Activity 09

Basic operations with Git

CSC 2310

In this lab you will perform basic operations using git. You will use the files in the repository to perform these operations.

Pre-work

Download the lab source files using the following command:

```
git clone https://gitlab.csc.tntech.edu/%userid%/act09_git.git
```

replacing `%userid%` with your own TNTech issued userid.

Concept

You are working on a project with many files. There are also many developers working on the project. Your goal is to maintain a productive and collaborative environment where multiple developers can submit their changes without affecting each other. In this exercise, you will clone a remote repository, create a branch, and perform other basic git operations.

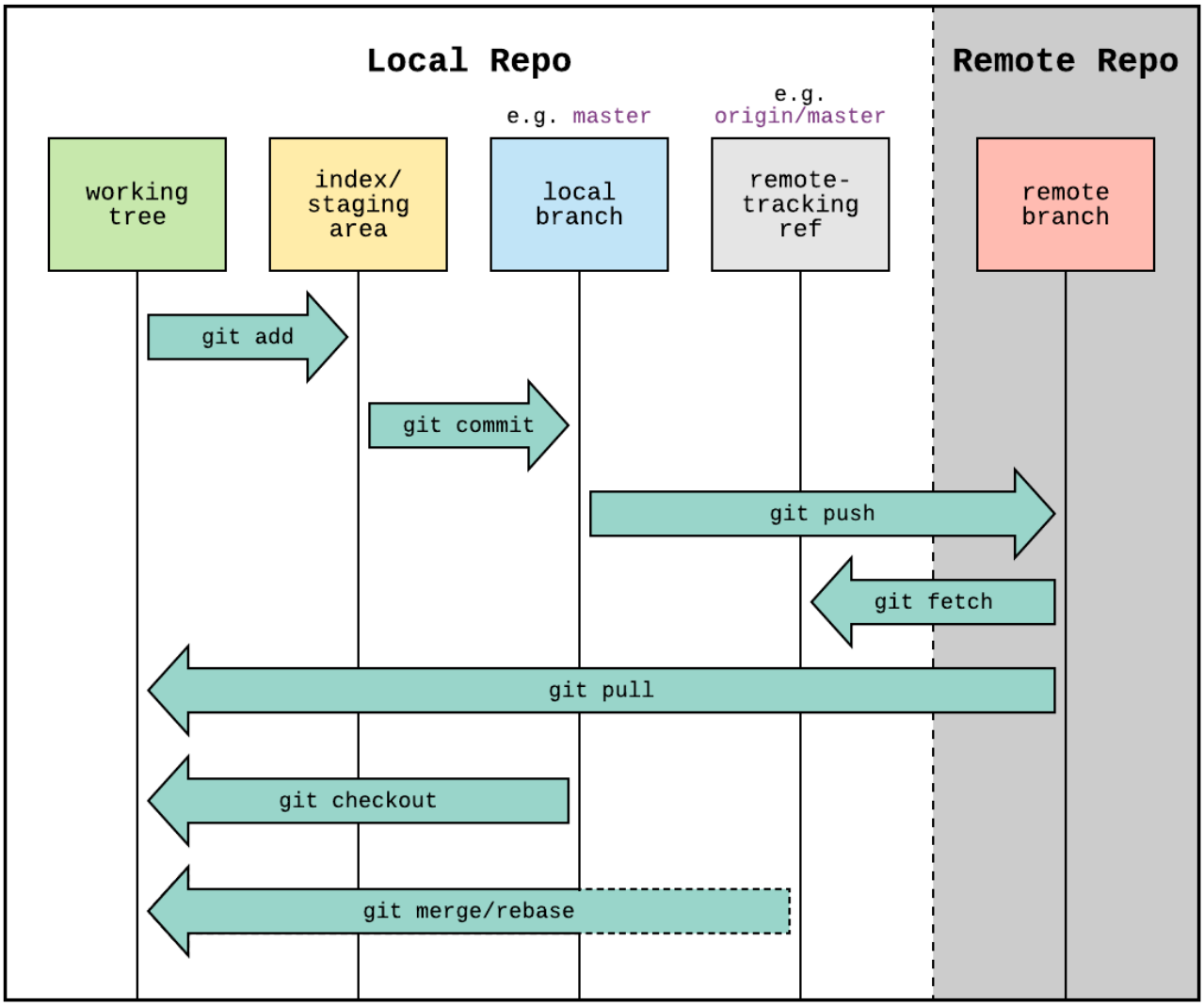
Activity

In this activity, you will be guided through a series of exercises to familiarize you with Git. You will explore the following commands, some of which you have used before in previous labs:

command	description
git add	Adds changes in the working directory to the staging area, preparing them to be committed.
git commit	Records changes made to the repository, creating a new commit with a unique identifier and commit message.
git push	Uploads local repository commits to a remote repository, updating the remote branch with local changes.
git status	Displays the current status of the working directory and staging area, showing any changes or untracked files.
git branch	Creates, lists, or deletes branches within a repository.
git checkout	Switches between different branches or restores files from the repository to the working directory.
git merge	Combines changes from one branch into another, integrating the changes and creating a new merge commit if necessary.

command	description
git diff	displays the differences between the current state of the working directory and the index.

For further exploration, see the git reference guide.



Step 1: Creating and navigating branches

The first step for this activity is to create a series of branches to complete your work on; it's typically discouraged to complete your work on your main branch, so we will practice creating new branches and switching between them. The following command is used to create a new branch where <branch_name> is replaced with the name of your new branch:

```
git branch <branch_name>
```

And to switch branches, you use the command

```
git checkout <branch_name>
```

Alternatively, you can create a new branch and switch to it at the same time with the checkout command using the `-b` flag.

```
git checkout -b <branch_name>
```

This is shorthand for:

```
git branch <branch_name>
git checkout <branch_name>
```

Use the above commands to create three new branches:

- <user_id>-branch-for-merge
- <user_id>-temp-branch
- <user_id>-personal_branch

You can run the command:

```
git branch
```

to see all branches where your working branch will be indicated with an * and highlighted.

```
[calliestewart@Callies-Air-5 GitActivity % git branch
  castewart44-branch-for-merge
  castewart44-personal-branch
  castewart44-temp-branch
* main
```

Step 2: editing files and tracking changes

The next is to making changes on each branch to track the changes.

Switch to you branch-for-merge branch. Edit the Assignment.txt file, and fill in your information for name, date, and branch. Additionally, you should add a new file to the directory by downloading the artwork for *The Lament for Icarus* from The Tate Modern Art Gallery. If you have trouble downloading the file, you can create text file titled Icarus.txt instead.

Next add and commit the changes with the following commands (Note: Icaus.jpg should be whatever you named your new file):

```
git add Assignment.txt
git add Icarus.jpg
git commit -m "changed Assignment, added Icarus file"
```

Now push to your branch with the command:

```
git push --set-upstream origin <your-branch-name>
```

Now, switch to your temp branch where you will edit the Assignment.txt file again. Add, commit, and push this change using the appropriate branch name and the commit message "changed Assignment". Do not forget the `--set-upstream origin <your-branch-name>`.

Finally, switch to your personal branch and edit Assignment.txt. Additionally, add a new file by downloading *The Titan's Goblet* from "The Met". If you cannot download the image, create a text file titled Goblet.txt instead. Add, commit, and push this change using the appropriate branch name and the commit message "changed Assignment, added Goblet file".

Step 3: merging and managing conflicts

In this step you will merge your temp branch with your branch for merge. Switch to your branch for merge and verify with the `git branch` command. Then use the following command to merge your temp branch into your branch for merge:

```
git merge <user_id>-temp-branch
```

You will see that your merge failed. Run the following command to see the details of the conflict:

```
git diff
```

You should see an output similar to the following:

```
++<<<<<<< HEAD
 +name: cal
 +date: 2024
 +branch: branch-for-merge
 +
++=======
+ name: cal
+ date: 2024
+ branch: temp-branch
++>>>>>>> temp-branch
```

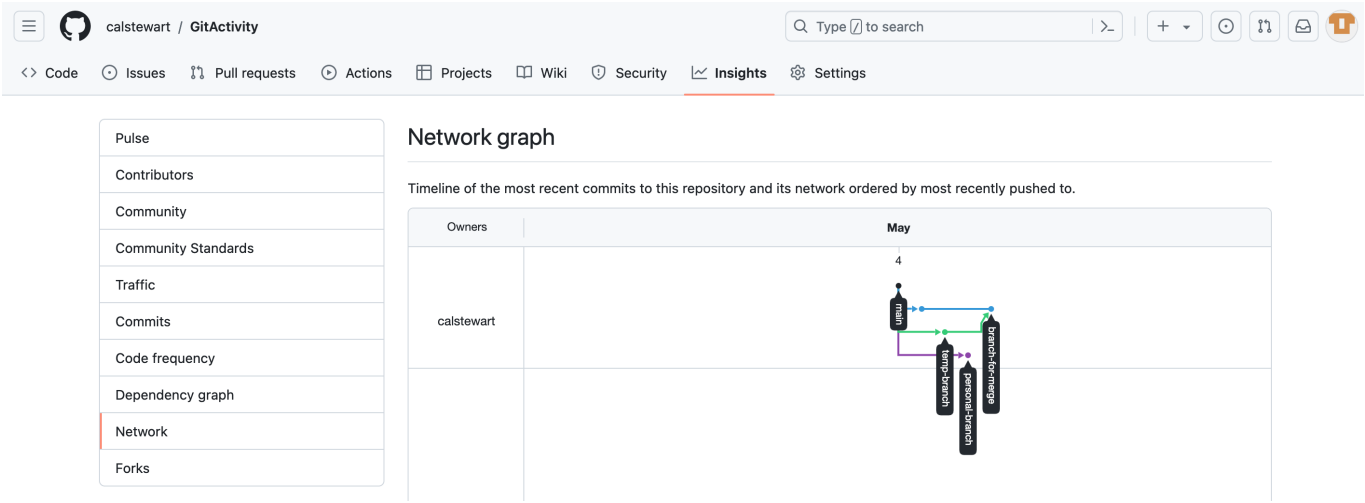
The lines between `<<<<<<< HEAD` and `=======` represent changes from the current branch (branch-for-merge), while the lines between `=======` and `>>>>>>> temp-branch` represent changes from the branch being merged (temp-branch). The conflict arises because both branches have made changes to the

same section of the file. Resolving the conflict involves choosing which changes to keep or combining them appropriately. Use the following command to accept the value of the temp branch:

```
git checkout --theirs Assignment.txt
```

Using --ours keeps the changes from the current branch (the branch you are merging into), while --theirs keeps the changes from the branch being merged. After using one of these commands to resolve the conflict for the specific file, you need to add and commit the changes to finalize the resolution. Use the commit message "resolved conflict with Assignment.txt". Finally, rerun the merge command and push.

Login to GitHub on your browser. Navigate to the repository for this activity. On the top ribbon, you will see a "Insights" button. On the Insights page, there is a side ribbon with the option to view your repository's network. Take a screenshot of the network graph. It should look like the following, demonstrating that the temp-branch was merged successfully into the branch-for-merge.



Step 4: deleting branches

Next, you will delete the temporary branch since it has been merged successfully. To delete the branch, run the following command:

```
git branch -d branchname
```

Run the `git branch` command to ensure that your deletion was successful. However, this only deletes the local copy of the branch. To remove the branch from your remote repository as well run the following command:

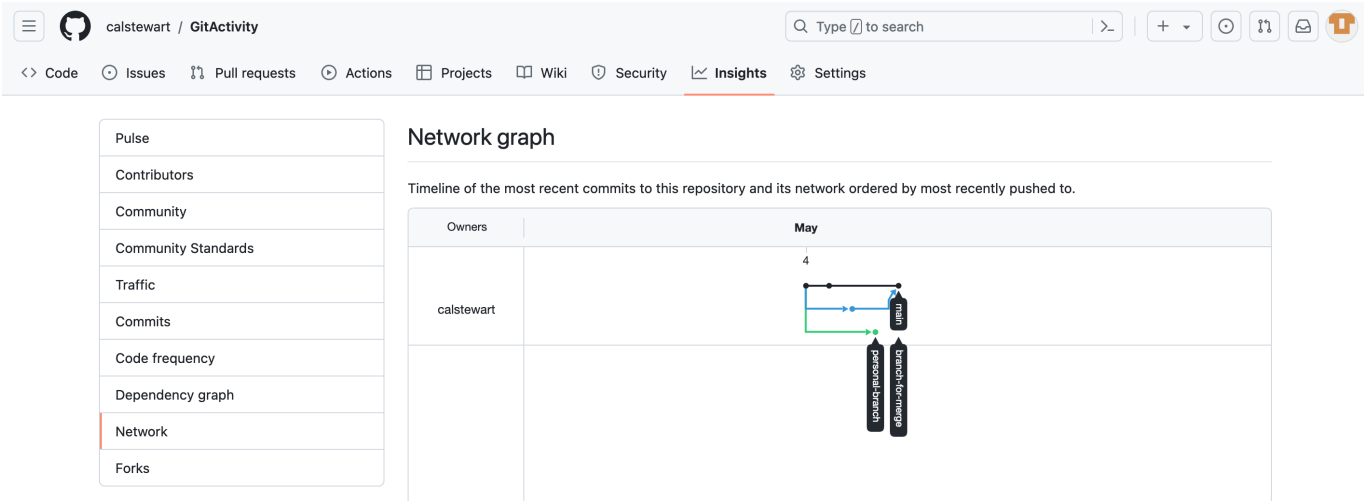
```
git push origin --delete branchname
```

Step 5: merging with main

Once your development is done, you should merge any working branches with main. In this activity you will only merge the branch-for-merge. Run the following commands to merge your branch into main. Since you did not develop on main, there are no conflicts to resolve.

```
git checkout main
git merge <user_id>-branch-for-merge
git push origin main
```

Reload the webpage displaying your network. Screenshot your network graph again. It should look like the following demonstrating that branch-for-merge and main have been merged and temp-branch has been successfully deleted.



At this point should have successfully completed the lab. If you are unfamiliar with Git, you should continue to explore the branches and investigate the differences between branches. For example, "branch" label the Assignment.txt file in main should still reflect the edits made in the temp-branch. Main should also display the Icarus file you added from branch-for-merge. While the still unmerged personal-branch should show a different value for the "branch" label in Assignment.txt and the Goblet file that was added. These differences can be explored in the command line, PyCharm, or your web browser.

Turn-in

By the end of the exercise, your gitlab repository will have three branches. One is the unmerged person branch. The branch-for-merge, while merged with main was never deleted so should still be present. And of course, your main branch.

Your iLearn submission should include the link to your git repository as well as the two screenshots of your network at the different points in the assignment.