Git and LaTeX Worksheet

Justin Li
justinnhli@oxy.edu
Occidental College

1 Instructions

This worksheet is due March 3, 2025 at midnight, to be submitted as a GitHub repository URL to Canvas. The repository should contain all files requires to compile this worksheet with your answers. You should only change this `document.tex` file and the `references.bib` file; do not change any other file in this starting repository. You should not use any additional packages, and are not allowed to use the `\usepackage{}` command. Additionally, the output should be formatted correctly: your answers should be appropriately nested under the questions, command-line commands should be in monospace, and images should be positioned appropriately.

2 Git Questions

2.1 General questions

1. What is a version control system? Why are they useful?

   A version control system (VCS) is a tool that helps manage changes to source code over time. It allows multiple developers to work on a shared project simultaneously, keeps track of modifications, and enables reverting to previous versions if needed. Version control systems are useful because they prevent code conflicts, maintain a history of changes for reviewing and debugging purposes, and faciliate collaboration.

2. What is the difference between git and GitHub?

   Git is a distributed version control system that allows developers to track code changes, branch, and merge efficiently.

   GitHub is a the standard-use cloud-based platform that provides hosting services for Git repositories. It offers collaboration tools, issue tracking, pull requests, and more, making it easier for teams to work together on projects.

3. What is a repository?

   A repository (or "repo") is a storage location for a project's code, configuration files, and history of changes. It contains all commits, branches, and metadata needed to manage a software project.

4. What is a commit?

   A commit is a snapshot of changes made to a repository at a specific point in time. Each commit has a unique identifier (SHA hash) and includes a commit message describing the modifications.

5. What is the commit graph?

   The commit graph is a visual representation of the commit history in a Git repository. It shows the relationships between commits, branches, and merges, helping developers understand the project's evolution and collaboration structure.

6. What is your preferred local git client (eg., command line, GitHub Desktop, GitKraken, etc.)?

   I usually use GitHub Desktop.

2.2 Local Usage

1. What is the difference between adding a file to the staging area and committing a file?

   Adding a file to the staging area (git add) means you are preparing the changes to be included in the next commit. The staging area acts as a preview of what will be committed.
   Committing a file (git commit) takes the staged changes and saves them as a new snapshot (commit) in the repository history. Each commit has a unique identifier and a commit message.

2. What is a commit message, and why is it important for them to be meaningful?

A commit message is a short description that explains what changes were made in a commit. Meaningful commit messages are important because they:

- Help developers understand the purpose of changes.
- Make debugging easier by providing context.
- Improve collaboration by keeping a clear record of updates. A good commit message should be concise yet descriptive, such as "Fix login bug by correcting authentication logic"

3. Starting with an empty repository, what sequence of commands/actions would result in the following commit graph? You may give a sequence of `git` commands, or describe (with screenshots) how you would do this in your preferred graphical git interface.

   `A---B---C---D`

   git init
   touch file.txt
   git add file.txt
   git commit -m "Commit A"

   echo "Change 1" file.txt
   git commit -am "Commit B"

   echo "Change 2" file.txt
   git commit -am "Commit C"

   echo "Change 3" file.txt
   git commit -am "Commit D"

4. If you are currently at commit D above, how would you recover code from commit B? What sequence of commands/actions would let you do so? You may give a sequence of `git` command-line commands, or describe (with screenshots) how you would do this in your preferred graphical git interface. Assume the commit hashes are AAAAAA..., BBBBBB..., etc.

   You can use one of the following approaches:

   Option 1: Checkout (temporary switch to commit B, but keeps repository at D)

   git checkout BBBBBB

   This makes your working directory reflect commit B, but you are in a detached HEAD state.

   Option 2: Create a new branch from commit B

   git checkout -b new-branch BBBBBB

   This allows you to work from commit B without losing commit D.

   Option 3: Reset to commit B (removes commits C and D from history!)

   git reset –hard BBBBBB

   This will remove commits C and D permanently unless you have them backed up.
   Option 4: Restore only specific files from commit B

   git checkout BBBBBB – file.txt

   This takes file.txt from commit B and brings it into the working directory without changing the commit history.

5. Imagine you created a git repository for your project, but only commit your changes once a week on Sundays. You got your code working on Tuesday, but then broke your code on Friday. What can you do to get the working version of your code back?

   If you commit only once a week (on Sundays) and your last commit was before breaking your code, you have several options:

   Option 1: Use git stash (if you haven't committed broken code)
   git stash

   This removes the broken changes while keeping them saved in a stash. You can recover them later with git stash pop.

   Option 2: Use git checkout to restore an older version

   git checkout HEAD 1

   This temporarily restores the last committed version of the code.

   Option 3: Reset to the last good commit (if you've already committed the broken code)

   git reset –hard HEAD 1

   This removes the last commit (the broken changes) and resets the repository to its previous state.

Option 4: Use git revert (safe option that keeps history)

git revert HEAD

This creates a new commit that undoes the broken changes while keeping the commit history intact.

Best Practice: To avoid losing work, it's a good idea to commit more frequently and use branches to experiment with changes before merging them into the main branch.

2.3 Branching and Merging

1. What is a branch? Why are they useful?

   A branch in Git is a separate line of development that allows you to work on new features, bug fixes, or experiments without affecting the main codebase. Branches are useful because they:

   - Enable multiple developers to work on different features simultaneously.
   - Make debugging easier by providing context.
   - Allow safe experimentation and rollback if needed.

2. Starting with an empty repository, what sequence of commands/actions would result in the following commit graph? You may give a sequence of `git` command-line commands, or describe (with screenshots) how you would do this in your preferred graphical git interface.

```
A---B---C---D
     \
      E---F
```

   git init
   touch file.txt
   git add file.txt
   git commit -m "Commit A"

   echo "Change B" → file.txt
   git commit -am "Commit B"

   echo "Change C" → file.txt
   git commit -am "Commit C"

   echo "Change D" → file.txt
   git commit -am "Commit D"

   git checkout -b feature-branch B
   echo "Change E" → file.txt
   git commit -am "Commit E"

   echo "Change F" → file.txt
   git commit -am "Commit F"

3. Why is a merge? Why are they useful?

   A merge integrates changes from one branch into another. Merges are useful because they combine features or fixes into the main branch, preserve commit history while integrating multiple lines of development, and ensure collaboration without overwriting others' work.

4. Imagine you are currently at commit D above. What sequence of commands/actions would result in the following commit graph? You may give a sequence of `git` commands, or describe (with screenshots) how you would do this in your preferred graphical git interface.

```
A---B---C---D---G
     \         /
      E---F---/
```

   git checkout main # Ensure you are on the main branch
   git merge feature-branch # Merging feature-branch into main
   git commit -m "Merge feature-branch into main"

5. What is a merge conflict? When do they occur?

   A merge conflict happens when Git cannot automatically reconcile changes from two branches because they modify the same part of a file differently. It occurs when two branches edit the same line in a file, one branch deletes a file that another branch modifies, or structural changes cause ambiguity in merging.

6. Starting with an empty repository, despite a sequence of commands/actions that would result in a merge conflict. Include the exact edits and `git` commands or screenshots of the graphical git interface. Include the output or screenshot that shows the resulting merge conflict.

git init
echo "Initial content" → file.txt
git add file.txt
git commit -m "Commit A"

git checkout -b branch1
echo ”Change from branch1” → file.txt
git commit -am ”Commit B

git checkout main
echo ”Change from main branch” → file.txt
git commit -am ”Commit C”

git merge branch1

Expected output:

Auto-merging file.txt
CONFLICT (content): Merge conflict in file.txt
Automatic merge failed; fix conflicts and commit the result.

To resolve the conflict, open file.txt, manually choose the correct changes, then:

git add file.txt
git commit -m ”Resolve merge conflict”

2.4 Remotes

1. What is a remote?

   A remote in Git is a reference to a version of your repository that is hosted on a server (e.g., GitHub, GitLab, Bitbucket). Remotes allow multiple developers to collaborate by sharing and syncing code across different machines.

2. What does pushing and pulling do?

   Pushing `(git push)` uploads local commits to a remote repository, making them available to others. Pulling `(git pull)` fetches the latest changes from a remote repository and merges them into the local branch. This keeps your local copy up to date with changes made by others.

3. Imagine you created a git repository for your project on your laptop and commit regularly, but only push your code to GitHub once a week on Sundays. Your laptop caught on fire on Friday. What can you do to get your code back?

   If you only push your code to GitHub on Sundays and your laptop was lost on Friday, the latest version of your code available on GitHub is from last Sunday. Unfortunately, any work done after that (Monday–Friday) that wasn't pushed is lost.

   To recover the latest available version:
   Get a new computer. :)
   Clone the GitHub repository:

   git clone https://github.com/your-username/your-repo.git

   This will restore the latest version of your code that was pushed on Sunday.

3 LaTeX

Find a source of each of the following types and add it to `references.bib`, with the appropriate data. Your sources do not have to relate to your project. Looking at Overleaf (2025) [8] and Wikipedia (2025) [9] may be helpful,

- a journal article [7]
- a conference article [6]
- a PhD or Master's thesis [1]
- an article in an edited popular media venue (newspaper, magazine, etc.) [4]
- a book [5]
- a chapter of a book [2]
- a YouTube video [10]
- a piece of technical documentation (e.g., a programming language reference, and API documentation, etc.) [3]

Additionally, in you own words, explain the difference between `\cite{}` and `\textcite{}`. When should they each be used? Demonstrate your answers by using one of them with each of your references from above.

`\cite{}` generates an in-text citation, usually appearing as a number or author-year format, and is used when the author's name is not directly mentioned in the sentence (e.g., ”Deep learning has advanced significantly [6].”).

`\textcite{}` integrates the author's name into the sentence naturally, followed by the year in parentheses, and is used when referring to the author directly (e.g., ”He et al. (2016) [6] introduced deep residual learning.”).

References

[1] Arsenault, Ashley M. “Understanding Media Richness and Social Presence: Exploring the Impacts of Media Channels on Individuals' Levels of Loneliness, Well-Being, and Belonging”. accessed 2025-03-03. PhD thesis. University of Montana, 2022.

URL: https://scholarworks.umt.edu/etd-communication/.

[2] Darwin, Charles. “Chapter IV: Natural Selection”. In: *The Origin of Species*. accessed 2025-03-03. John Murray, 1859. URL: https://www.gutenberg.org/files/1228/1228-h/1228-h.htm#chap04.

[3] Foundation, Python Software. *Python 3.9.1 Documentation*. accessed 2025-03-03. 2025. URL: https://docs.python.org/3/.

[4] Franzen, Jonathan. “The Climate Crisis Is Worse Than You Can Imagine. Here’s What Happens If You Try.” In: *The New Yorker* (2019). accessed 2025-03-03. URL: https://www.newyorker.com/culture/cultural-comment/what-if-we-stopped-pretending.

[5] Harari, Yuval Noah. *Sapiens: A Brief History of Humankind*. accessed 2025-03-03. Harper, 2015. URL: https://www.ynharari.com/book/sapiens-2/.

[6] He, Kaiming et al. “Deep Residual Learning for Image Recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. accessed 2025-03-03. 2016. URL: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf.

[7] accessed 2025-03-03.

[8] Overleaf. *Bibliography Management with bibtex*. accessed 2025-02-11. 2025. URL: https://www.overleaf.com/learn/latex/Bibliography_management_with_bibtex.

[9] Wikipedia. *BibTeX*. accessed 2025-02-11. 2025. URL: https://en.wikipedia.org/wiki/BibTeX.

[10] YaleCourses. *The Science of Well-Being*. accessed 2025-03-03. 2018. URL: https://www.youtube.com/watch?v=6zQ0nx0vVnM.