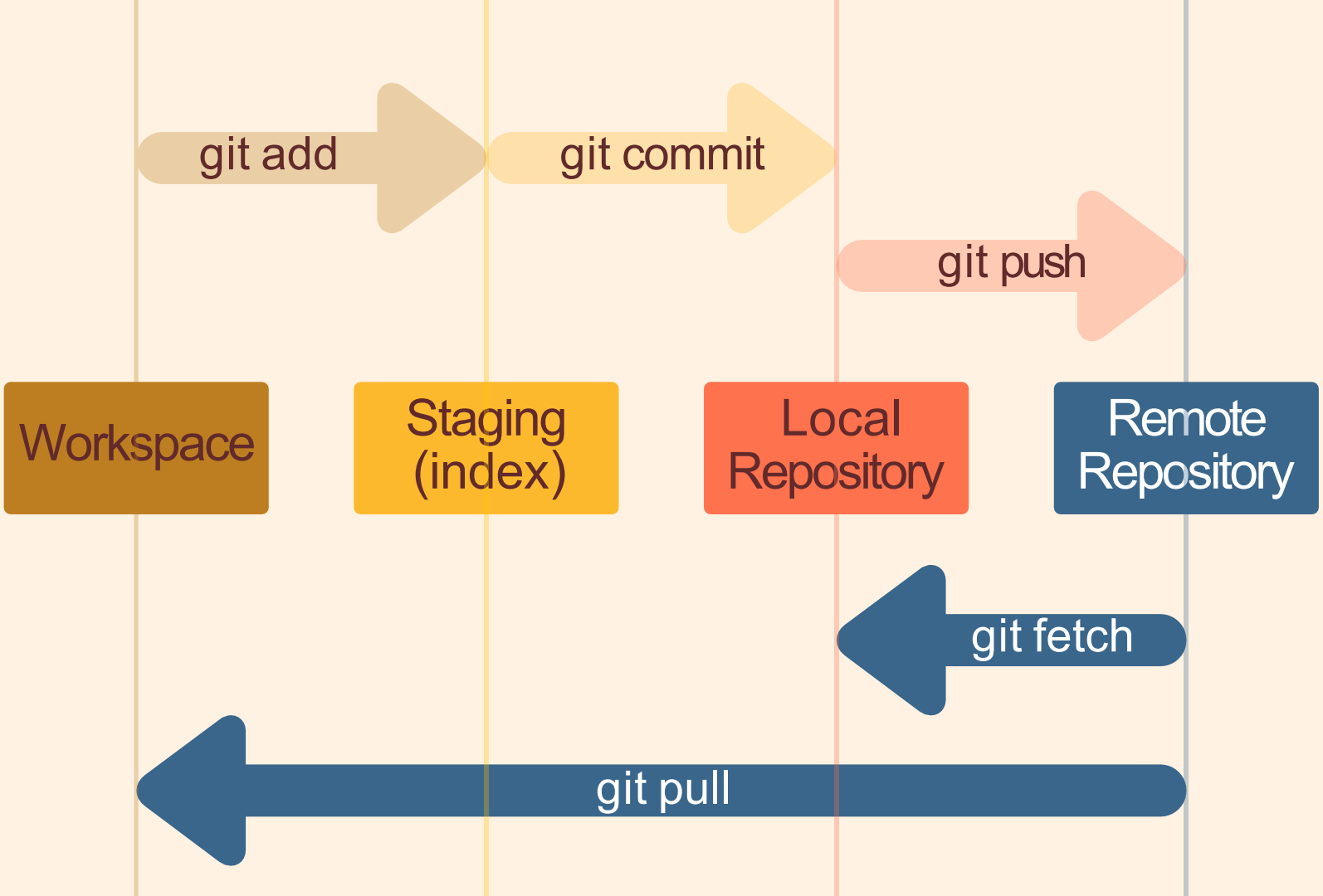

Github
git fetch origin master
Local
origin/master
master

Github

I now have those changes on my machine, but if I want to see them I have to checkout origin/master. My master branch is untouched!

Local




git add
git commit
git push
Workspace
Staging (index)
Local Repository
Remote Repository
git fetch
git pull

Pulling

git pull is another command we can use to retrieve changes from a remote repository. Unlike fetch, pull actually updates our HEAD branch with whatever changes are retrieved from the remote.

"go and download data from Github AND immediately update my local repo with those changes"

git pull = git fetch + git merge

update the remote tracking branch with the latest changes from the remote repository

update my current branch with whatever changes are on the remote tracking branch

git pull

To pull, we specify the particular remote and branch we want to pull using git pull <remote> <branch>. Just like with git merge, it matters WHERE we run this command from. Whatever branch we run it from is where the changes will be merged into.

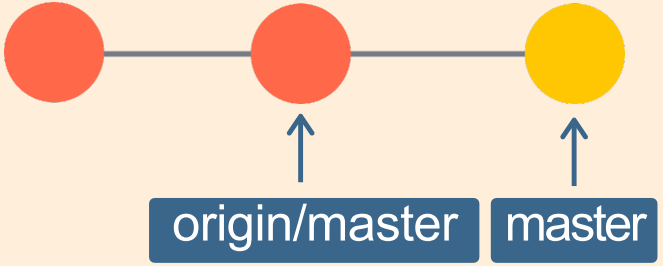
```
❯ git pull <remote> <branch>
```

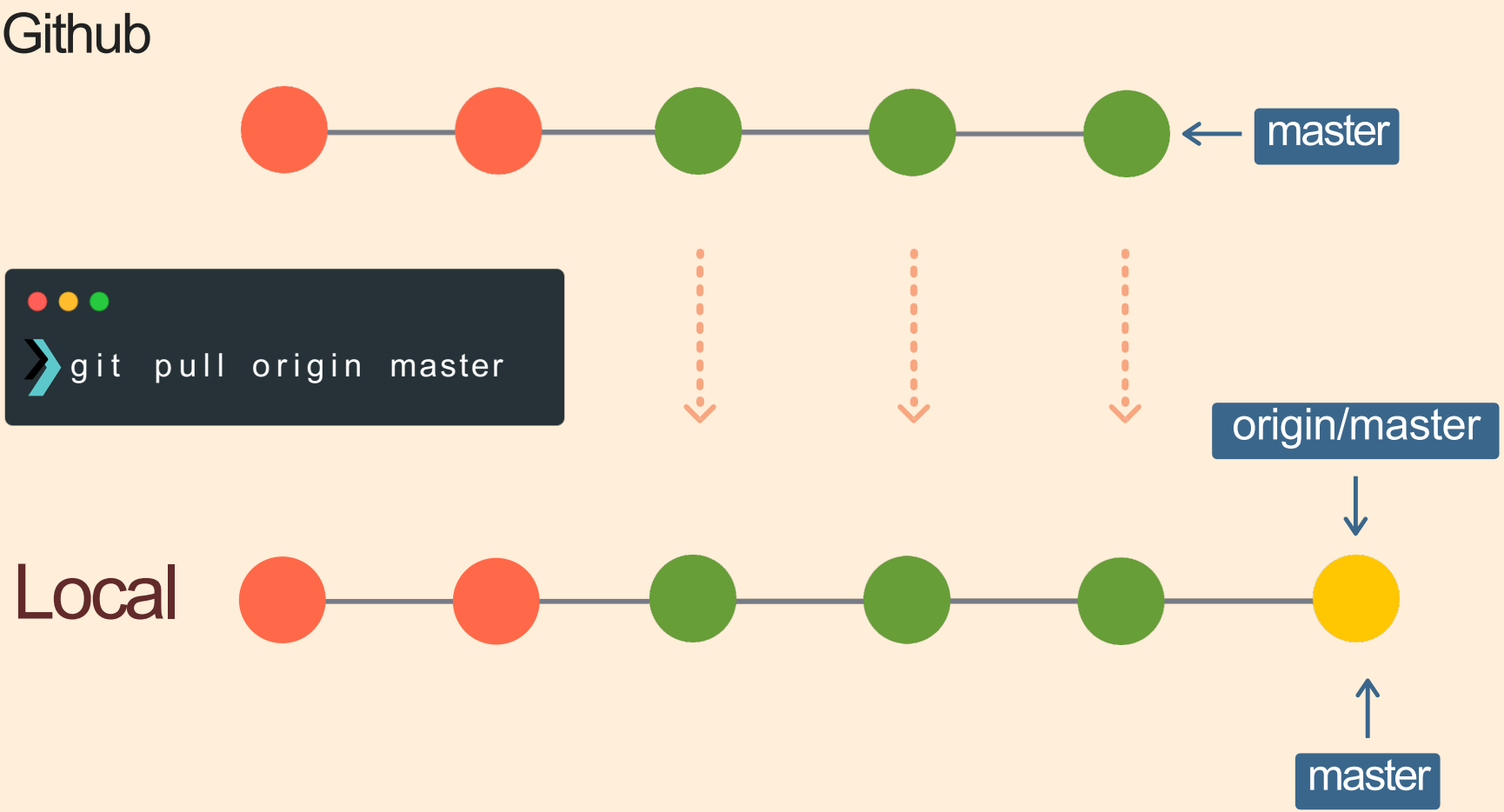
git pull origin master would fetch the latest information from the origin's master branch and merge those changes into our current branch.

pulls can result in merge conflicts !

Github

Local

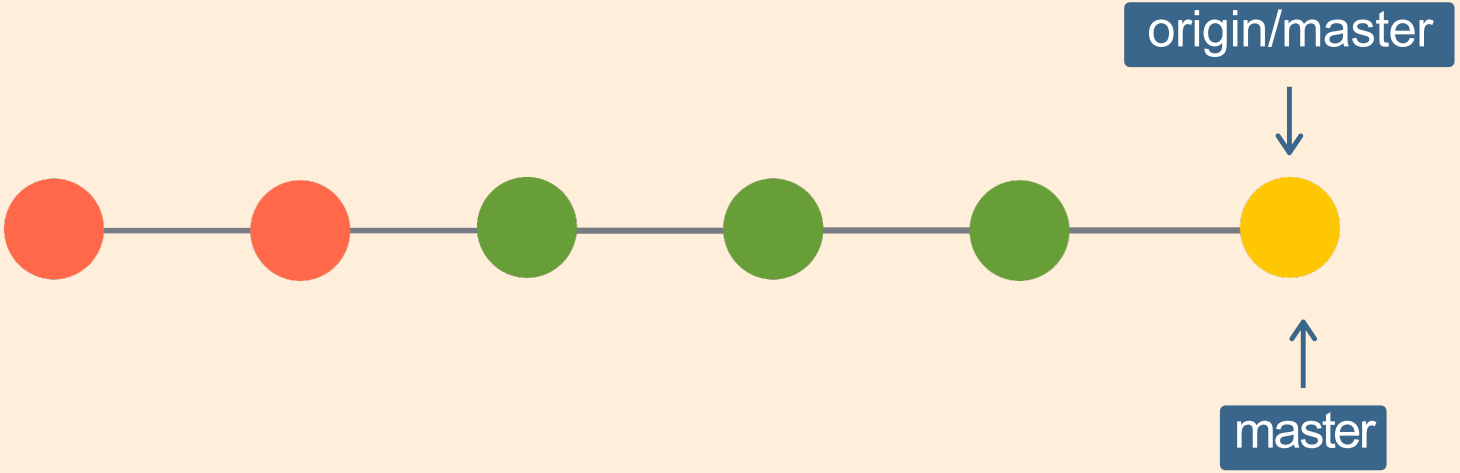



Github
master
git pull origin master
origin/master
Local
master

Github

I now have the latest commits from origin/master on my local master branch (assuming I pulled while on my master branch)

Local



Github

I have a commit locally that is not on Github.
When I pulled, it was merged with the new commits.
As with any other merge, this can result in merge conflicts.

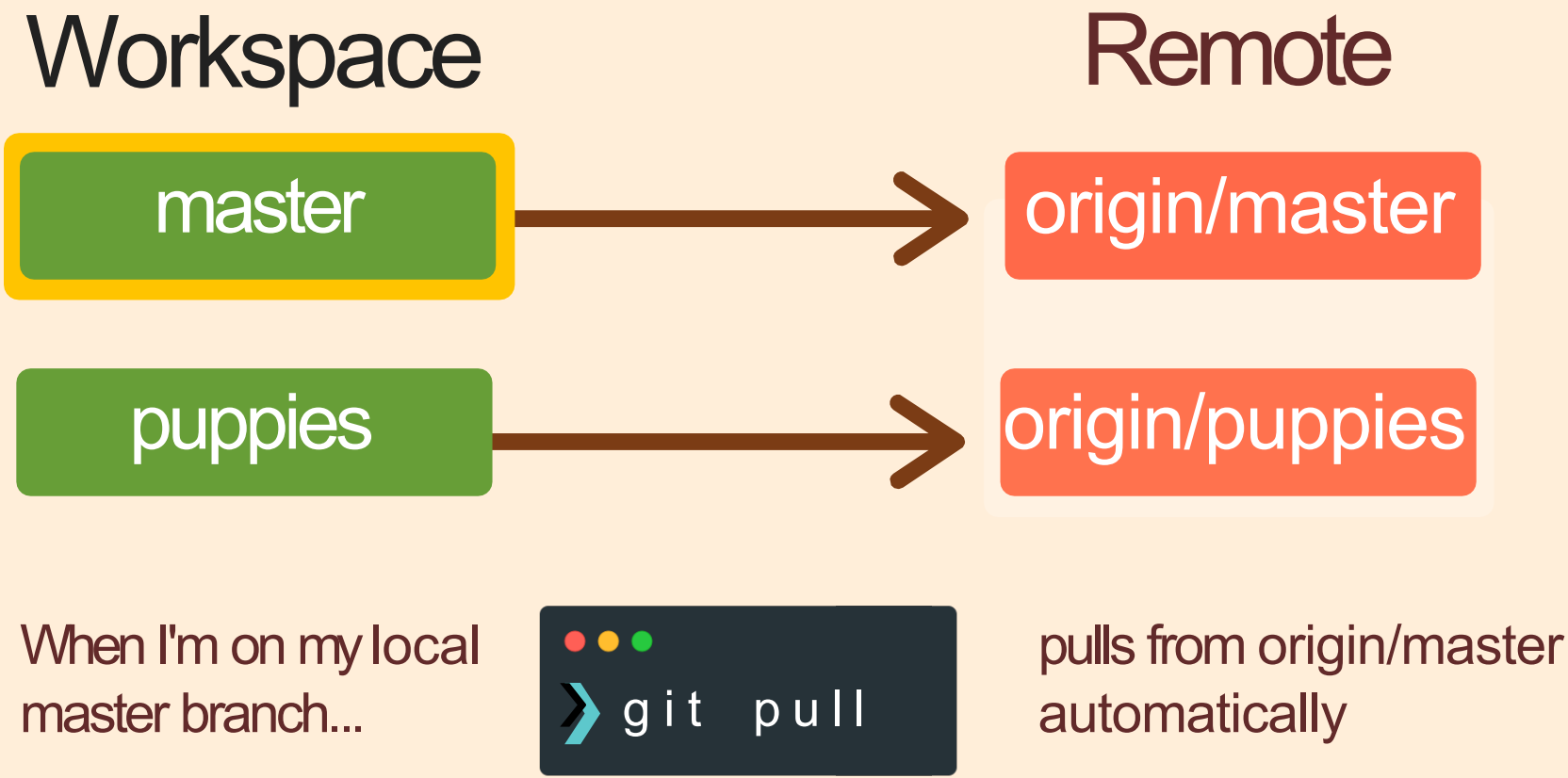
An even easier syntax!

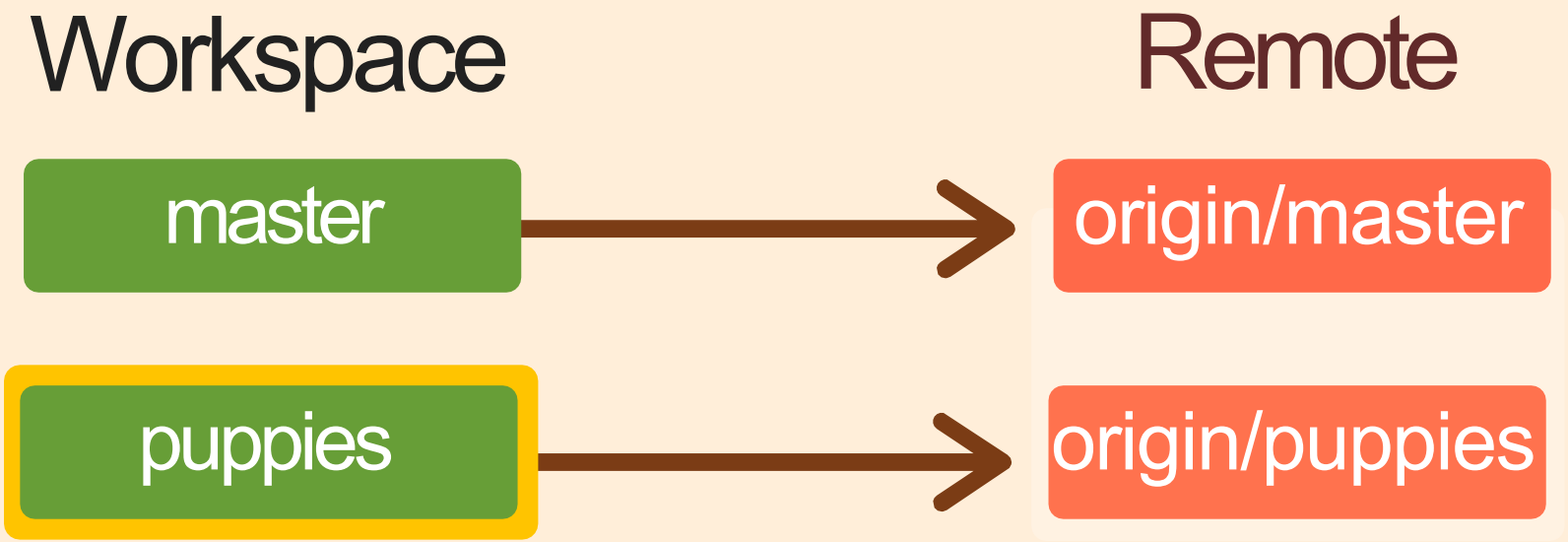
If we run git pull without specifying a particular remote or branch to pull from, git assumes the following:

- remote will default to origin
- branch will default to whatever tracking connection is configured for your current branch.

```
> git pull
```

Note: this behavior can be configured, and tracking connections can be changed manually. Most people dont mess with that stuff

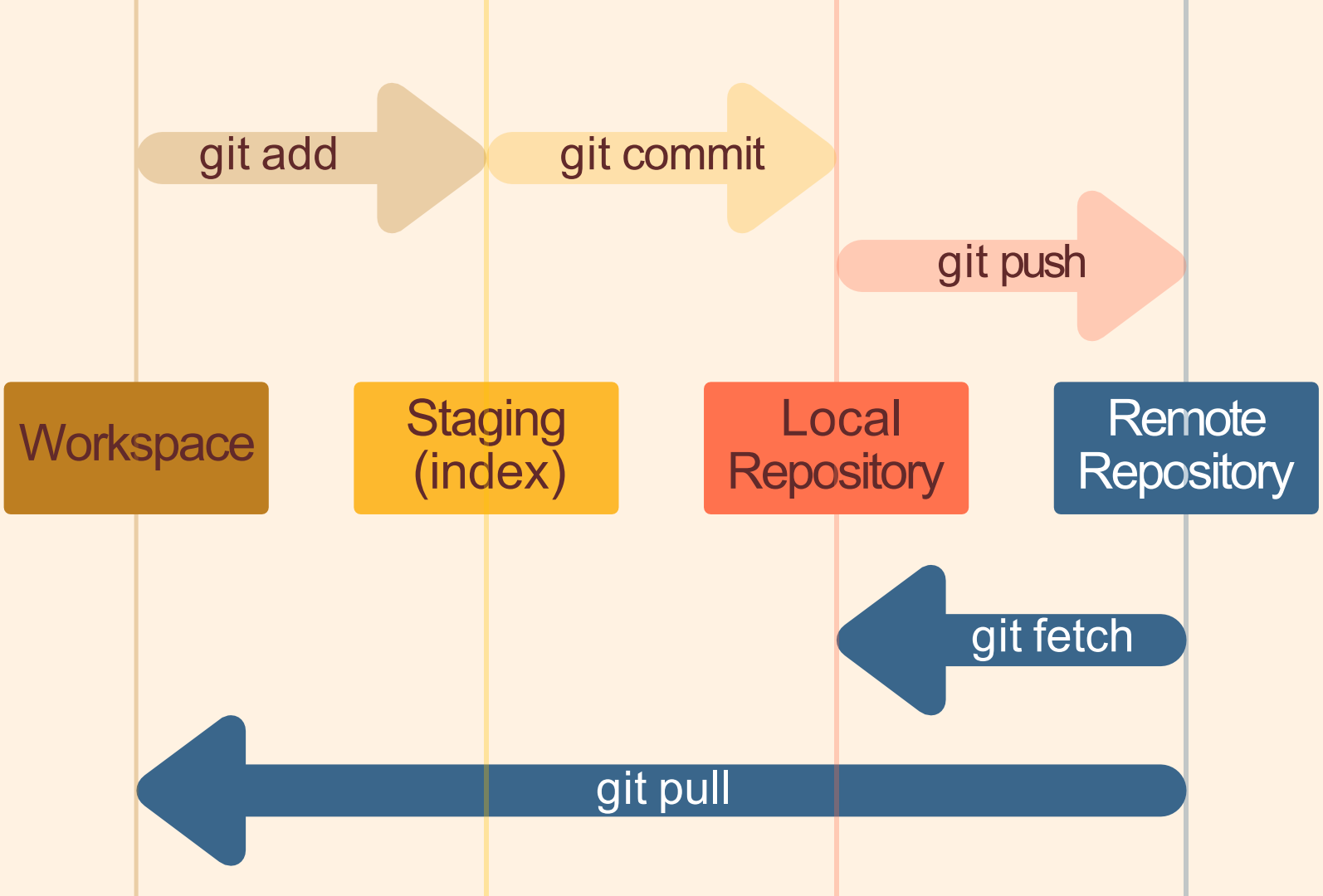

Workspace
Remote
master
origin/master
puppies
origin/puppies
When I'm on my local master branch...
git pull
pulls from origin/master automatically



When I'm on my local puppies branch...

```
❯ git pull
```

pulls from origin/puppies automatically


git add
git commit
git push
Workspace
Staging (index)
Local Repository
Remote Repository
git fetch
git pull

git fetch

- Gets changes from remote branch(es)
- Updates the remote-tracking branches with the new changes
- Does not merge changes onto your current HEAD branch
- Safe to do at anytime

git pull

- Gets changes from remote branch(es)
- Updates the current branch with the new changes, merging them in
- Can result in merge conflicts
- Not recommended if you have uncommitted changes!

Week 2 Getting Started with Git

- Using **git fetch** will download changes from the GitHub remote repository, however you will not see those changes be part of the files you have in the working directory.
- Fetch allows you to grab additional work done on the remote master branch, without needing to merge it in your working directory files.

Week 2 Getting Started with Git

- Using **git fetch** makes sense when you're working with others and want to see what changes they've made but aren't ready to overwrite your own files yet.
- Also if you are simply working by yourself, you may want to just fetch remote changes without overwriting your latest work (later we'll discover branches are a better way of doing this).

Week 2 Getting Started with Git

- When using fetch, often you'll just use:
  - **git fetch**
- But you can specify to fetch specific branches using:
  - **git fetch <remote> <branch>**
  - **git fetch origin <branch>**

Week 2 Getting Started with Git

- Using **git pull** makes sense when you want to directly grab changes from the remote branch and directly merge them into your current branch.
- This means you will literally update the files in your working directory to match up and merge with the remote branch.

Week 2 Getting Started with Git

- If you're a solo developer working on a single master branch, you often skip using a combination of **git fetch** and **git merge** and go straight for a **git pull**.
- We're not going to condone this as the "best practice", but we also want to be realistic of the workflow of a solo developer on a single branch.

Week 2 Getting Started with Git

- In the next Weeks when we learn about branches and merging changes more carefully or stashing current changes, we'll have a more nuanced understanding of using **git fetch** vs. **git pull**.
- In general you should pull before pushing to a remote branch, to make sure you're in sync.

Week 2 Getting Started with Git

- Let’s explore these git methods by making changes directly on GitHub on the remote origin and then bring those changes to our local repository and working directory.

Week 2
Exercise and Solution

Week 2 Getting Started with Git

- Let’s test your understanding of what you’ve learned so far with a quick exercise.
- We’ll give you a list of tasks to complete.

Week 2 Getting Started with Git

- **Tasks:**
  - Create a new local repo
  - Add a text file locally
  - Create the remote repo on GitHub
  - Push the Local repo changes to GitHub
  - Create a new file on GitHub
  - Pull these changes from GitHub to the local repo

Week 2 Solution