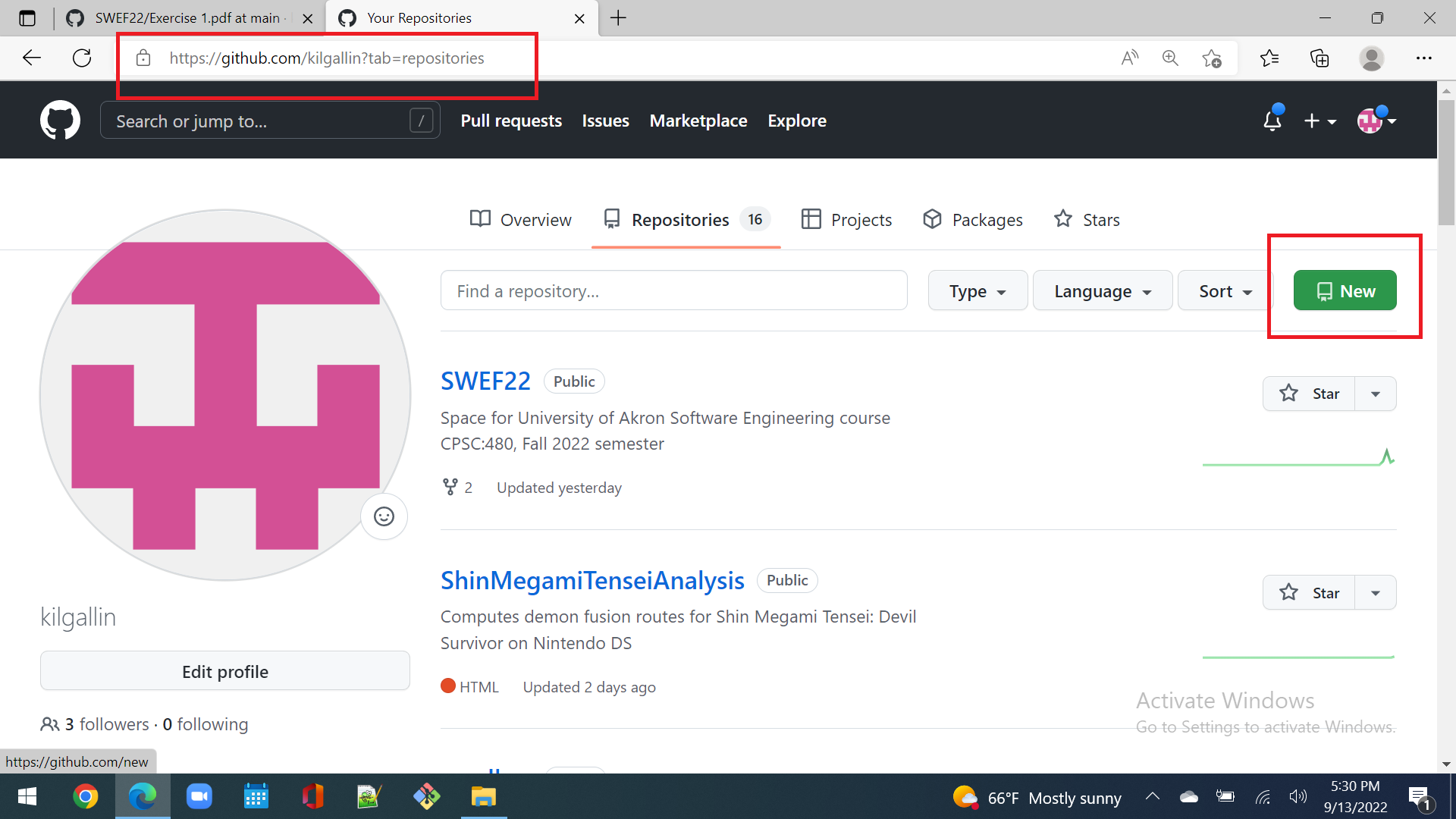
CPSC:480 Software Engineering Exercise 1

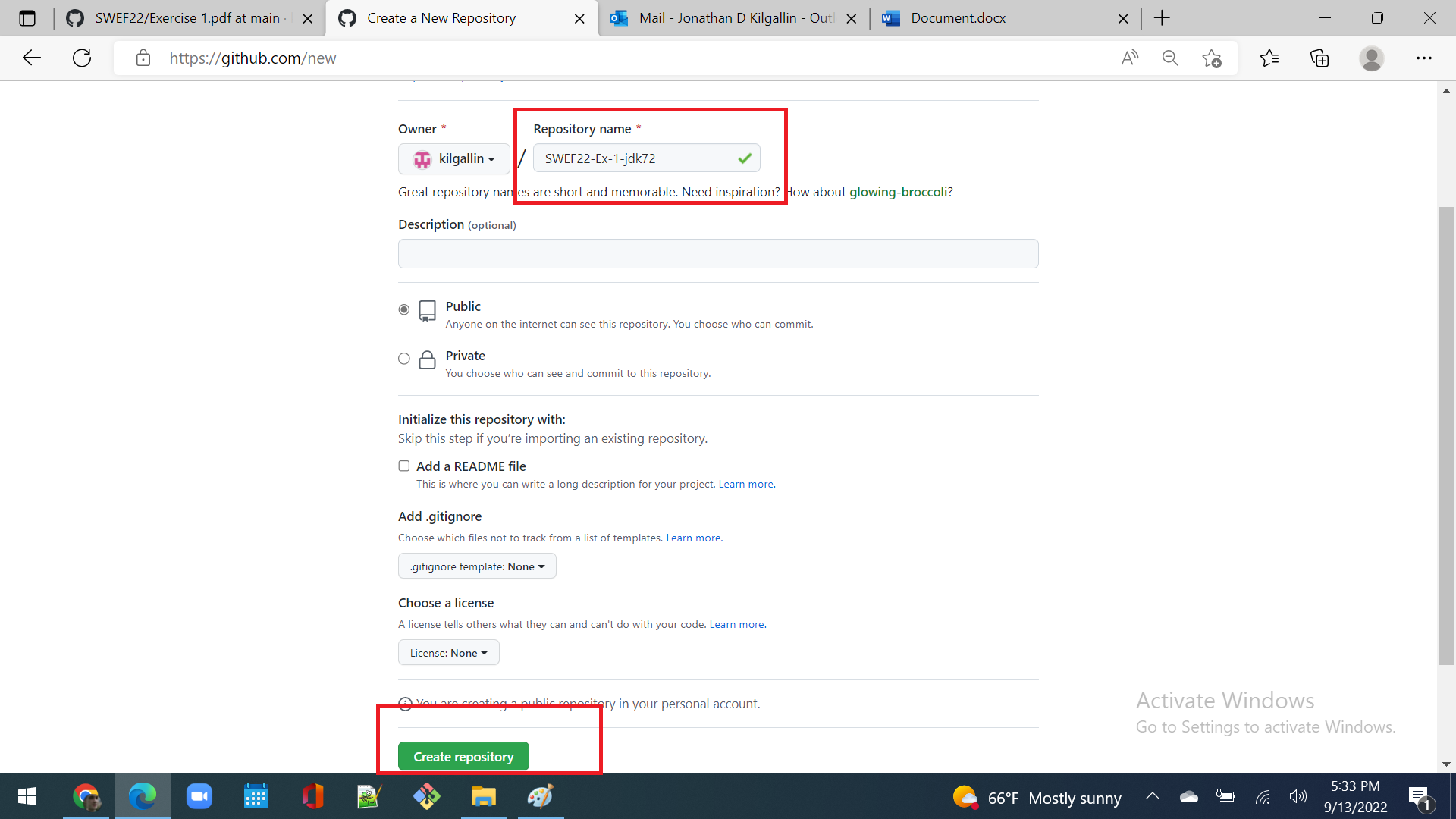
Complete this exercise in pairs with the student next to you. *Steps in Italics are completed on GitHub, others on local machine using git.*

*Note: In this walkthrough, GitHub user “kilgallin” UAid jsk72, is the primary user completing these steps, and GitHub user “JDKeyfactor” repo “SWEF22-Ex-1-otherJDK" is simulating the partner.*

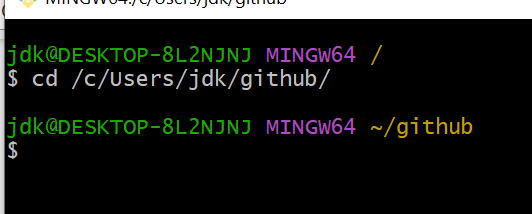
Part I – Initialization

1. Open laptop with Git installed and create parent directory for projects (e.g. C:\Users\me\github).
2. *Create a public GitHub repository named "SWEF22-Ex-1-<UAid>".*

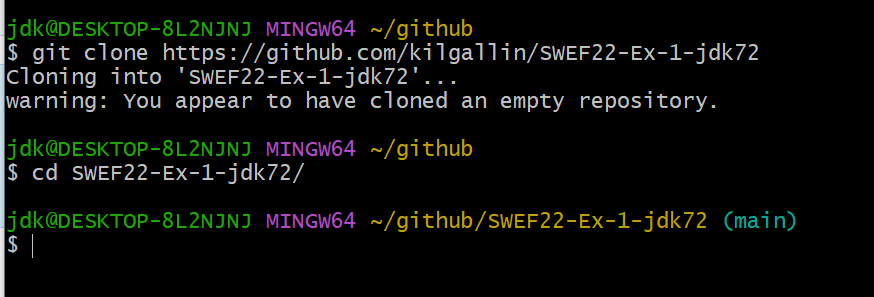




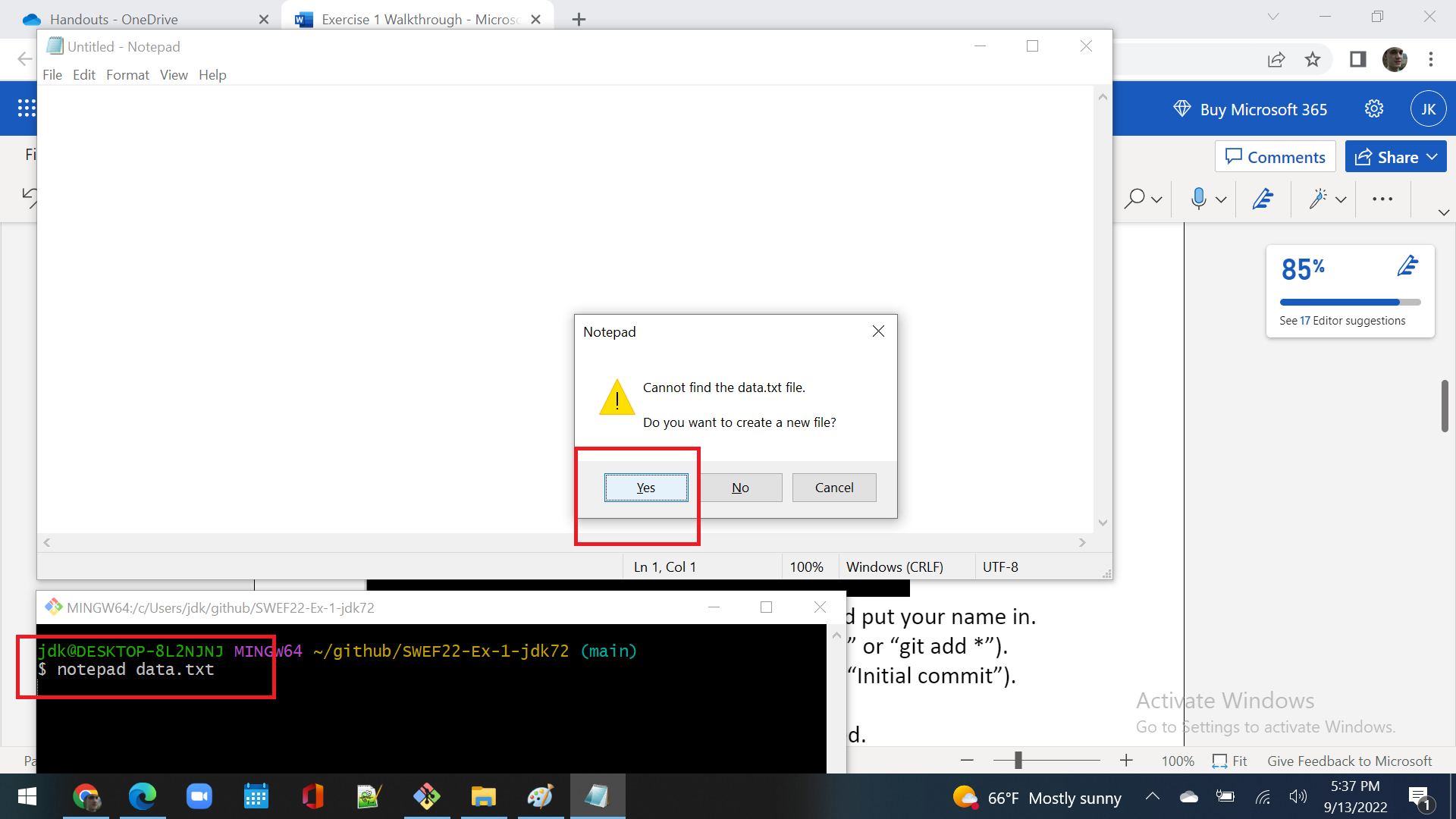
1. Open git command line and go to project parent directory.

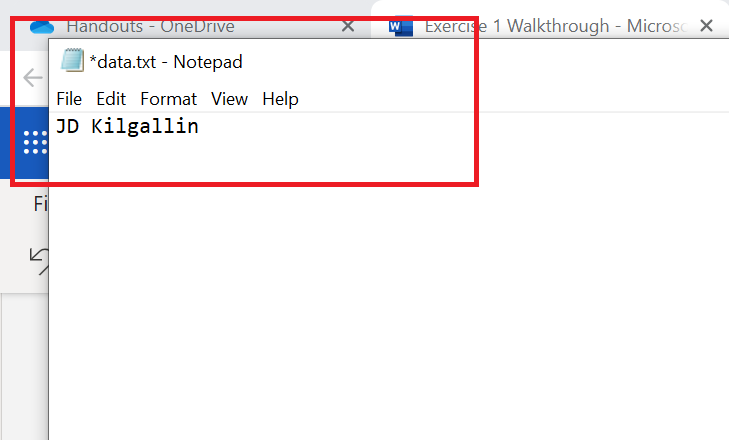


1. Clone a local copy (“git clone https://github.com/<user>/SWEF22-Ex-1-<UAid>”) and make sure you’re in the repository directory.



1. Create file "data.txt" in local repo folder and put your name in.

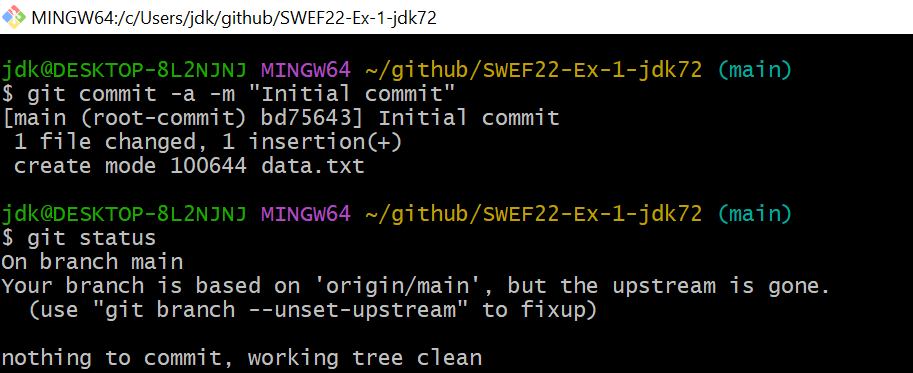




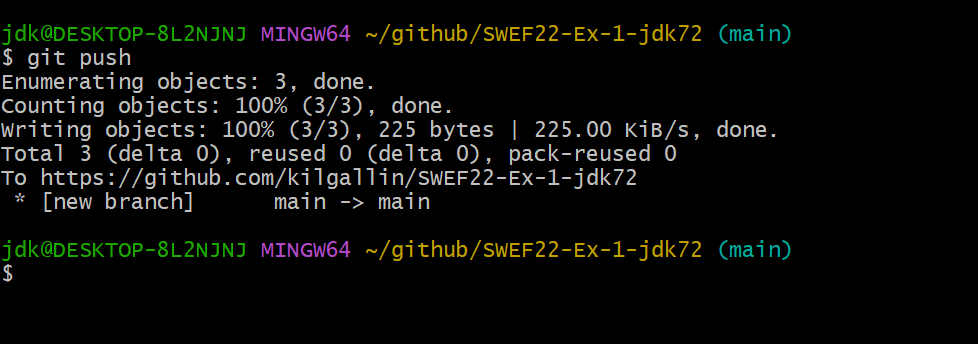
1. Add to files tracked by Git (“git add data.txt” or “git add \*”).



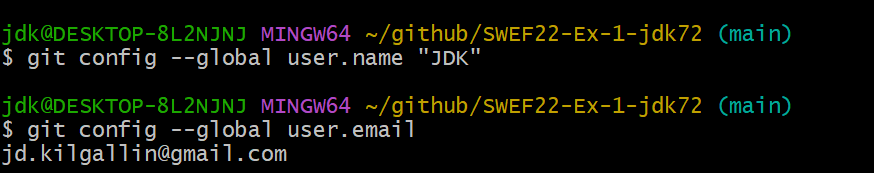
1. Create an initial commit (“git commit -a -m “Initial commit”).



1. Push change to GitHub (“git push”).

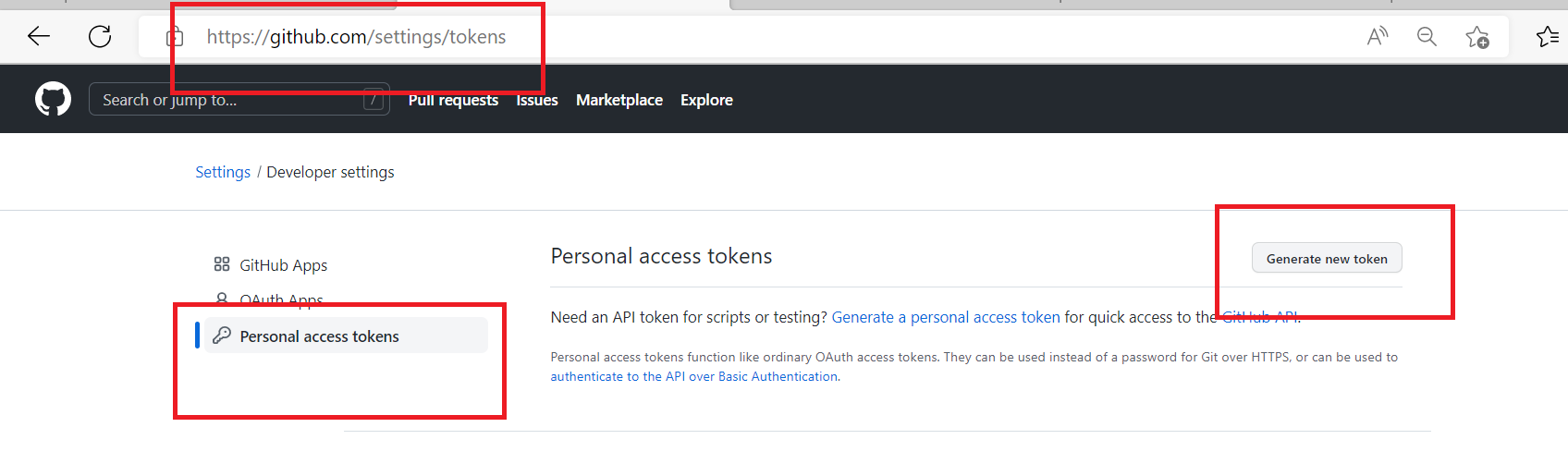


1. Set author and GitHub credentials as needed.

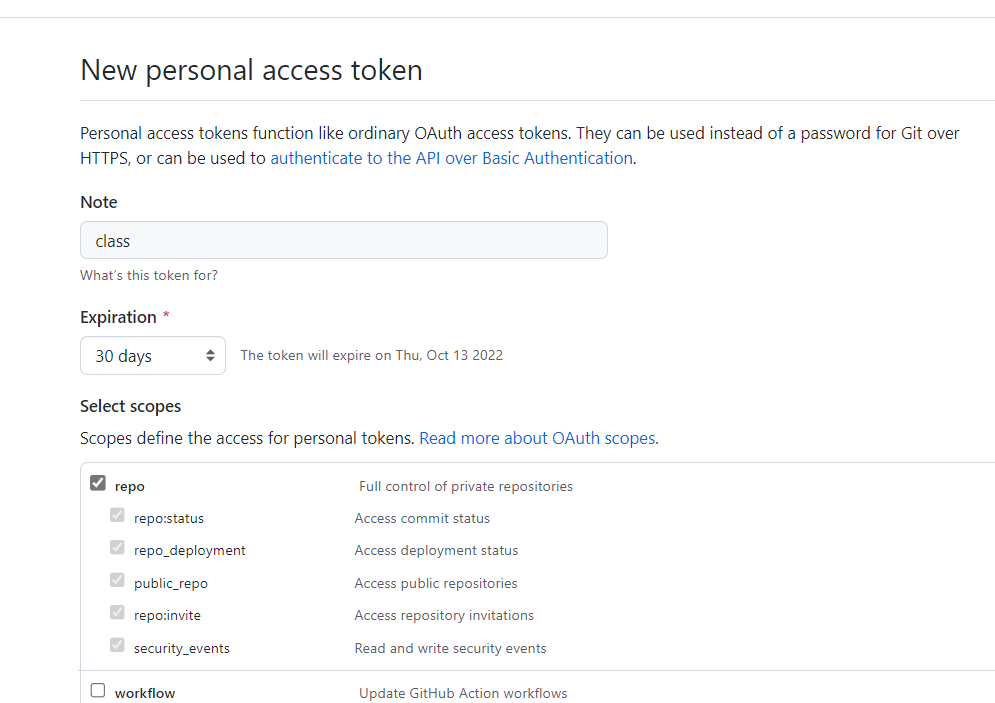


*Note: If you get an error about passwords not being supported, as seems to be the case on Apple OS X, do the following:*

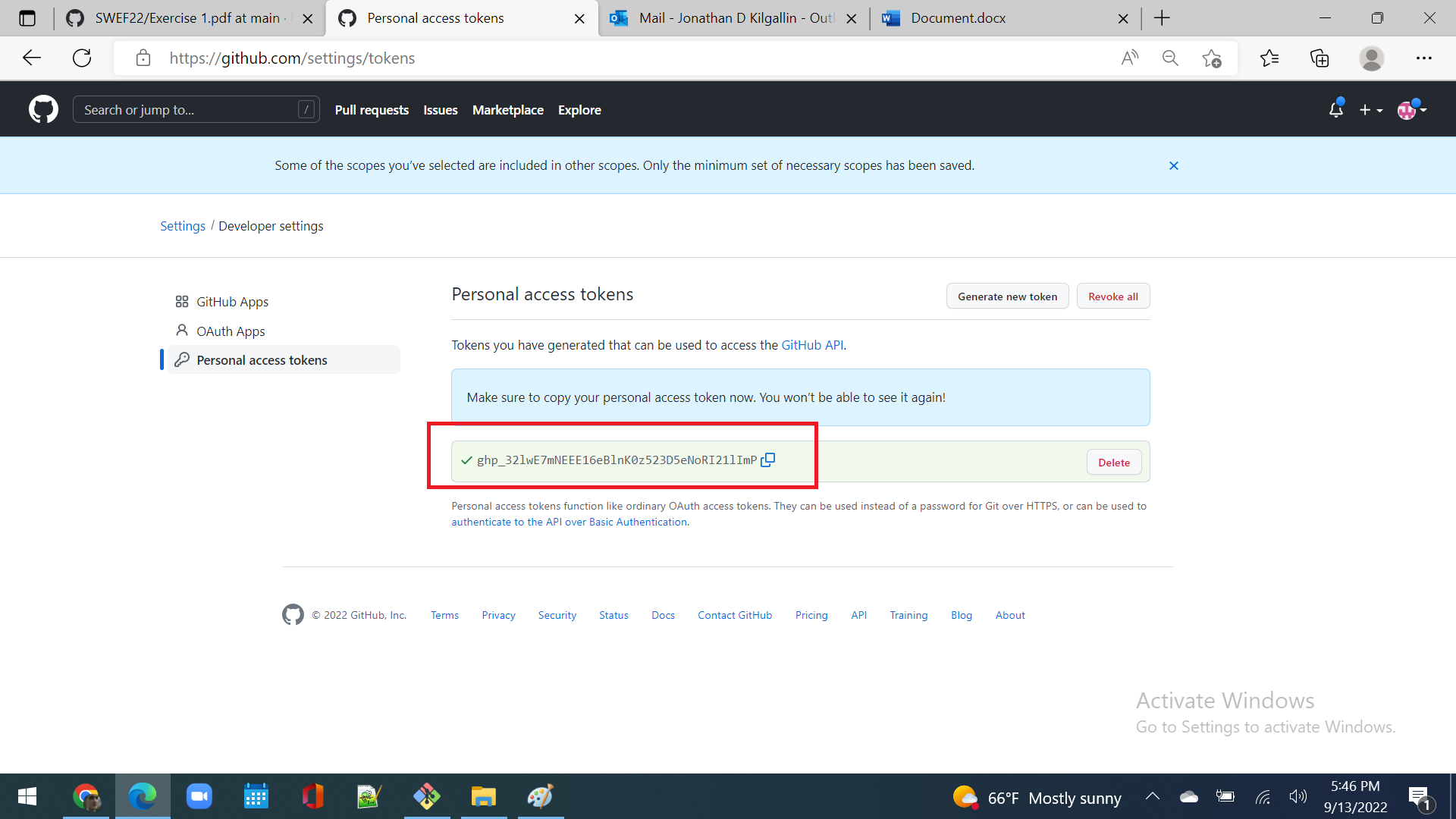
*Click your profile in the top-right of the GitHub page and go to “Settings”. Find “Developer Settings” in the far bottom left. Select “Personal Access Tokens” and click “Generate new token”*



*Enter a name (e.g. “class”) and check the box next to repo, then click “Generate Token” at the bottom.*

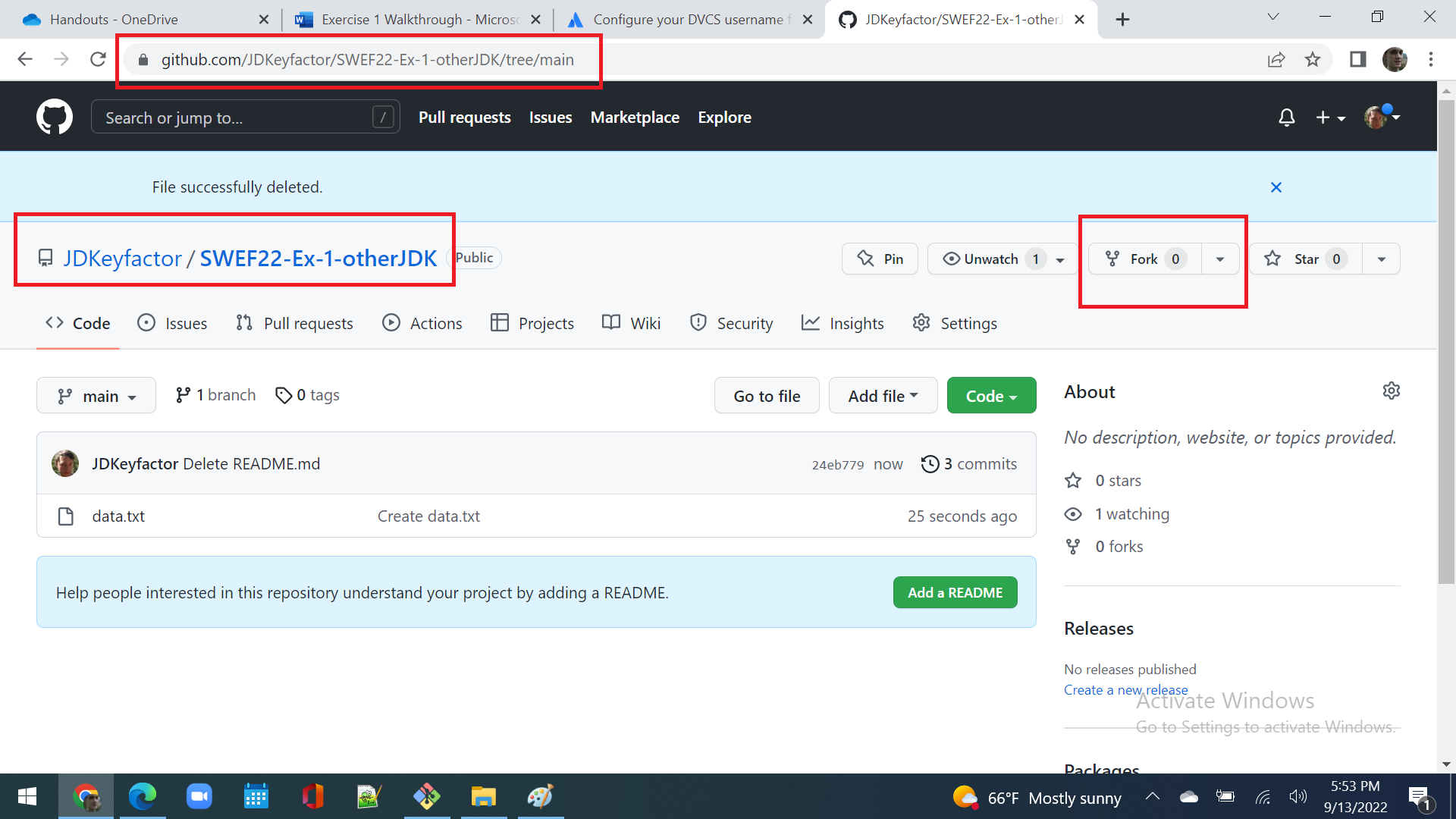


*Copy the value shown and enter it as your password in Git*

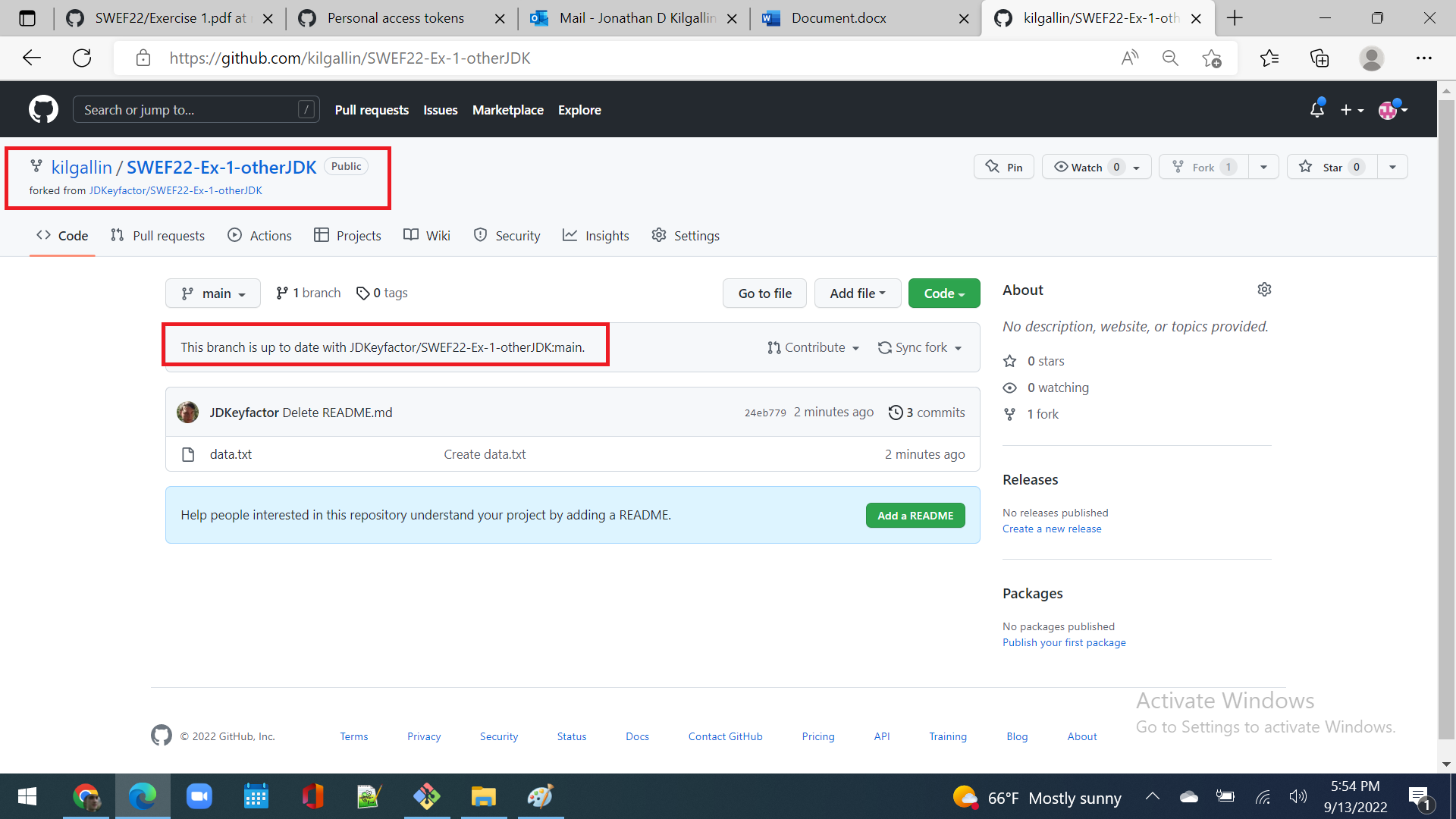


Part II – Collaboration

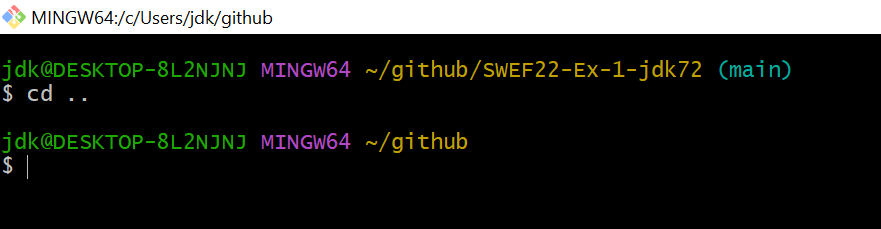
1. *Open a browser to https://github.com/<partner-username> /SWEF22-Ex-1-<partner-UAid> while logged in to your account.*



1. *Click “Fork” in top right, accept defaults, and click “Create fork”.*

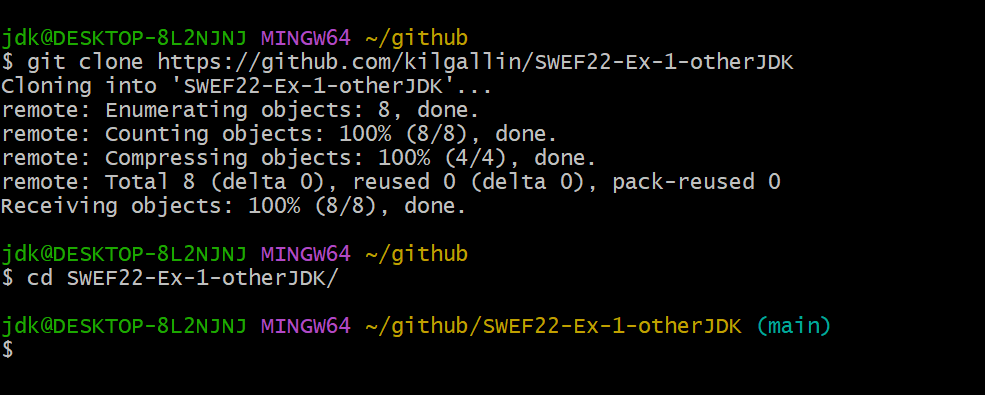


1. On your machine, go back to github directory, outside of repo

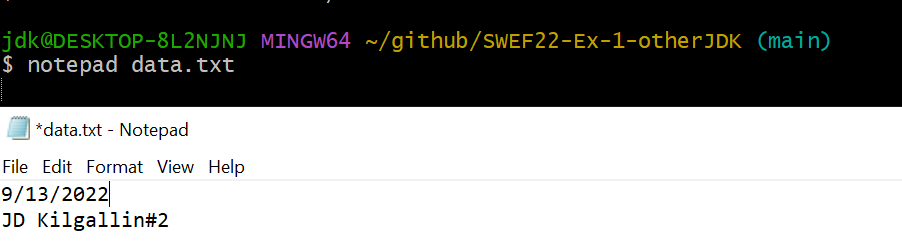


1. Create local copy of fork (“git clone https://github.com/<**your**-username>/SWEF22-Ex-1-<**partner**-UAid>”) and go in new repo.

*Note: Make sure you clone* ***your fork****, and don’t clone your partner’s repo directly, as you won’t be able to push.*

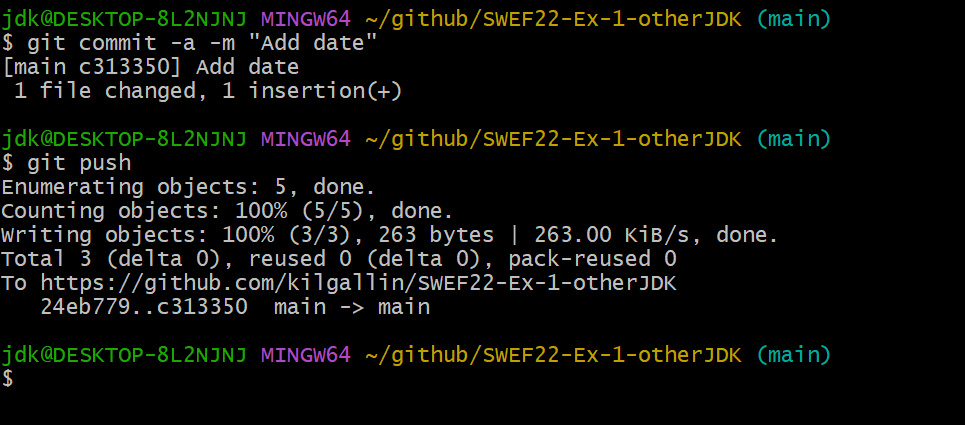


1. Edit data.txt and add today’s date **above** partner’s name.

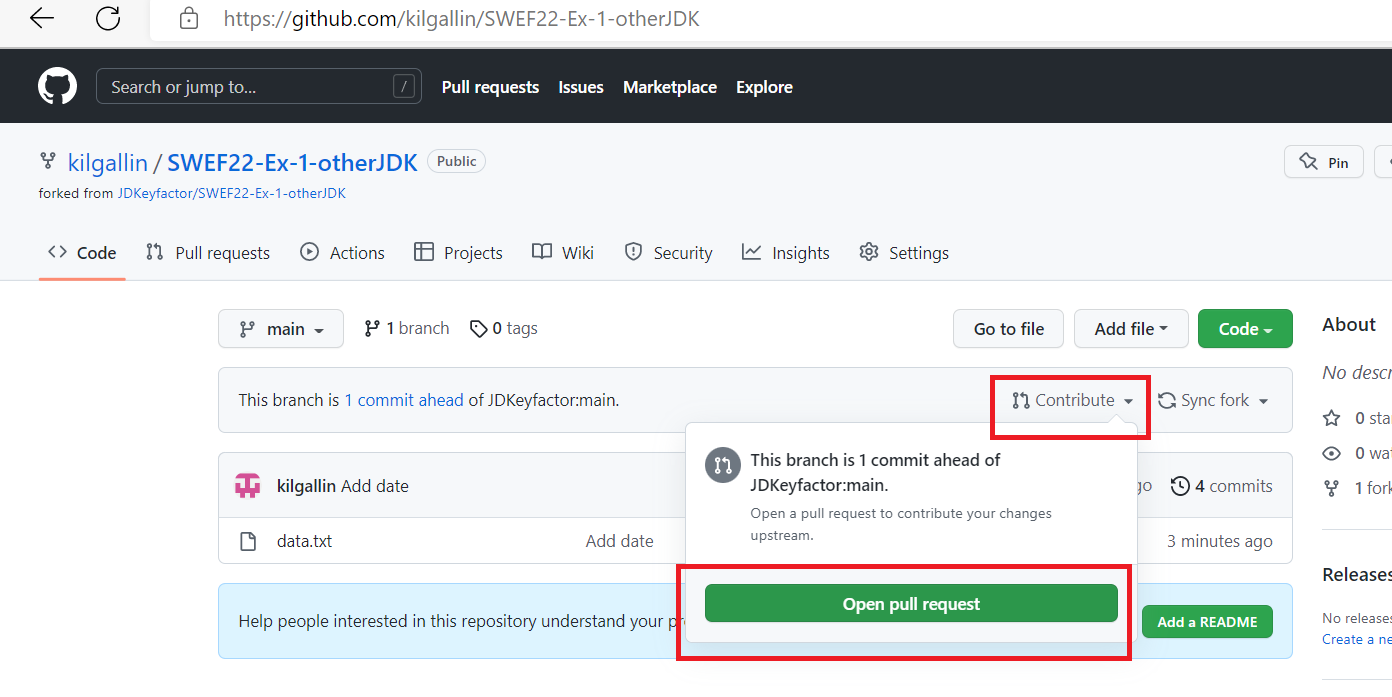


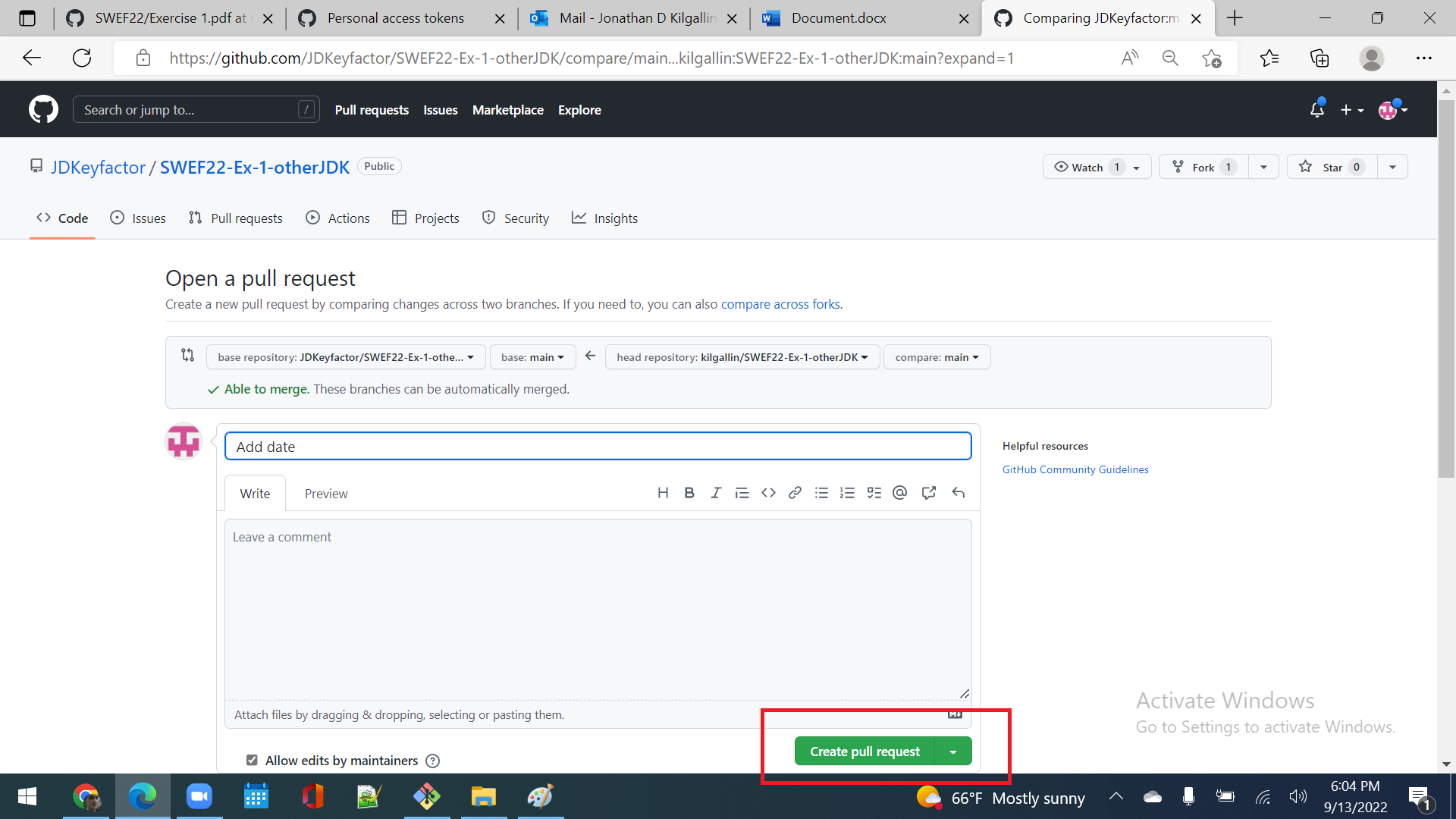
1. Commit and push to your fork on GitHub.

*Note: If you get error 403, check that you cloned* ***your fork***

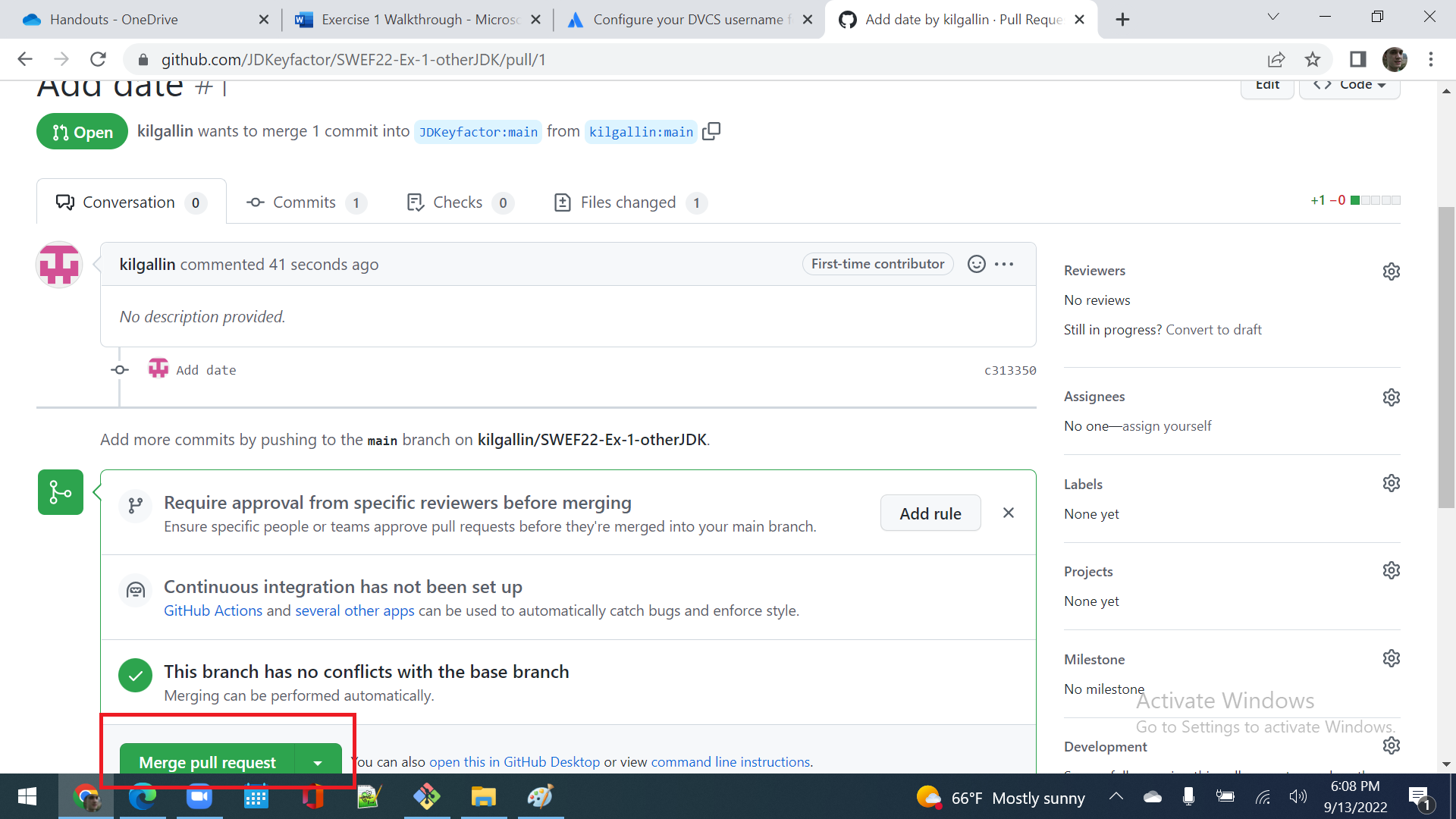


1. *Open your fork in a browser and click on “Open pull request” under “Contribute”, then “Create pull request”.*



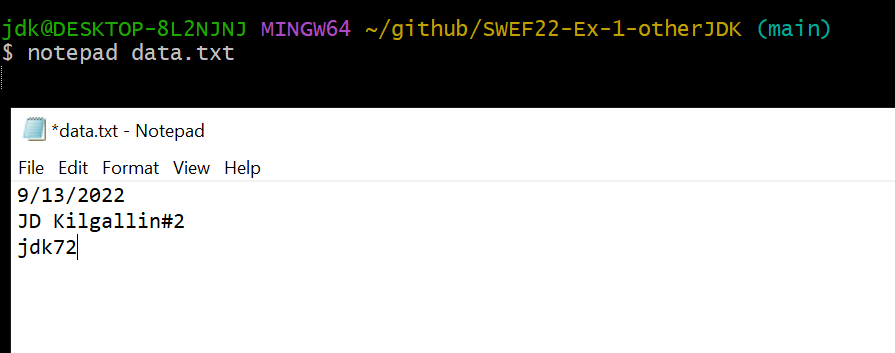


1. *Wait for your partner to complete step 7 before starting step 9.*
2. *Go to your own repo, click “Pull requests” tab, find and open your partner’s PR, and click “Merge pull request”, then “Confirm”.*

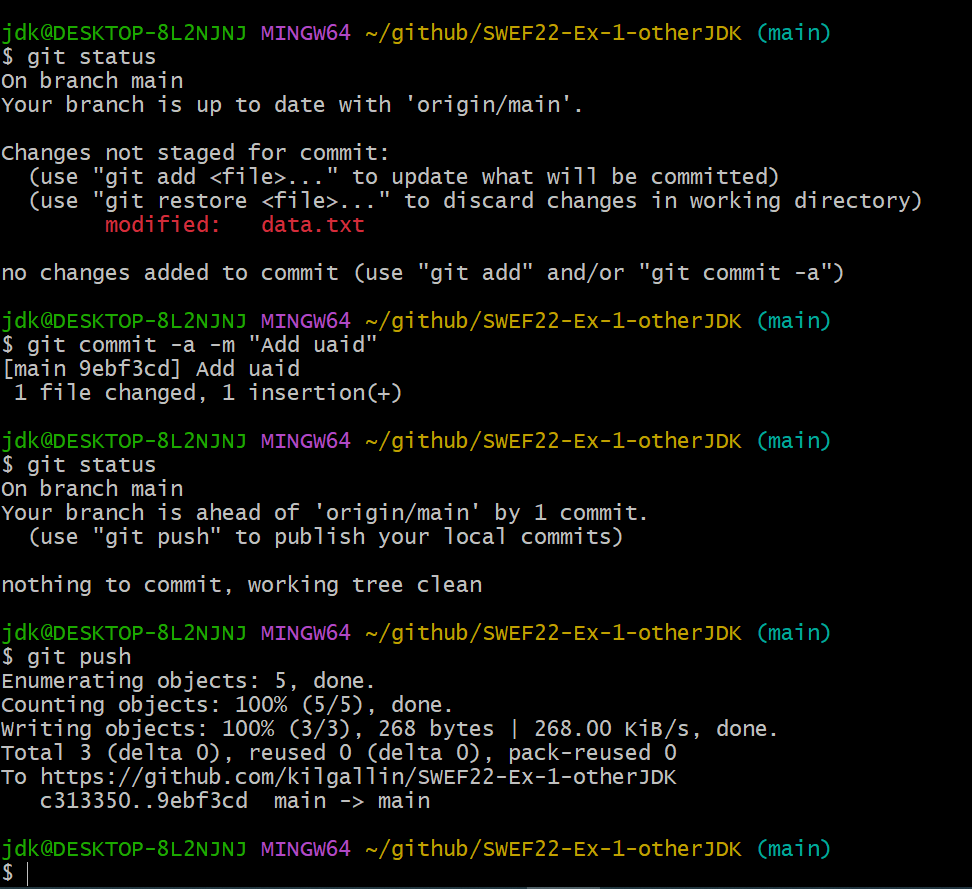


Part III – Conflicts

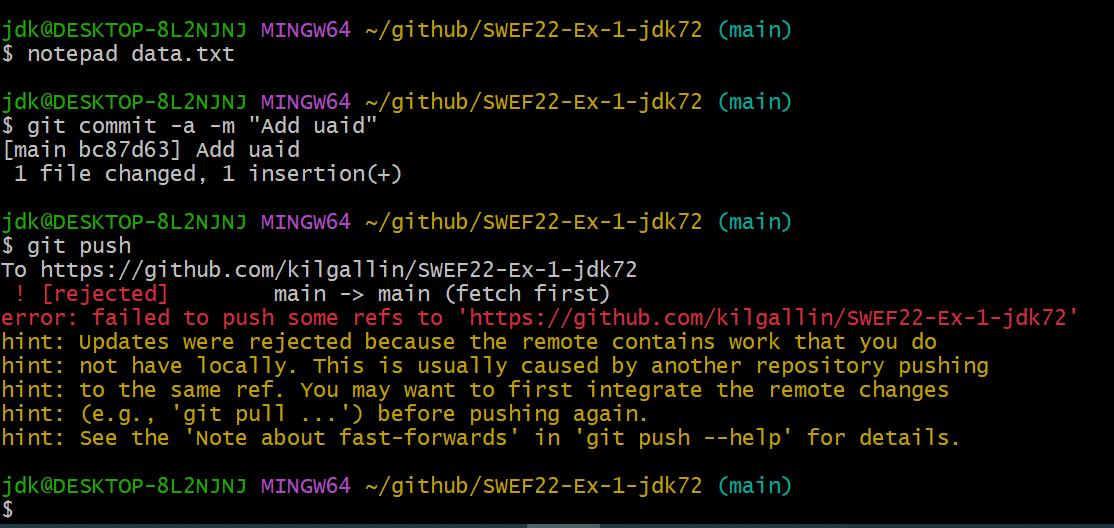
1. On your local machine, “git pull” **both** repos to current state.
2. Edit data.txt in both repos’ copies and add your UAid at bottom.



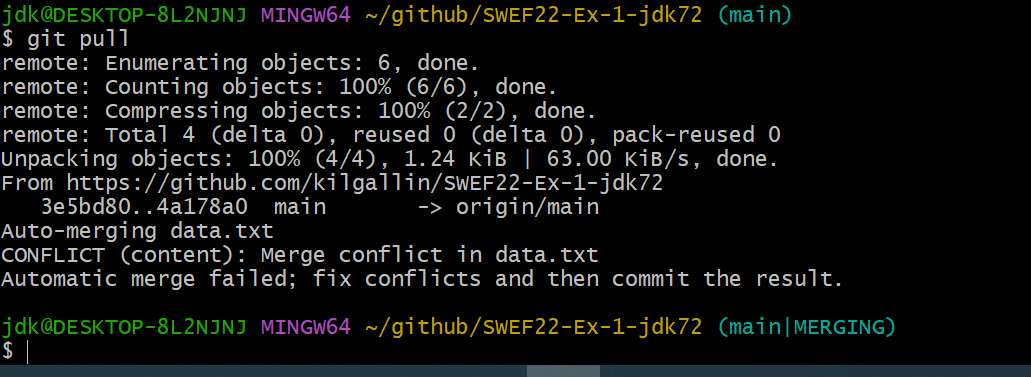
1. Commit & push the change **only to your fork of *partner’s* repo**.



1. *Issue another PR to your partner & accept the one they issue you.*
2. *Wait for your partner to complete step 4 before starting step 6.*
3. Go to **your** repo, commit your local change and try to push.

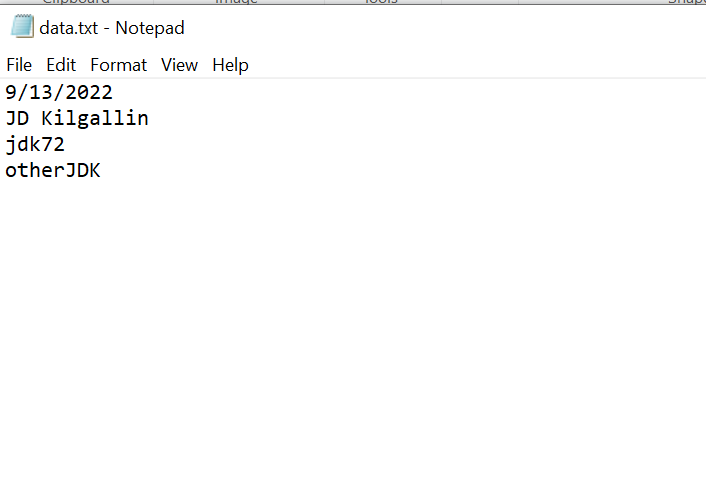


1. Note the failure, then git pull to download the commits on GitHub that come from your partner’s pull request.

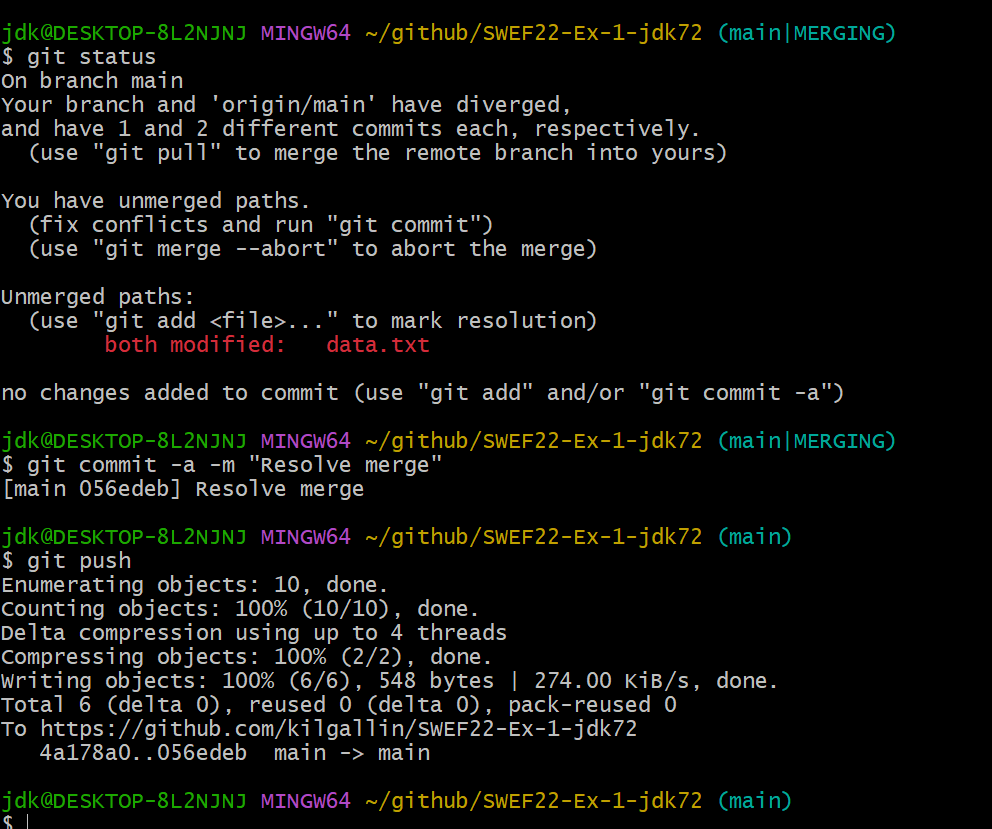


1. Open data.txt and view the conflict to be resolved. Edit the file to contain your UAid, then your partner’s.



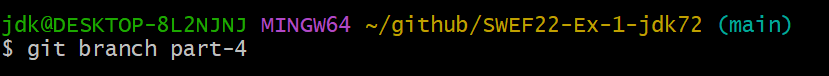


1. Commit the merge and push to GitHub.

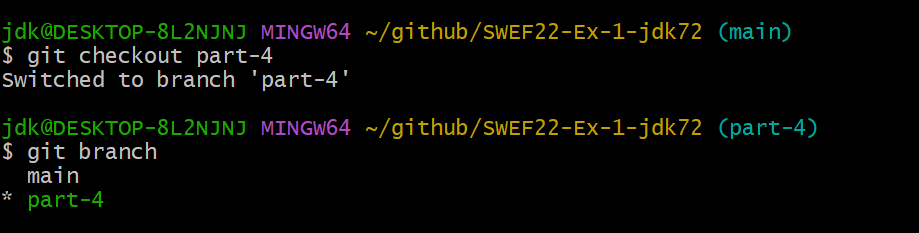


Part IV – Branching

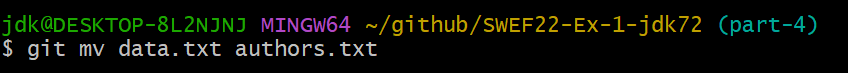
1. Create branch in your repo named "part-4” (“git branch part-4).



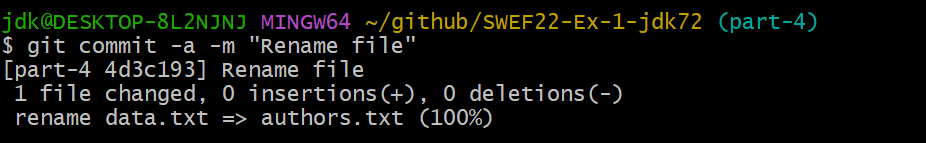
1. Set the new branch as the active branch (“git checkout part-4").



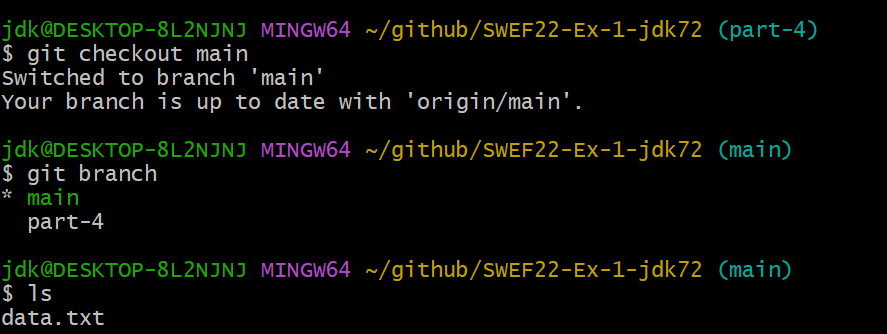
1. Rename data.txt to authors.txt (“git mv data.txt authors.txt”).



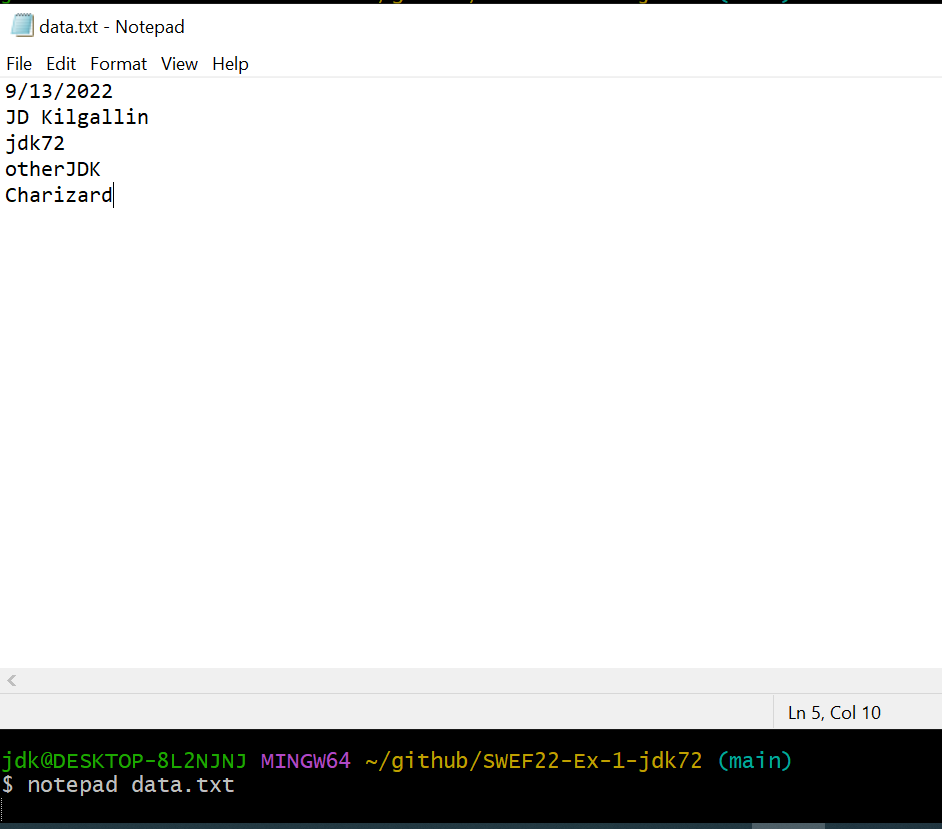
1. Commit the change to this branch (**do not push**).



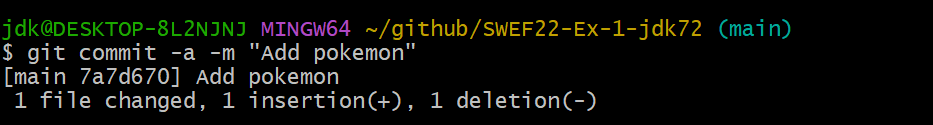
1. Switch to main branch (“git checkout main”) and note that the file is again called data.txt.



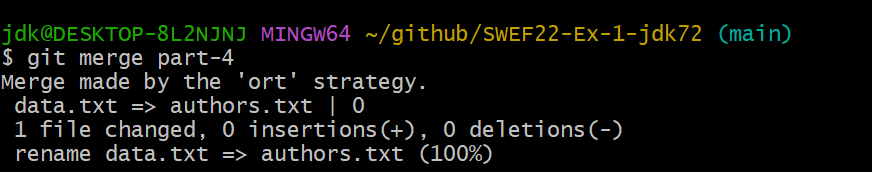
1. Add your favorite Pokémon (eg “Pikachu”) at the end of data.txt.



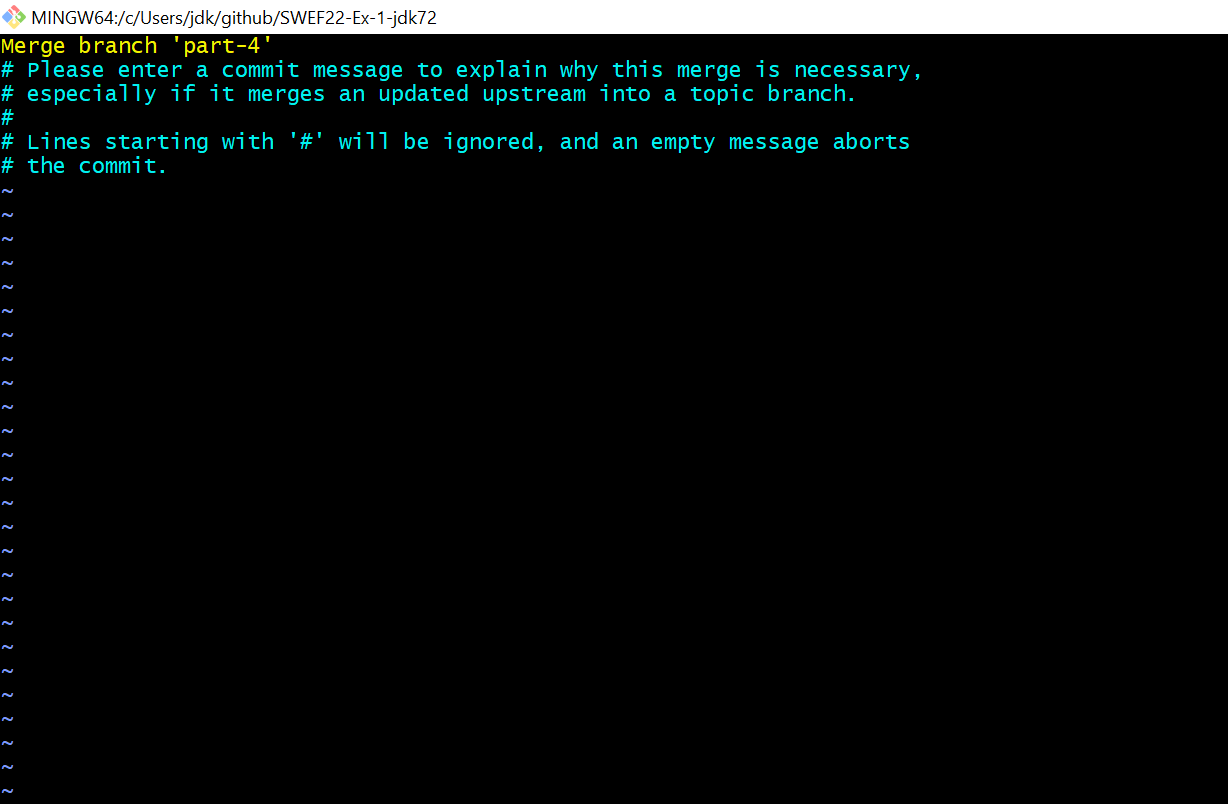
1. Commit the change (**do not push**).



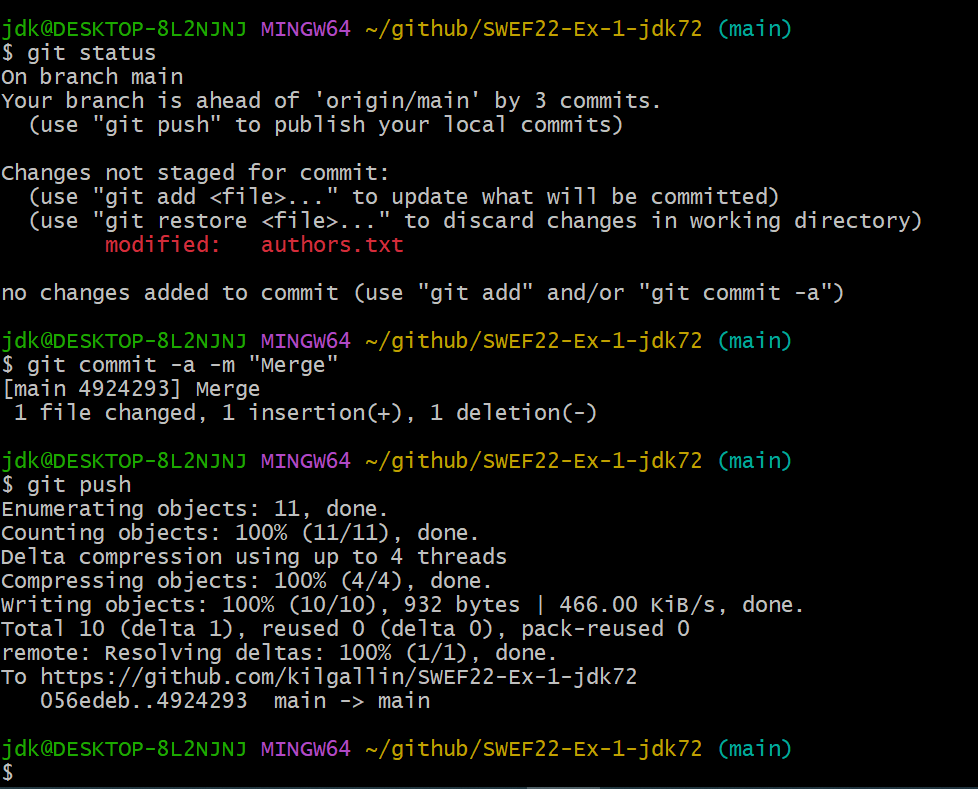
1. Merge part-4 branch (“git merge part-4").

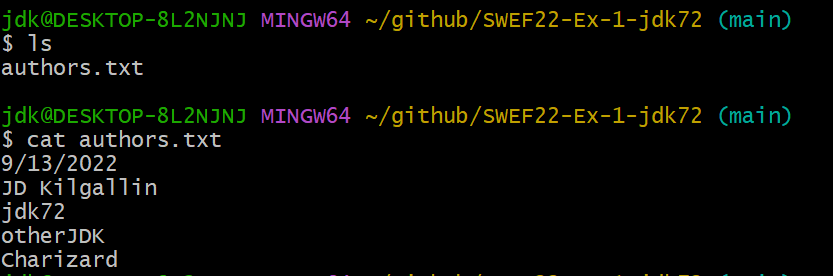


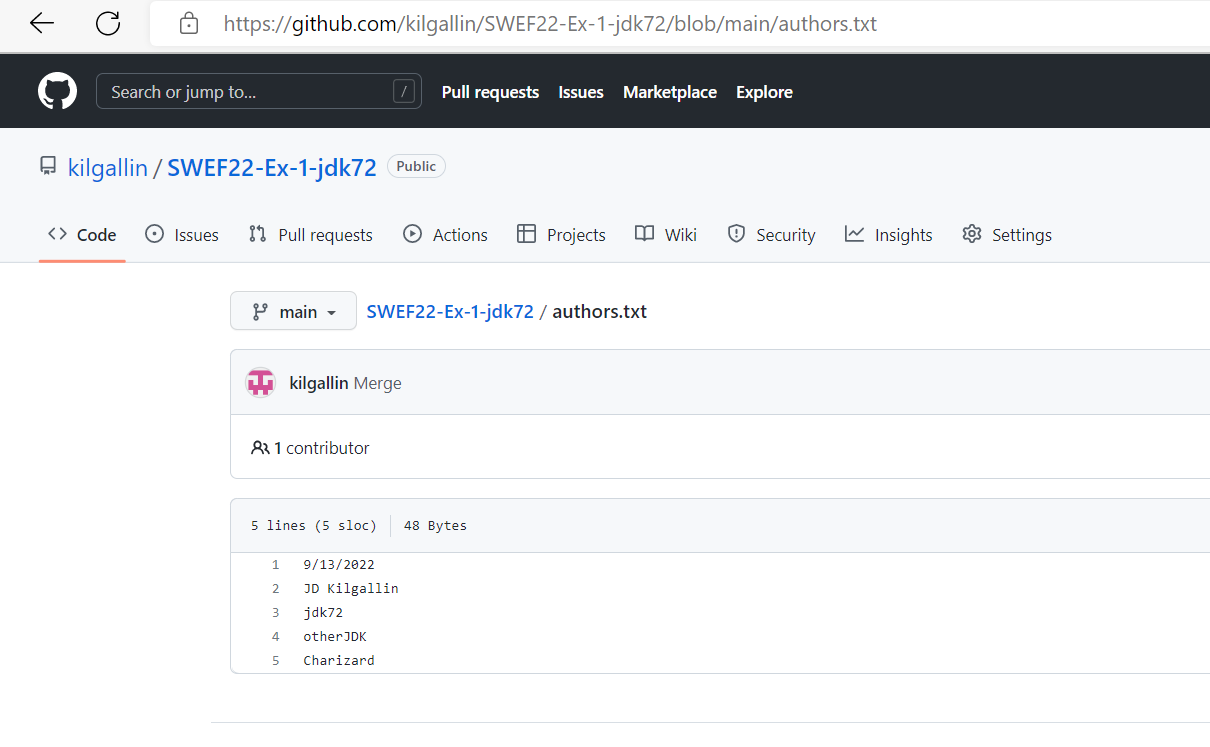
*Note: Git will open its default text editor prompting you to enter a commit message for the merge. The default is likely vim, a UNIX text editor that’s popular but has a learning curve. You can accept the auto-generated commit message and close the editor by typing “:wq” and pressing enter. This should show at the bottom of the screen as you type You can read this as “:” = run command, “wq” = Write+Quit, and enter to execute. If this shows up in the editor instead, “-- INSERT --” is probably at the bottom and you need to hit escape to switch to command mode, then type “:wq”.*



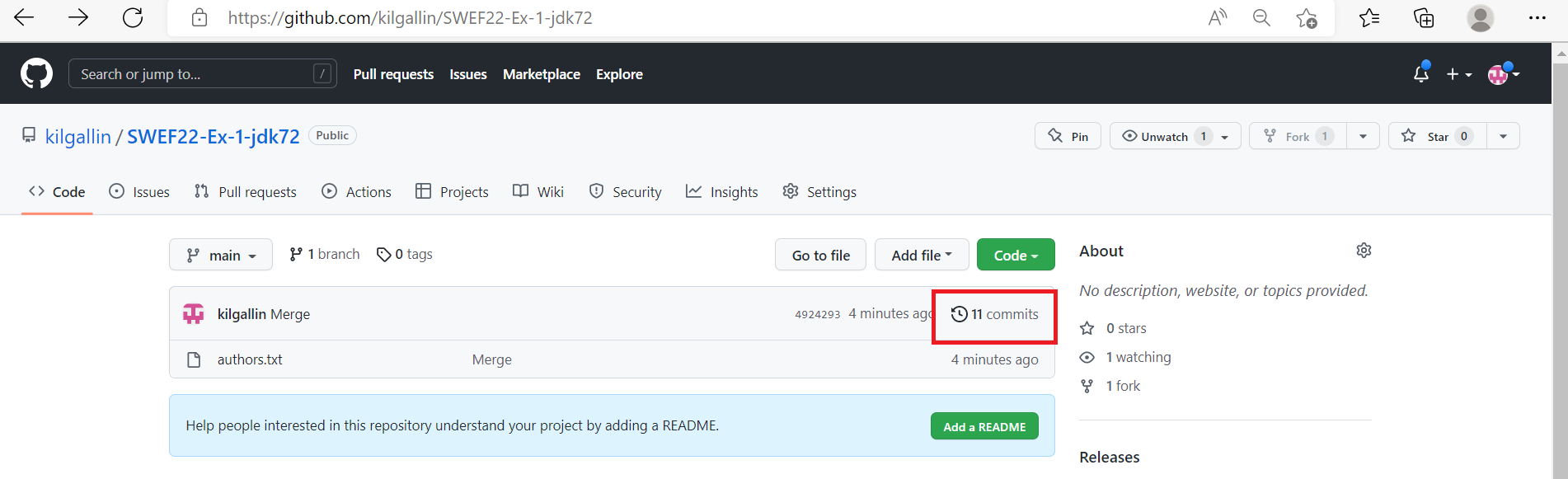
1. Commit and push the change. Note that the file is called authors.txt *and* has the Pokémon you added while the file was called data.txt.

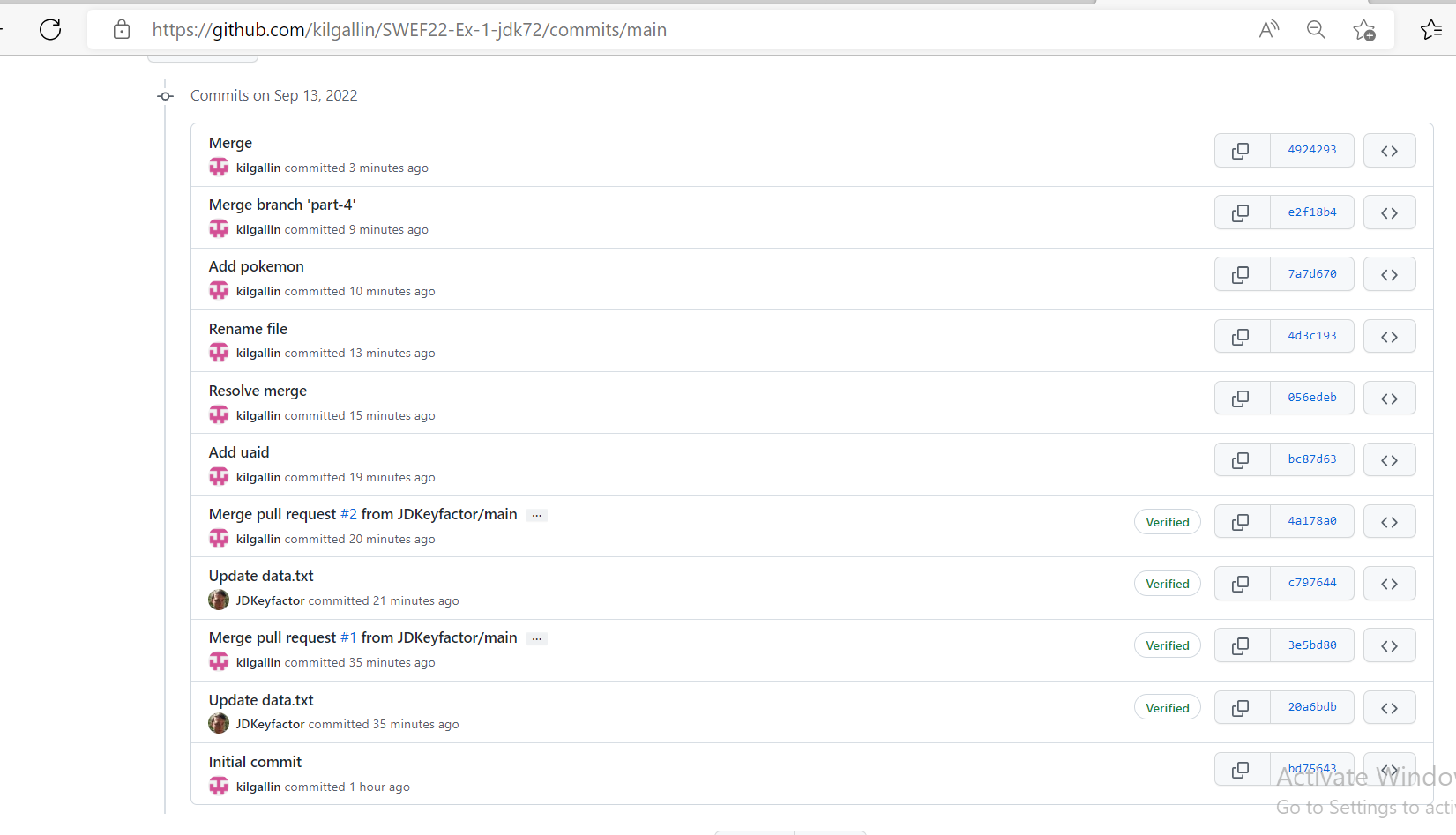






After this step, your repo should have the file “authors.txt” (and nothing else except possibly a README.md if you created it at the beginning) with content like the above. If you go to “Commits” you should see a history similar to the following.





If you go to “Insights->Network”, you should see something like this. It may vary slightly depending on how you and your partner committed changes. But it should show three merges: two from pull requests and one from a branch merge.

