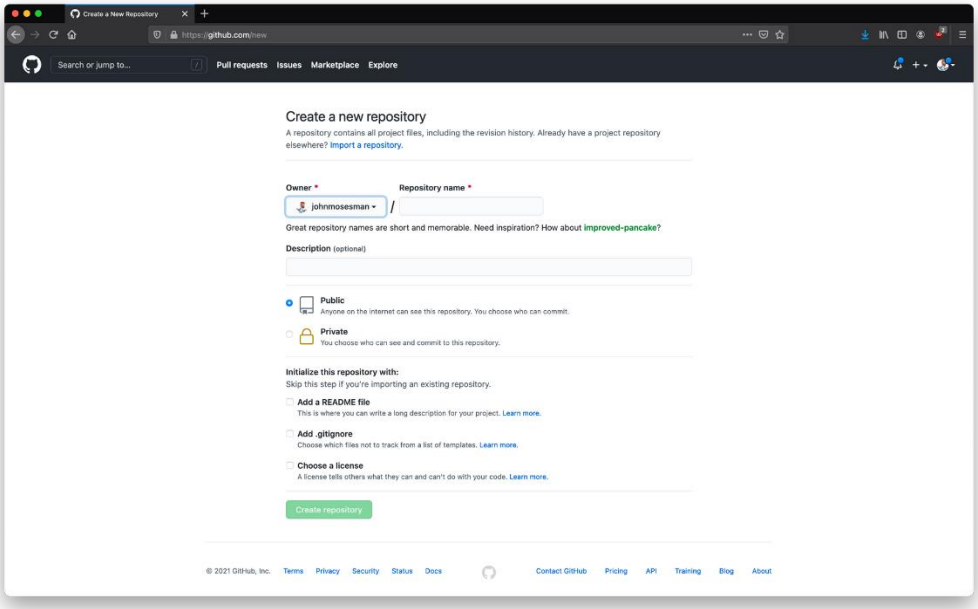
Git And GitHub Workflow

1. Create GitHub Account - https://github.com/
2. Install Git and set up (configure) GitHub account - https://www.javatpoint.com/how-to-install-git-on-windows
    a. **git config –global user.name “name”**
    b. **git config –global user.email “email”**
3. Create a new repository in GitHub –
    a. Click the "New" repository button on GitHub home page
    b. Next, choose a name for the repository and whether we want the repo to be public or private.
    c. We can optionally add a README file if like, and then click "Create repository."



4. Clone a Git repository
    a. Cloning a repo means downloading all of the project's code and metadata from GitHub repo and stored in our working directory.
    b. To clone a repo, we use **git clone <URL>.**

5. Change into the new directory (Repo) and terminal will show directory name: (main) - **cd dirname**
6. Check the status of a Git project - **git status**
    a. What changes have been made and what do we want to do with them?
    b. We're on branch main
    c. Our local main branch is identical to the origin's (GitHub's) main branch
7. Add Files to staging Area
    a. We haven't made any changes to the project yet
    b. Create and edit the file in any text editor
    c. Let's run **git status** again and see the difference in its output:
    d. Here we see a different output than before. We see a section describing "Untracked files," and our new file is listed there.
    e. Before Git will start tracking changes to a file we first have to tell Git to track it — and as the bottom of the message states we can use git add to do that:
        i. **git add -A**
8. Make our first commit
    a. To commit our changes, they must first be added to the staging area by using git add.
    b. Let's check the status again
    c. The message has changed again. It now says that we have some changes that are ready to be "committed."
    d. Next, we need to finalize the commit by using **git commit -m "message"**

e. It's best practice to provide a detailed message of what changes you made—and more importantly—why you are committing these changes.

9. we haven't told GitHub about the newest commit we made. GitHub still thinks that the repo is up to date with what it has seen.
10. Push first commit to GitHub
    a. We have a new commit on our local machine and we need to update our "source of truth"—the origin remote—aka GitHub.
    b. We're currently on the main branch locally, so we need to tell GitHub to update its own main with the new commit that we made.
    c. To do that we use the git push command and we can specify where we want to push and what branch we want to push to.

       git push -u origin main
    d. Here we pushed to the origin remote (GitHub) and to the main branch.

- **git config –global user.name “name”**
- **git config –global user.email “email”**
- **git clone <URL>**
- **git init**
- **git add -A**
- **git status**
- **git commit -m "message"**
- **git push -u origin main**

How to collaborate with others in Git

- So far we've been looking at the simplest use case: working by ourselves on one branch.
- In reality, we'll usually be working with multiple people working on multiple different branches. This is the real power of Git after all: a system to collaborate and track changes over time amongst many collaborators.
- In general, it's best practice to not work directly on the main branch.
- The main branch is supposed to be the "source of truth" for the project—changes to it should be carefully reviewed. Any change in origin/main becomes the new "source of truth" for anyone else working on the project, so we shouldn't just change it without some thought and review by others.
- Instead of working on main directly, let's branch off of main into our own feature branch, and then merge those changes back into main.

Feature branches in Git

- To begin, let's branch off of main and create our own feature branch to work on.
- When you create a branch off of another branch, you create a copy of that branch at that point in time. You can now change this new branch independently of the original branch.
- To do this we use git checkout with the -b flag and name of the new branch **git checkout -b branch_name**
- So we have a new branch, and for now that new branch is identical to main (we haven't made any changes yet).
- Notice that the terminal now shows us on feature the branch. Changes on feature branch will not affect the main branch at all.

- Next let's repeat what we've already done before and create a new file, add file to staging area and commit to local

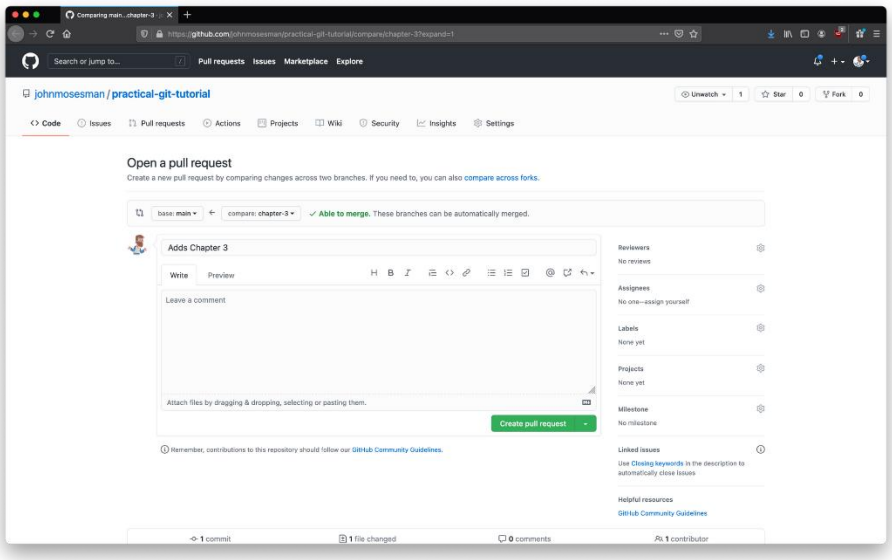
Git workflows for collaboration

let's talk about a couple different workflows

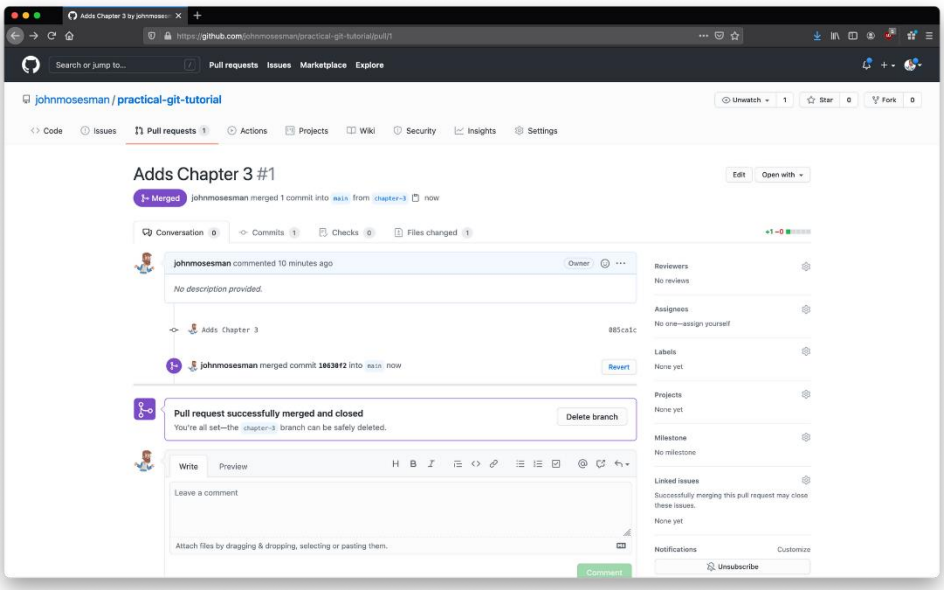
- The first one is the most straightforward:
    - Merge changes from feature branch into our local main branch
    - Push local main branch to origin/main
- The second way is a little more complicated:
    - Push our local feature branch to origin (this creates a new branch on origin called origin/feature)
    - Merge origin/feature into origin/main on GitHub
    - Pull down the new changes from origin/main into our local main
- The first workflow is definitely easier, and it is something I would use if I was working on this project by myself without any other collaborators.
- However, if I had collaborators, I wouldn't want to push directly to the main branch from my local. By doing so I would be changing and taking control of the history of the project solely on my own changes—without any input or review from collaborators.

Workflow -

- The first step is to push our new branch to GitHub. Since this branch doesn't exist yet on GitHub, GitHub will make a new branch for us that is a copy of what we pushed:
  git push -u origin branchName
- Now that we have our branch on GitHub, we can create a pull request to be reviewed by our teammates.



- We'll notice that all the files are shown where we changed.
- After clicking "Create Pull Request" we're taken to the new PR we just made.
- At this point we could assign a reviewer to the PR and have a back-and-forth discussion around the code by leaving comments on specific lines in the diff. After the code has been reviewed and we make any changes that need to be made, we're ready to merge.
- And with that, our pull request has been merged into main!



Bring our local up to date

- We've now made a change to origin/main in a safe, controlled, and peer-reviewed way.
- But our local doesn't know anything about this change. Locally, Git still thinks we're on our feature branch which isn't merged into main:
- We need to pull in the new information from our origin to update our local repository.
- To start, let's switch back to our main branch locally:

  git checkout main
- Our local thinks we're up to date with origin/main because we haven't asked the remote repository (origin) for new information since we downloaded the project at the beginning using git clone.
- After running git pull, if we run git status once again we'll see that everything is up to date.

  git pull origin main
- And with that, we've pulled in changes from our remote and got our local up to date!

Review: how to start a new feature workflow

- I'll end with a quick review of how to approach starting a new task and the commands and flows to do it.
- Say you've been given your first ticket at a new job: a small bug to squash in your team's product.
- The first thing you'd need to do is pull down the repo using git clone <URL>.
- Next, you'd want to make a feature branch off of main using git checkout -b <BRANCH_NAME>. After that, you'd fix the bug and commit the change(s) using git add and git commit.
- Maybe solving this problem takes multiple commits—or maybe you make a few commits in an attempt to solve it before you finally arrive at the solution. That's ok too.
- After committing, you push your new branch to the origin (git push origin <BRANCH_NAME>) and create a pull request. After a code review your branch is merged in (yay!).
- You've now completed your feature, and it's time to switch back to main (using git checkout main), use git pull to get your latest changes plus any other changes other people have made, and start off again with a new branch.