CS 520
In-class exercise 1
Advanced uses of Git

Due: **Tuesday, September 28, 2021, 9:00 PM** via Moodle. This in-class exercise is a group submission. This means that **each group only needs to submit their solution once** and also that every student in a group will get the same grade. You will work with students within your group, but not with students from other groups. Multiple groups' submissions may not be created jointly. Late assignments will not be accepted without **prior** permission except for extenuating circumstances.

Overview and goal

The high-level goal of this exercise is to gain more experience with Git, in particular working with querying, branches, cherry-picking commits, and understanding the differences among reset, rebase, and revert.

What to do?

Forming groups

1. Team up in groups of size 4 or 5. (If you cannot find enough members, raise your hand and ask the instructor.)
2. Create a new group on Moodle (see "In-class exercise 1: Group selection"), and add all group members.

Set up

1. Check if you are using a newer version of git (version $\geq$ 2.7.4). Update your git if needed.
2. Clone the basic-stats git repository:
   `git clone https://github.com/LASER-UMASS/basic-stats basic-stats`
3. Create a (second) local fork by locally cloning **again**, this time from the first local clone:
   `git clone basic-stats basic-stats-fork`

Querying

1. In the `basic-stats` folder, what is the set of the different authors of the commits for the README.md file?
2. In that README.md file, what is the hash of the commit that mentions the ant build tool?

Branching and cherry-picking

Goal: Cherry-pick changes to a particular file from a different branch.

1. In `basic-stats-fork`, checkout the tagged version `v1.0.0` and create a branch, named `feature-branch`. **Visualize the relationship** between feature-branch and main (used to be called master).

   USEFUL TIP: Use options available for `git log` command, such as `--graph` and `--oneline`, or use an external tool, such as gitk or SourceTree.

2. Familiarize yourself with the `cherry-pick` command.

   USEFUL RESOURSE: http://think-like-a-git.net/sections/rebase-from-the-ground-up/cherry-picking-explained.html

3. Determine all commits in the `main` branch that affect `README.md` and note their commit hashes.

   USEFUL TIP: A commit may affect files other than `README.md`.

4. Cherry-pick all of the those commits from `main` that do not already exist in `feature-branch`. Note the number of commits, and **visualize the relationship** between `feature-branch` and `main` after cherry-picking.

   USEFUL TIP: If you are seeing a conflict when cherry-picking, you are doing something wrong. Consider:

   - Are you in the basic-stats-fork directory?
   - Did you checkout the tagged version? Are you on the feature-branch?
   - Is the commit-hash used for cherry-pick applicable to `README.md`?
   - Are the commits applied to `README.md` in the correct chronological order?

5. Push `feature-branch` back to `basic-stats`:
   `git push --set-upstream origin feature-branch`

   USEFUL TIP: Verify that the `feature-branch` is reflected in `basic-stats` repository.

Rebasing
Goal: Squash multiple commits into one commit.

1. Familiarize yourself with the `rebase` command.

   USEFUL RESOURCE: https://www.atlassian.com/git/tutorials/rewriting-history/git-rebase

2. In `basic-stats-fork` (feature-branch), rebase and squash the last 4 commits:
   `git rebase -i HEAD~4`

3. Verify the result with `git log` (**visualize the relationship** between `feature-branch` and `main` after rebasing).

   USEFUL TIP: Did you choose the correct option for each of the four commits in a file that pops up while running the `rebase` command in previous step?

4. Run `git status` and discuss the risks of rebasing. What happens if you pull from another repository?

5. Pull from `basic-stats` and run `git log` again (**visualize the relationship** between `feature-branch` and `main` after pulling).

6. do `git push` to push your changes to the `basic-stats`

7. In `basic-stats`, checkout the `feature-branch` branch and verify that changes pushed are reflected.

Resetting and reverting
Goal: Undo a commit before/after it is pushed.Answering questions

1. Familiarize yourself with the `reset` and `revert` commands.

2. In `basic-stats-fork`, checkout the `main` branch:
   `git checkout main`

3. Edit the `README.md` file, and commit your change.

4. Use `reset` to undo the commit:
   `git reset HEAD~1`

   USEFUL TIP: Do not push your changes to the repository; only commit your changes.

5. Make further edits to the `README.md` file, and commit and *push* your change.

   USEFUL TIP: Git won't let you push changes to a branch to a repository with that branch checked out. If you have having trouble pushing, make sure you have the `feature-branch` checked out in `basic-stats`.

6. Discuss the risks of resetting at this point. What happens if you pull from another repository?

7. Use the `revert` command to undo your commit.

8. Verify the result with `git log`.

Questions

Using your notes and results, and answer the following questions:

1. For the `basic-stats` folder in `main`, list the set of the different authors of the commits to the README.md file. What is the hash of the commit to that README.md file that mentions the ant build tool?

2. How many commits did you cherry-pick?

3. Are the commit hashes of the cherry-picked commits identical in `main` and `feature-branch`? Briefly explain why.

4. What happens if you merge a branch from which you previously cherry-picked single commits? How often do the cherry-picked commits appear in the history? Briefly explain why.

5. What are the risks of rebasing? Briefly describe a use case in which rebasing can be safely applied.

6. What are the risks of using reset when a commit has already been pushed?

7. Does revert remove the reverted commit? Briefly explain how revert works.

Deliverables

Your submission, via Moodle, must be a single (one per group) archive file (.zip or .tar.gz) with name <group name>-inclass1.<zip/tar.gz>, containing:

1. `answers.txt`: A plain-text file with your answers to the above 7 questions. List all group members on top of this file.

2. `relationships`: A file that shows the 4 visualized relationships (git log screenshots, ascii art, diagram, picture, etc.).

3. `basic-stats`: The basic-stats repository. Note that this should **not** be the files in the basic-stats working copy, but instead the **repository** (which is the `.git` directory in `basic-stats`.) For example, on a Linux-based machine (e.g., MacOS), you can use the terminal from the `basic-stats` directory and run the command
   ```
   tar -vczf basic-stats.tar.gz .git
   ```

4. `basic-stats-fork`: The basic-stats-fork repository. Note that this should **not** be the files in the basic-stats-fork working copy, but instead the **repository** (which is the `.git` directory in `basic-stats-fork`.) For example, on a Linux-based machine (e.g., MacOS), you can use the terminal from the `basic-stats-fork` directory and run the command
   ```
   tar -vczf basic-stats-fork.tar.gz .git
   ```

Other useful resources to learn Git

- `try.github.io`: An interactive Git tutorial.
- `https://learngitbranching.js.org/`: Learn Git branching. An interactive tutorial with a graphical view of what is happening in a git repository.
- `https://git-scm.com/book/en/v2`: The Pro Git Book. A dense "what makes Git tick" guide.