Beginner

Become a pro with Git

Antonio Oliveira Jr

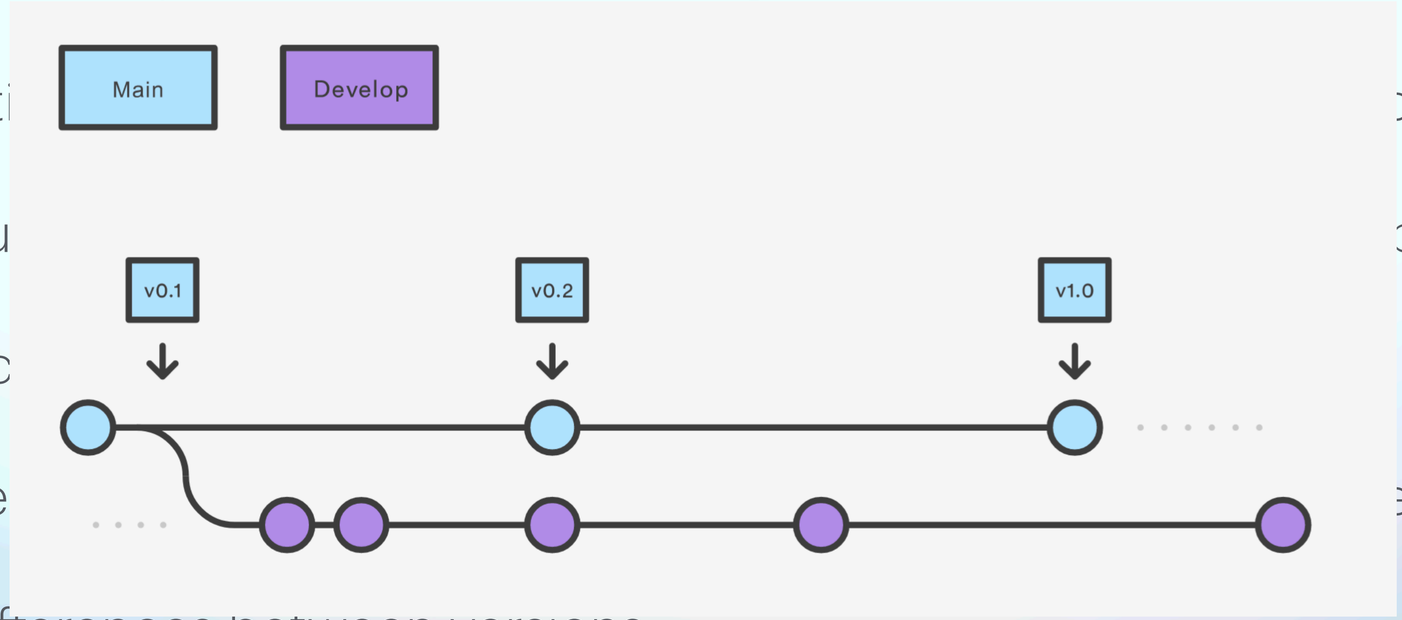
CTBP Tech Talk - 10/18/2023

What are Git and GitHub

- **Git** is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency
- **GitHub** is a web-based Git repository hosting service, which offers all of the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features.

Introduction to Version Control & Git

- Version Control: A system that records changes to files over time.
  - Keeps multi ce code)
- Git: A distribu orate.
  - Requests c
  - Allows “che eone else is working on
  - Displays differences between versions
- Each user has a complete copy of the repository



Introduction to Version Control & Git

- Version Control: A system that records changes to files over time.
  - Keeps multiple (older and newer) versions of everything (not just source code)
- Git: A distributed version control system allowing multiple users to collaborate.
  - Requests comments regarding every change
  - Allows “check in” and “check out” of files so you know which files someone else is working on
  - Displays differences between versions
- Each user has a complete copy of the repository

Installing Git

- **Mac**
  - Using Homebrew: **brew install git**
  - Direct download from git-scm.com
- **Linux**
  - For Debian/Ubuntu: **sudo apt-get install git**
  - For Fedora: **sudo dnf install git**
- **Windows**
  - **Don’t do it with yourself**

Verify installation:

git --version

Introduce yourself to Git

- On your computer, open the Git Shell application.
  - **git config --global user.name “Antonio Jr"**
  - **git config --global user.email “abo1@rice.edu”**

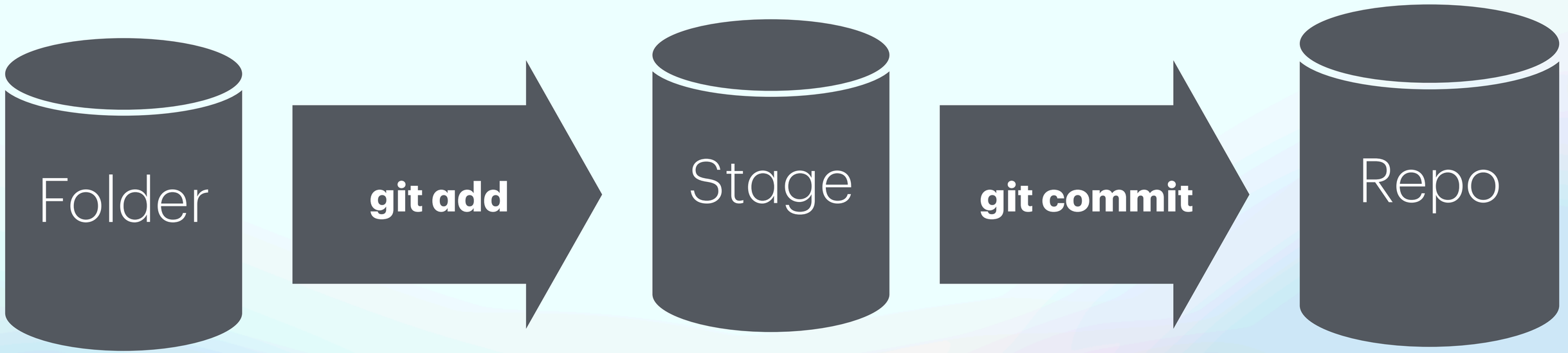
You only need to do this once

Git init and .git

- Create a directory
  - **mkdir project_git**
- Create a git inside the folder
  - **git init**
- When you said git init in your project directory, or when you cloned an existing project, you created a repository
- The repository is a subdirectory named **.git** containing various files
- The dot indicates a “hidden” directory
- You do not work directly with the contents of that directory; various git commands do that for you

Workflow

Add a file or change it



Git add

- Create a file inside the folder
  - **vim code_1.py**
- Check the status of your repository
  - **git status** <- Use it all the time
- Let put it on Stage by command add
  - **git add code_1.py**

Git commit

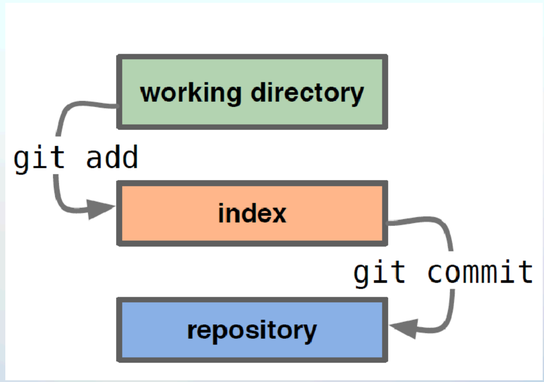
- Check the git again
  - **git status** <- Use it all the time
- Now we need to commit it to repo.
  - **git commit -m “create the first code of my repo”**

Commit messages

- In git, “Commits are cheap.” Do them often.
- When you commit, you must provide a one-line message stating what you have done
- Terrible message: “Fixed a bunch of things”
- Better message: “Corrected the calculation of median scores”
- Commit messages can be very helpful, to yourself as well as to your team members
- You can’t say much in one line, so commit often

Another commit

- Open the file and create some code.
- Now try to commit it
- Remember every time you changed something you need to stage it
  - **git add .**
- Now you can commit it =)



Always check with **git status**

Take a look on the log

- Inside the repository folder, make this command:
  - **git log**

```
(base) antonio@antonios-air-2 proj1 % git log
commit 878ca2a85746948b55b0f17914ad58a9e90404e5 (HEAD -> master)
Author: Antonio Jr <abo1@rice.edu>
Date:   Tue Oct 17 16:49:40 2023 -0500

    add my second file

commit 13fa6d83ee1935e5e87c5828da79216e26c9e00e
Author: Antonio Jr <abo1@rice.edu>
Date:   Tue Oct 17 16:48:46 2023 -0500

    insert a print function

commit 603b6c5bca852c704ef4430a7f6bc9d9e60fd854
Author: Antonio Jr <abo1@rice.edu>
Date:   Tue Oct 17 14:29:00 2023 -0500

    create the first code of my repo
(base) antonio@antonios-air-2 proj1 %
```

Back to the Past

- Command to change between commits
  - **git checkout <6 first digit of hash>**
- To return you can use the command:
  - **git checkout master (or main)**

```
(base) antonio@antonios-air-2 proj1 % git log
commit 878ca2a85746948b55b0f17914ad58a9e90404e5 (HEAD -> master)
Author: Antonio Jr <abo1@rice.edu>
Date:   Tue Oct 17 16:49:40 2023 -0500

    add my second file

commit 13fa6d83ee1935e5e87c5828da79216e26c9e00e
Author: Antonio Jr <abo1@rice.edu>
Date:   Tue Oct 17 16:48:46 2023 -0500

    insert a print function

commit 603b6c5bca852c704ef4430a7f6bc9d9e60fd854
Author: Antonio Jr <abo1@rice.edu>
Date:   Tue Oct 17 14:29:00 2023 -0500

    create the first code of my repo
(base) antonio@antonios-air-2 proj1 %
```

Back to the Past

```
(base) antonio@antonios-air-2 proj1 % git log
commit 603b6c5bca852c704ef4430a7f6bc9d9e60fd854 (HEAD)
Author: Antonio Jr <abo1@rice.edu>
Date:   Tue Oct 17 14:29:00 2023 -0500

    create the first code of my repo
(base) antonio@antonios-air-2 proj1 %
```

Screenshot

- Command to change between commits
  - **git checkout <6 first digit of hash>**
- To return you can use the command:
  - **git checkout master (or main)**

```
(base) antonio@antonios-air-2 proj1 % git checkout 603b6c5
Note: switching to '603b6c5'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 603b6c5 create the first code of my repo
(base) antonio@antonios-air-2 proj1 %
```

Branch (theory)

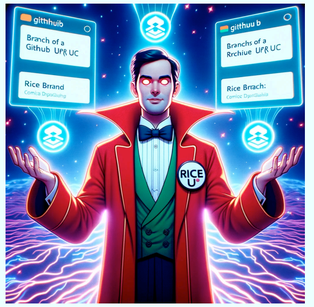
Just commit in a master branch

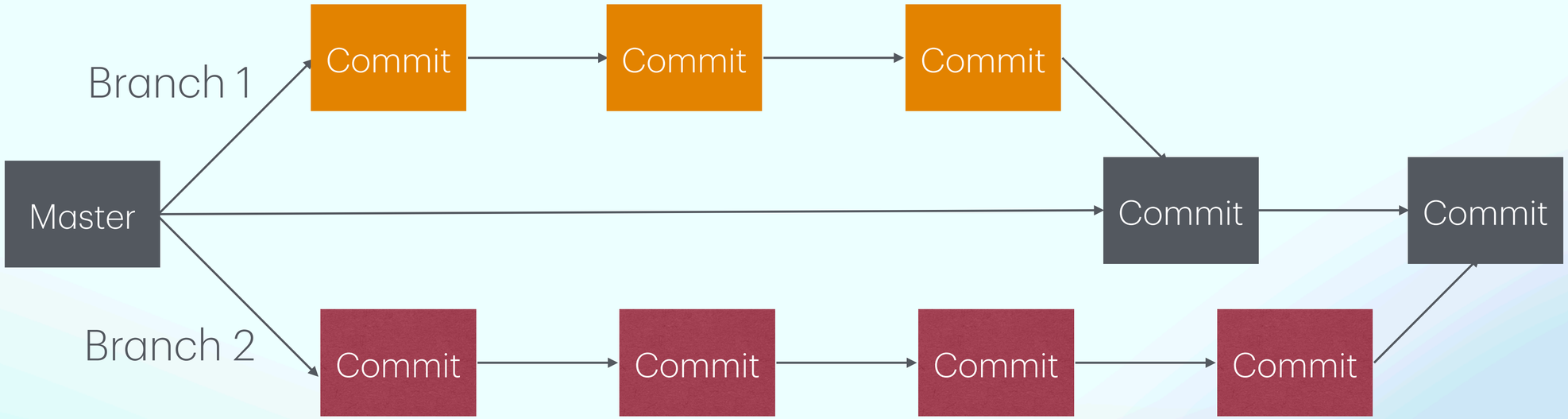
Just one timeline

Branch (theory)

Multiverse (of madness?)

Branch (theory)





Branch (Hands-on)

- Command to create a branch:
  - **git branch <name>**
- Check it:
  - **git log**
  - **git status**
- To change the branch:
  - **git checkout <name of branch>**

Branch (Hands-on)

- Make some change on the file and commit it
  - **git add .**
  - **git commit -m “ first commit in the branch”**
- Check it:
  - **git log**
  - **git status**
- To change the branch:
  - **git checkout master**

Always check with **git status**

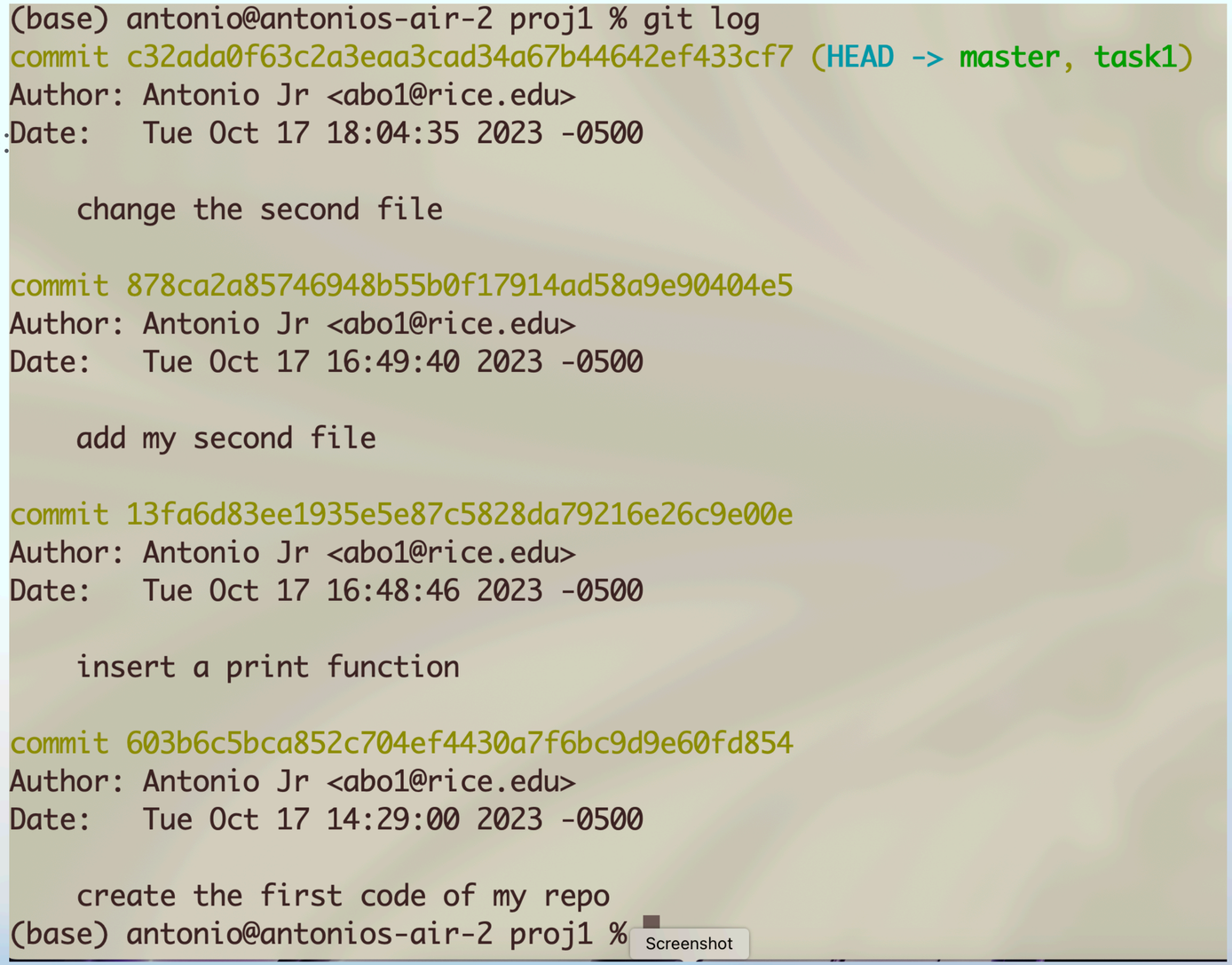
```
(base) antonio@antonios-air-2 proj1 % git log
commit c32ada0f63c2a3eaa3cad34a67b44642ef433cf7 (HEAD -> task1)
Author: Antonio Jr <abo1@rice.edu>
Date:   Tue Oct 17 18:04:35 2023 -0500

    change the second file

commit 878ca2a85746948b55b0f17914ad58a9e90404e5 (master)
Author: Antonio Jr <abo1@rice.edu>
Date:   Tue Oct 17 16:49:40 2023 -0500

    add my second file

commit 13fa6d83ee1935e5e87c5828da79216e26c9e00e
Author: Antonio Jr <abo1@rice.edu>
Date:   Tue Oct 17 16:48:46 2023 -0500

    insert a print function

commit 603b6c5bca852c704ef4430a7f6bc9d9e60fd854
Author: Antonio Jr <abo1@rice.edu>
Date:   Tue Oct 17 14:29:00 2023 -0500

    create the first code of my repo
(base) antonio@antonios-air-2 proj1 %
```

Screenshot

Branch merging (Hands-on)

Always check with **git status**

- To merge the branch that you work, uses:
  - **git merge <name of branch>**
  - **git merge task1**



GitHub

- Lets create a login on github.com
- Create a repo
- Add the repo to GitHub
  - **git remote add origin git@github.com:junioreif/git_exemplo.git**
- Change the name of Master to Main
  - **git branch -M main**
- Using command push to send the files to GitHub
  - **git push -u origin main**

GitHub - send a commit to online version

- In the folder repo, create a new branch:
  - **git branch feature2**
  - **Git checkout feature2**
- Add a new file and modified the first one
  - **git add .**
  - **git commit -m “ my new feature”**
- Back to the main branch and use the command push
  - **git merge main feature2**

GitHub - send a commit to online version

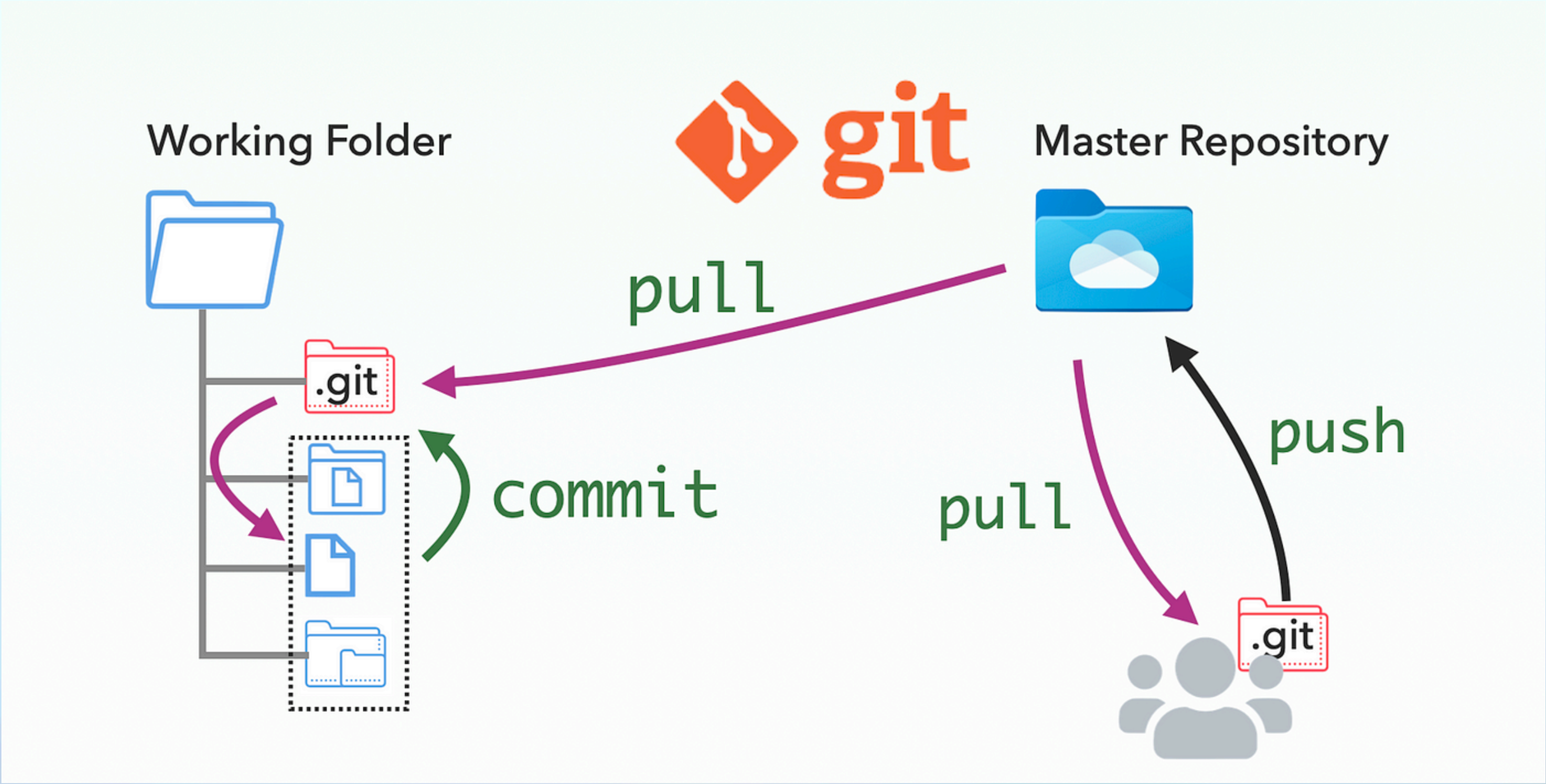
- Use the command push to submit the new main to the online repo
  - **git push main origin**
- The command push work as **git push <what> <where>**
- Check the online version on GitHub

GitHub

- **Tips**
  - Before create a new branch, always use the command Pull to get the updated version of the origin (online version)
    - **Git pull origin main**
  - Always uses **git status** and **git log** to follow the changes

GitHub workflow

- **Tips**



GitHub Fork & Pull: A Collaborative model

- A fork is a copy of a repository that you manage. Forks let you make changes to a project without affecting the original repository. You can fetch updates from or submit changes to the original repository with pull requests.
- A great example of using forks to propose changes is for bug fixes. Rather than logging an issue for a bug you've found, you can:
- Fork the repository.
- Make the fix.
- Submit a **pull request** to the project owner.

GitHub clone a repo

- Create a folder
  - **mkdir <folder name>**
- Get the link of the repo on GitHub
  - **git clone git@github.com:USERNAME/GitExercise.git**