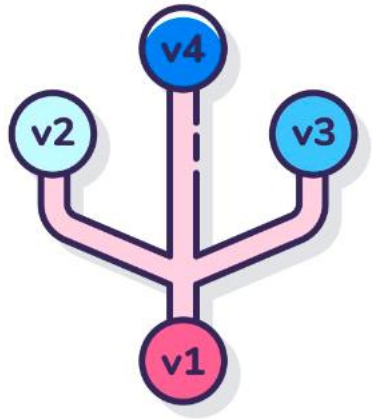
Git and GitHub

CMPUT 301

What is Git?

- Git is a version control system
- It tracks changes to your project over time
- Each saved version is called a **commit**
- You can compare changes, restore older versions, and work on separate branches



What is GitHub?

- GitHub is an online platform for hosting Git repositories
- It lets teams share code and collaborate
- It supports pull requests, issues, code review, project boards, and automation
- In CMPUT 301, your team will use GitHub to manage project work

Git vs. GitHub

Git	GitHub
A version control system	A website for hosting Git repositories
Installed on your computer	Accessed through a browser, Git client, or terminal
Tracks changes in project files	Stores a shared copy of the repository online
Saves project history through commits	Lets teammates share, review, and merge changes
Works without an internet connection	Used to *collaborate* with other people online

Git tracks the project history while GitHub helps a team share that history

Installing Git

- Installing Git is only needed if you want to use Git commands directly
  - GitHub can be used through the website, GitHub Desktop, Android Studio, or the terminal
- Check whether Git is already installed:

```
[student@cmput301 git --version
git version 2.50.1 (Apple Git-155)
student@cmput301
```

- If the command is not found, follow these instructions to install Git

Optional: Setting a Git Commit Identity

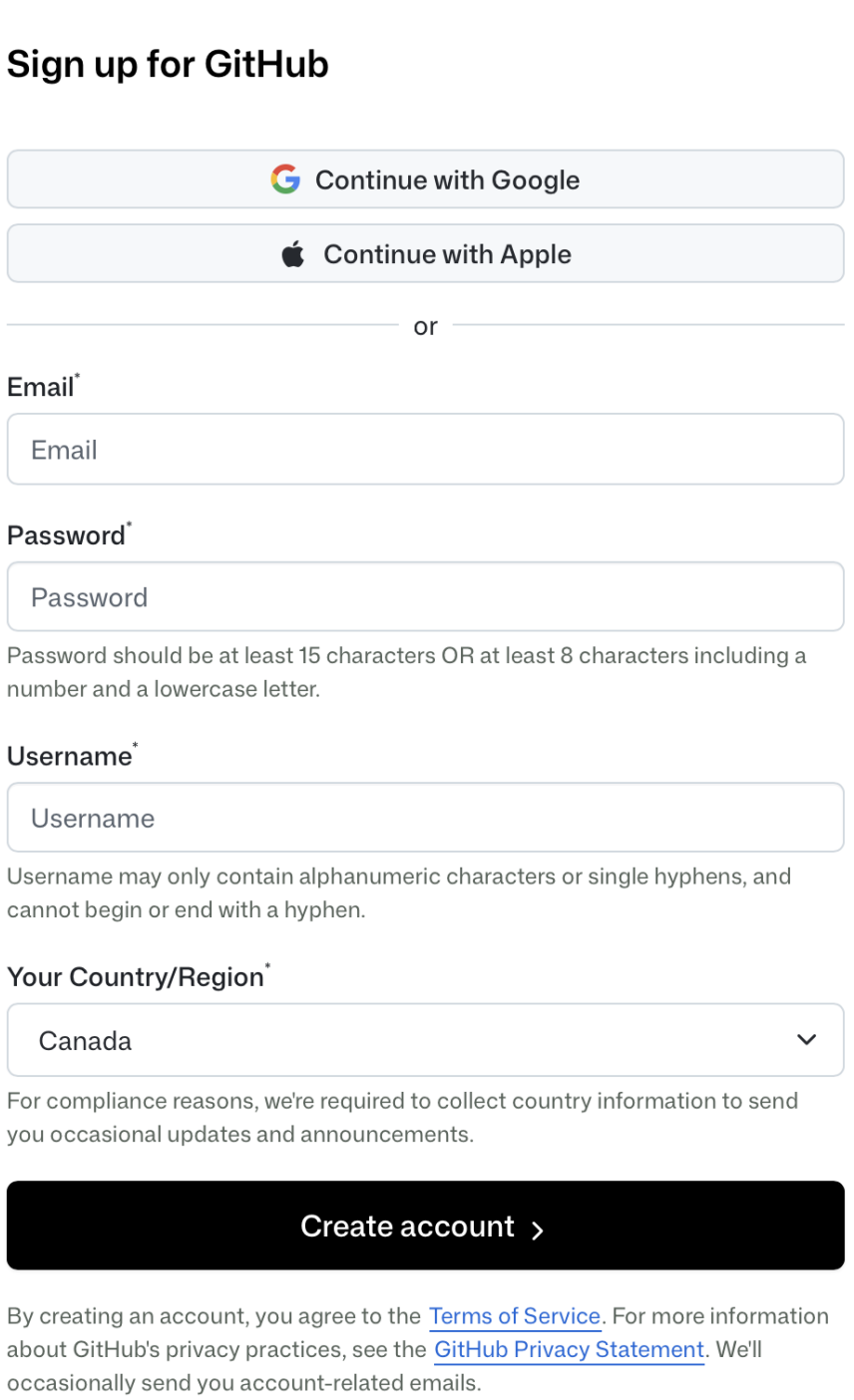
- Git records a name and email on each commit.
- If you use Git from the terminal, set these once:

```
$ git config --global user.name "John Smith"
$ git config --global user.email js@gmail.com
```

- More information on setting up editors for Git here: https://swcarpentry.github.io/git-novice/02-setup/
- GitHub Desktop and Android Studio may help configure this automatically (you must have Git installed to use these)

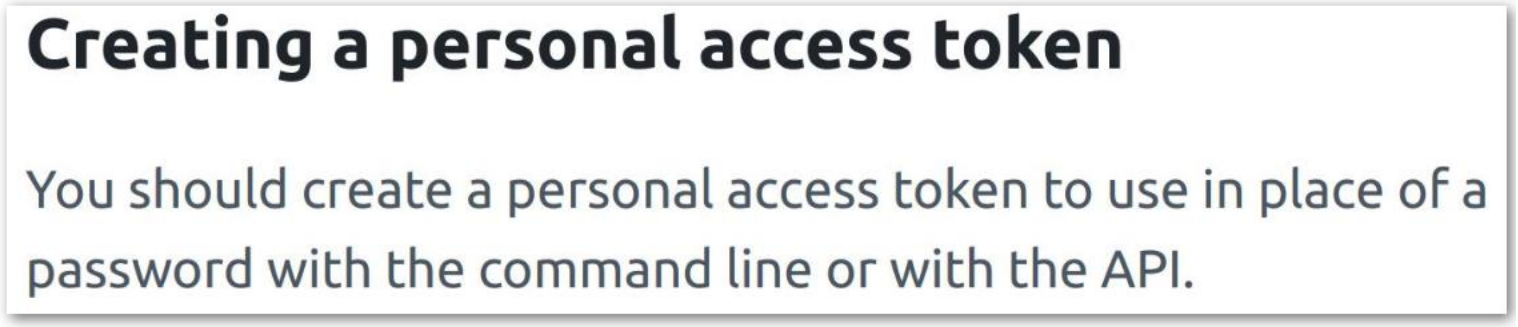
Using GitHub

- First, you need a GitHub account
  - If you do not have one, create one by following the link here
  - Remember to have a recognizable username so your teammates can identify you
  - Enable two-factor authentication if required



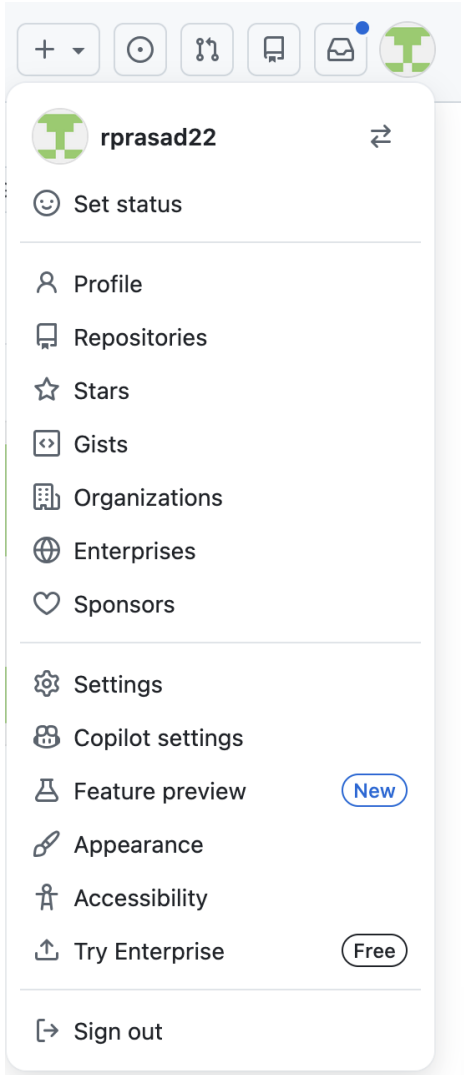
Personal Access Tokens

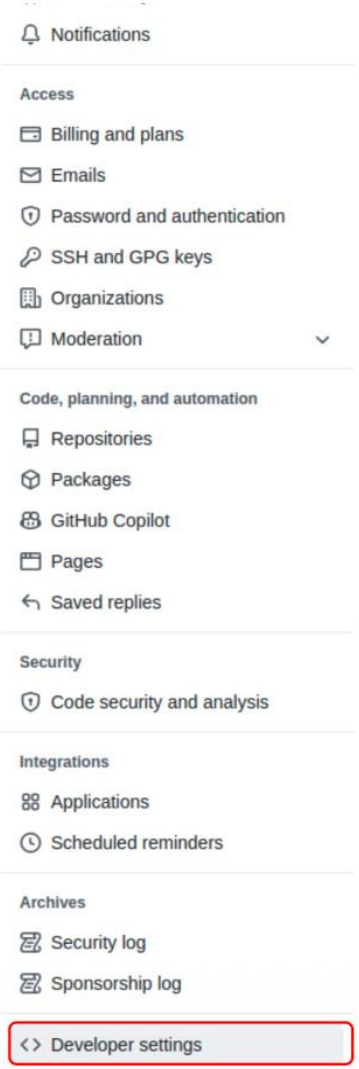
- GitHub does not accept account passwords for Git operations
- If Git asks for a password when you run git push, git pull, or git clone, use a **Personal Access Token** instead.
  - Git operations need another authentication method
  - A Personal Access Token is one option

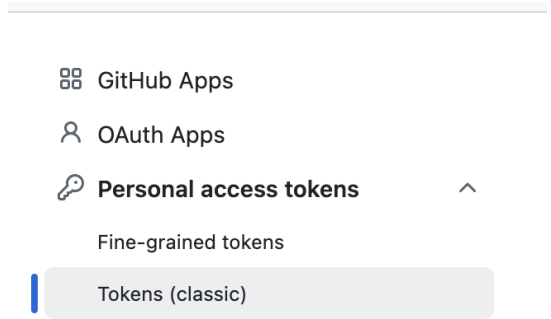


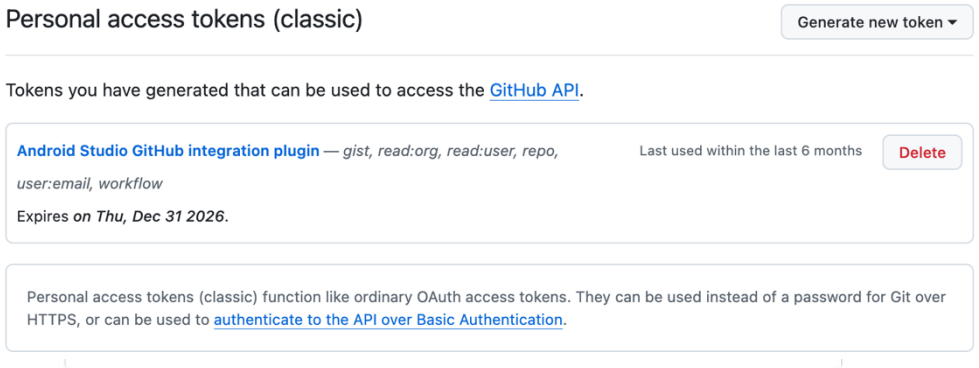
https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token

Create a Personal Access Token



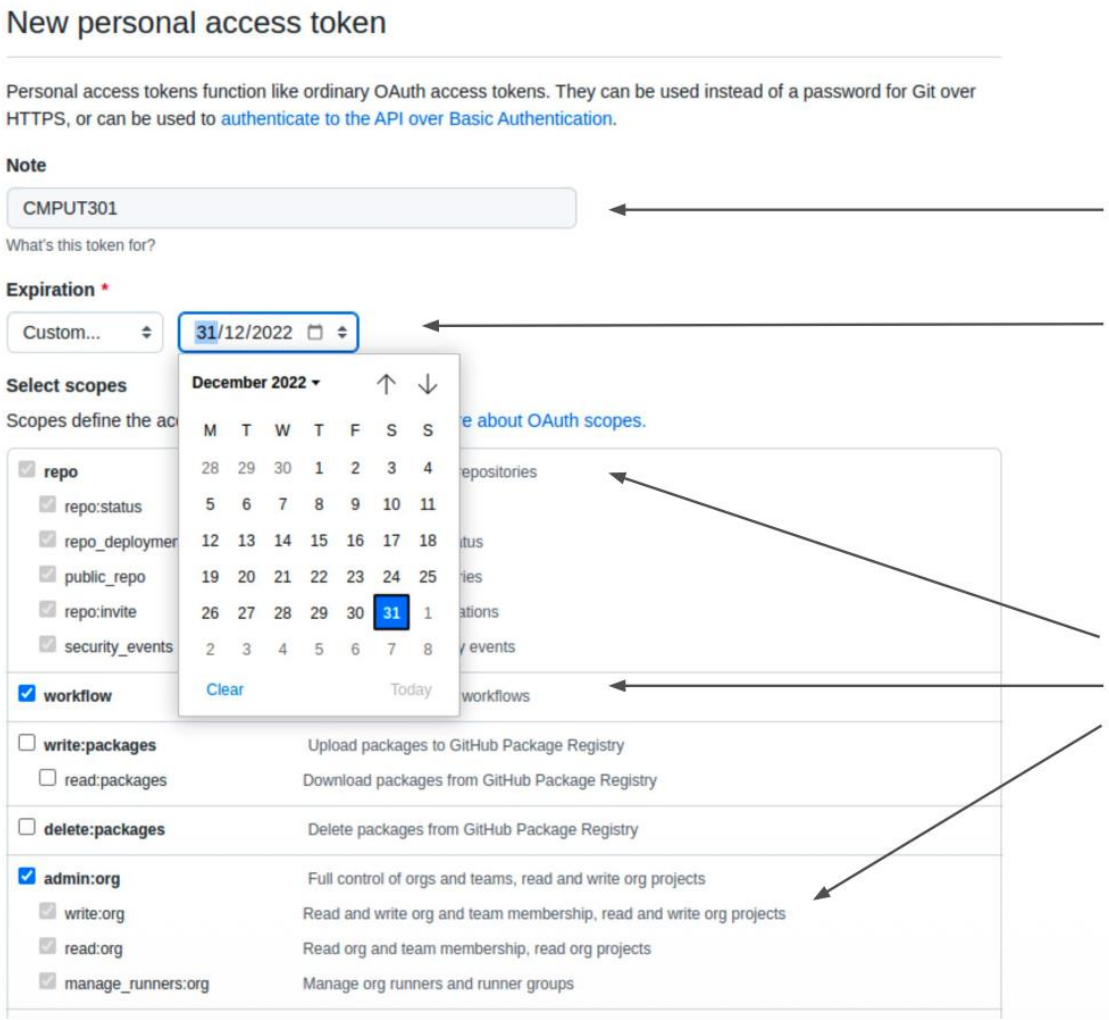






I have some tokens created already. In your case it might be empty.

Create a Personal Access Token

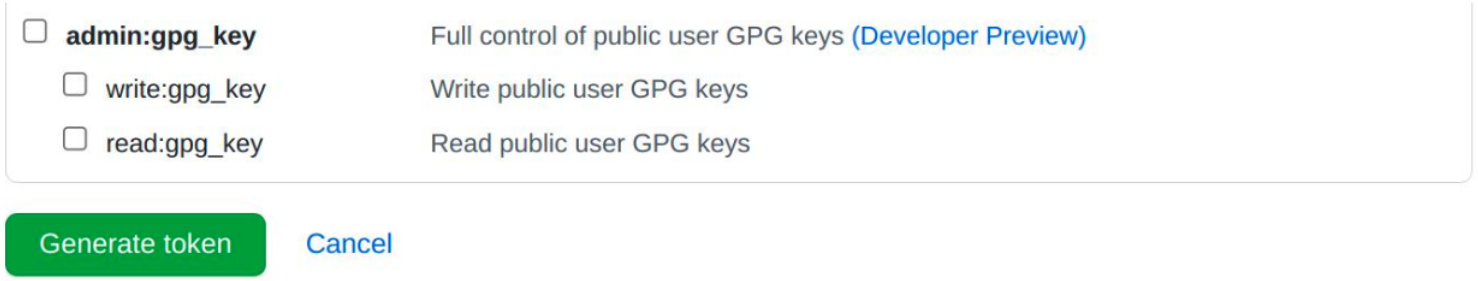


Not super important, but use a meaningful name so that you know the purpose

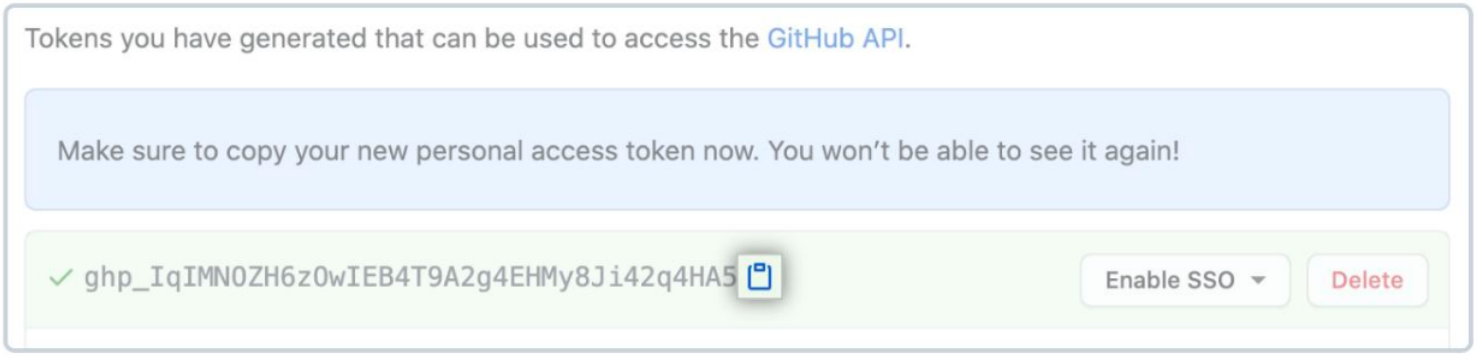
Important: keep until you finish the course

You may read the scope for more information. But these three option should enough for the project.

Create a Personal Access Token



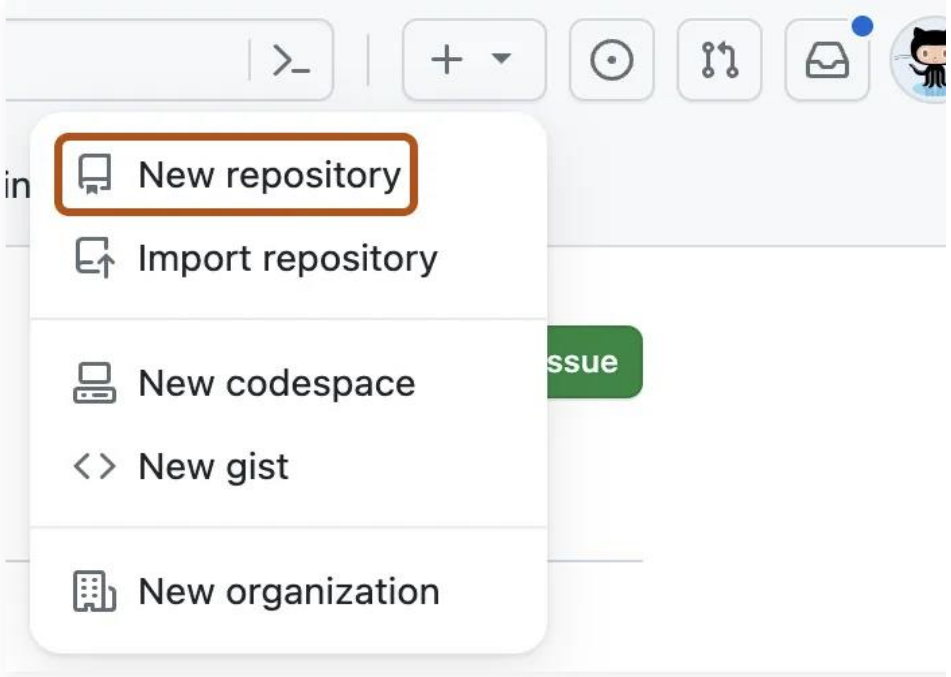
Click on the generate token



Copy the token to a file, because you will not able to see it!!!!

Hosting a Repo on GitHub

- Then, in the upper-right corner of any page, select +, then click New repository
- Type a short name for your repo, like “hello_world”, and optionally add a short description
- Choose a repository visibility
- Toggle README to On
- Click Create Repository to make the repository
- The GitHub Docs contains more information (here)



Clone a Git repo

```
[student@cmput301 mkdir GitDemo
[student@cmput301 cd GitDemo
[student@cmput301 git clone git@github.com:rprasad22/301Test.git
 Cloning into '301Test'...
 remote: Enumerating objects: 3, done.
 remote: Counting objects: 100% (3/3), done.
 remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
 Receiving objects: 100% (3/3), done.
[student@cmput301 ls
 301Test
[student@cmput301 cd 301Test
[student@cmput301 ls
 README.md
```

1. Make a directory ‘GitDemo’
2. Change current directory to ‘GitDemo’
3. Clone the repository

    You should use HTTPS to clone your repository (This example uses SSH to clone)

4. Check that it downloaded using ‘ls’

Adding a Local File to the Repo

```
[student@cmput301 more README.md
# 301Test
[student@cmput301 git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
[student@cmput301 echo "Hello world" >> README.md
[student@cmput301 more README.md
# 301TestHello world
[student@cmput301 git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
```

- 'echo "Hello world" >> README.md' adds text to the README.md file
- 'git status' shows which files changed

Adding a Local File to the Repo

- ‘git add’ stages the file for the next commit
- ‘git commit –m “Update README”’ saves the change locally
- ‘git push origin main’ uploads the commit to GitHub

```
student@cmput301 git add README.md
student@cmput301 git commit -m "Update README"
[main 046e5a3] Update README
 1 file changed, 1 insertion(+), 1 deletion(-)
student@cmput301 git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 269 bytes | 269.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:rprasad22/301Test.git
   45b8dbd..046e5a3  main -> main
```

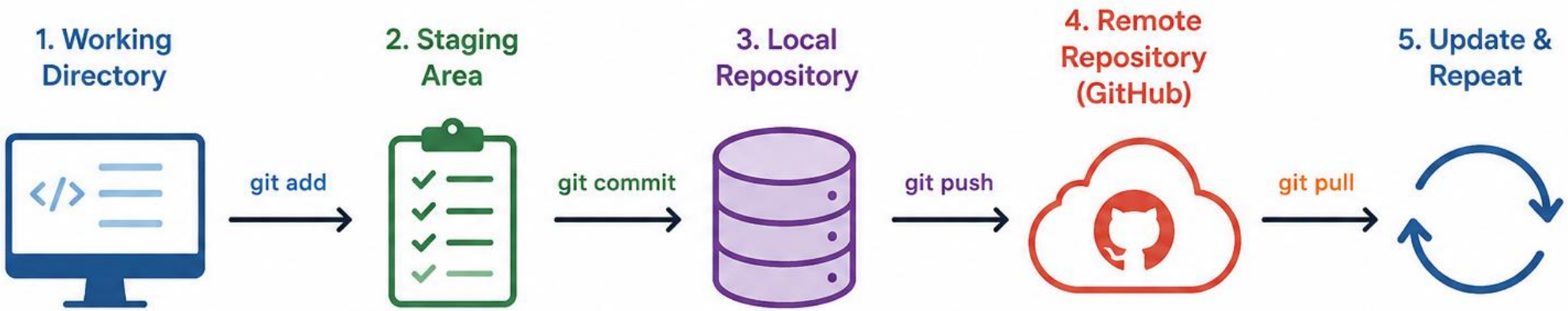
Get Updates from GitHub

- ‘git pull origin main’ downloads the latest changes from GitHub from the main branch into your local copy.
- clone once → pull often → edit/add → commit → push

```
[student@cmput301 git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 906 bytes | 151.00 KiB/s, done.
From github.com:rprasad22/301Test
   046e5a3..6b82a14  main       -> origin/main
Updating 046e5a3..6b82a14
Fast-forward
 README.md | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
[student@cmput301 more README.md
# 301Test
dlroW olleH
```

Git Workflow

The basic cycle of working with Git and GitHub.



You edit files in your project.

You stage the changes you want to commit.

Git saves a snapshot of your changes locally.

Your commits are stored on GitHub and shared with others.

Get the latest changes, repeat the cycle.

Clone once → Pull often → Edit → Add → Commit → Push → Repeat

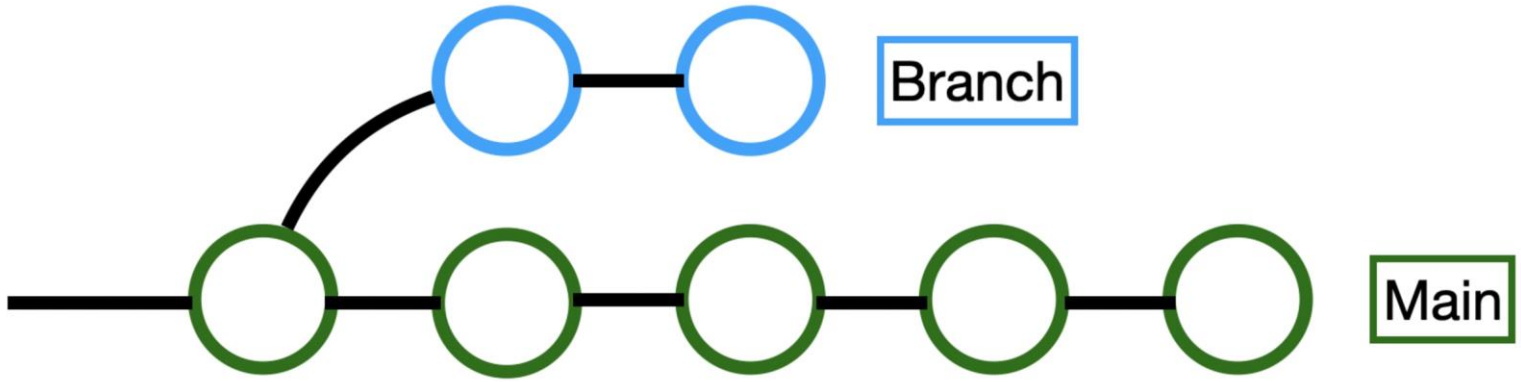
Common Commands	`git add <file>`	`git commit -m "..."`	`git push origin main`	`git pull origin main`	`git switch <branch>`
	Stage changes	Save changes locally	Upload to GitHub	Get latest changes from GitHub	Switch branches

Practice the Common Commands

- Create two new text files and push those to Github repo
  - `echo "Test 1" >> test1.txt`
  - `echo "Test 2" >> test2.txt`
  - `git add .`
  - `git commit -m "<message>"`
  - `git push origin master`
- Useful git commands
  - `git pull origin <branch>`
  - `git reset -` remove files from the staging area
  - `git rm --cached <filename> -` remove a file from working index
  - `git rm -f <filename> -` remove a file forcefully `(-f)`
  - `git rm -rf <dir name> -` remove a directory forcefully `(-r` recursively)
  - `git log -` view commit history
  - `git clone <https URL>`

Git Branches: What is a Branch?

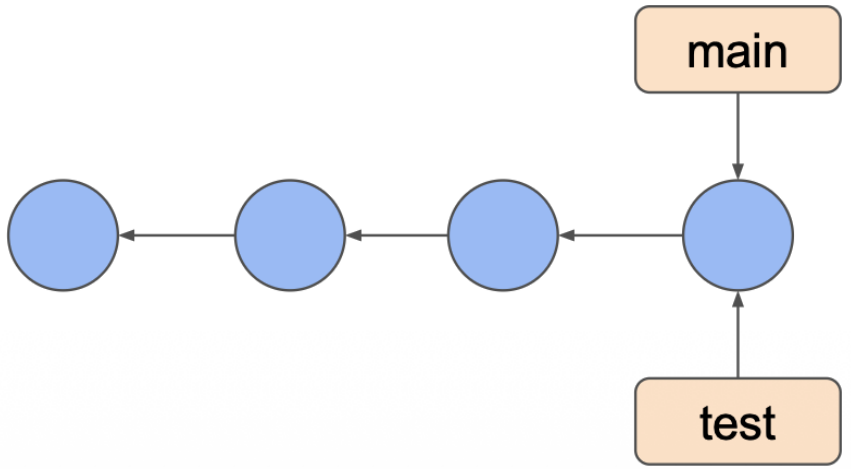
- A branch is a separate line of work in the same repository
  - Branch shows the evolution of your project (commits)
- Use branches to work on features without changing main
  - main should stay stable, while new work should happen on a branch
- Each commit has SHA-1 value which allows you to revert to that state



Git Branches

- Default branch -- main
- Create a branch in command line using ‘git branch <branch name>’ or ‘git switch -c <branch name>’
  - ‘git switch -c <branch name>’ creates the branch and switches to it
- Use git switch <branch name> to switch to that branch

```
git branch test
```



Push a Branch to GitHub

- Make a change and commit it
  - Do some work on the test branch, then commit those changes!

```
[student@cmput301 echo "branch demo" >> README.md
[student@cmput301 git add README.md
[student@cmput301 git commit -m "Add branch demo"
 [test b28fe00] Add branch demo
  1 file changed, 1 insertion(+)
[student@cmput301 git push origin test
 Enumerating objects: 5, done.
 Counting objects: 100% (5/5), done.
 Writing objects: 100% (3/3), 278 bytes | 278.00 KiB/s, done.
 Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
 remote:
 remote: Create a pull request for 'test' on GitHub by visiting:
 remote:      https://github.com/rprasad22/301Test/pull/new/test
 remote:
 To github.com:rprasad22/301Test.git
  * [new branch]      test -> test
```

Pull Requests

- In team projects, branches are usually merged through a GitHub pull request
- A pull request asks the team to review and merge a branch
  - test branch → pull request → review → merge into main
- Use a pull request when:
  - Your branch is ready to share
  - You want team feedback before merging
  - Your team wants to protect main from broken code
- **Essentially**, a pull request is a team checkpoint before code becomes part of main

Switching Between Branches

- Use ‘git branch’ to check what branch you are on
- Use ‘git switch <branch name>’ to switch to another branch
  - Switching branches changes which version of the project you are looking at

```
[student@cmput301 git branch
  main
  readme-update
* test
[student@cmput301 git switch main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
[student@cmput301 git branch
* main
  readme-update
  test
```

Merge a Branch

- Merging copies changes from one branch into another

```
[student@cmput301 git switch main
 Already on 'main'
 Your branch is up to date with 'origin/main'.
[student@cmput301 git merge test
 Updating 6b82a14..b28fe00
 Fast-forward
  README.md | 1 +
  1 file changed, 1 insertion(+)
[student@cmput301 git push origin main
 Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
 To github.com:rprasad22/301Test.git
    6b82a14..b28fe00  main -> main
```

git switch main	Move to the branch receiving the changes (main)
git merge test	Bring the test branch changes into main
git push origin main	Upload the updated main branch to GitHub

Git Demo – Merge Conflict

- Using your terminal, type:
  - git switch <name of the new branch you made (not main)>
  - echo “# test 1” > README.md
  - echo “message: Test branch message text” >> README.md
  - git add README.md
  - git commit –m “change message on my branch”
- Then, ‘git switch main’ to go back to main

Git Demo – Merge Conflict

- On main, perform the following:
  - echo “# 301 test 2” > README.md
  - echo “message: Main branch text” >> README.md
  - git add README.md
  - git commit –m “Change message on main”
  - git merge <your branch name>
- This triggers conflict between your branch and main:

```
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
```

Git Demo – Merge Conflict

- To view the conflict in terminal, type ‘cat README.md’
  - Git marks the conflict sections with:
    - <<<<< HEAD (version from the current branch)
    - =======
    - >>>>>>> test (version from the branch being merged)
- Open the file and choose the final text
  - Open using ‘nano README.md’ or ‘code README.md’

```
student@cmput301 cat README.md
<<<<<<< HEAD
# 301 Test 2
message: Main branch text
=======
# 301 test 1
message: Test branch message text
>>>>>>> test
```

Git Demo – Merge Conflict

- Delete the conflict markers

  <<<<<<< HEAD

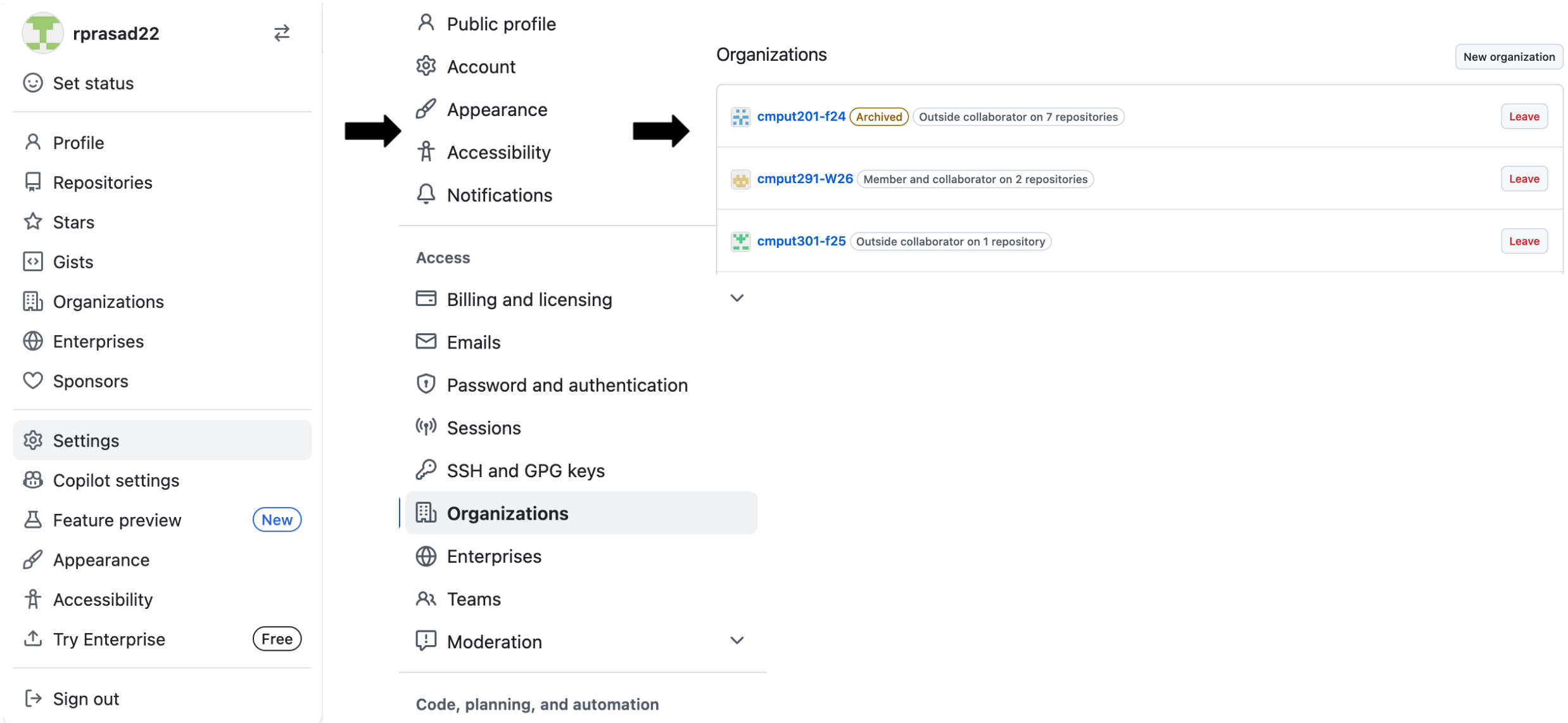
  =======

  >>>>>>> test

```
[student@cmput301 git add README.md
[student@cmput301 git commit –m "resolve issues"
[student@cmput301 git push origin main
```

- The file should contain only the final version you want to keep
- Then, stage using ‘git add README.md’
- Commit using ‘git commit –m “<your commit message>”’
- And push using “git push origin main”
- In general, to resolve a conflict, edit the file, keep the correct final content, delete Git’s conflict markers, then add and commit the file

GitHub Organizations



Making a GitHub Organization

- Choose the Free option, then provide the required information
- Add team members as collaborators
- Do the class participation exercise by creating a new repo in the organization
- For the Group Project, create a **separate repo in the same organization**