# Assessment: Introduction to Version Control

To earn a micro-badge for this workshop, complete the prompts on the following page. You should do yourwork entirely on a command line editor (Terminal, Git Bash, etc.). When you're finished, zip this directory and submit it to GradPathways.

This directory has already been put under version control with Git. For this assessment, you'll be working across a number of branches to change and create data. You'll also resolve conflicts in this data as needed.

For questions marked "Short answer", record your notes in a plain text file titled "answers.txt". Be sure to note which question you're responding to (e.g. 1b, 3c, etc.).

**Links**

- GradPathways badge
- Event page
- Workshop reader

## Rubric

1. Working code: were you able to produce successful script for each prompt?
2. Understanding your actions: can you explain what your actions do and why you implemented them?
3. Critical reflection: do your short answers provide context (conceptual, domain-specific, etc.) for your actions?

## Prompts

1. Getting started
    a. Before you make any changes to this directory, do some exploring. Note to yourself: which branch am I on? How many branches are? Do the branches seem similar?
    b. Once you've explored the directory, return to `main`. From `main`, make a new branch with your first name and last initial. It should look like so: `first-name_last-initial`.
    c. On your new branch, you'll find a simple `bash` script titled `make_madlib.sh`. It accepts three arguments from the command line and puts them together to create a file called `madlib.txt`. Those arguments are:
        - `-a` adjective
        - `-n` noun
        - `-v` verb (simple present works best) Run this script with arguments of your choosing to create `madlib.txt`. You can do so with

          ```
          bash make_madlib.sh -a <your adjective> -n <your noun> -v <your verb>
          ```

2. Commit your changes
    a. Once you've created `madlib.txt`, stage and commit your changes.
    b. Merge your branch into `main`.
    c. (Short answer) In a sentence or two, describe the results of this operation. Did any problems occur? Why or why not?
3. Fixing another branch
    a. With your changes made to `main`, it's time to look at something a little messier. Checkout the `fixme` branch. Merge `dev` into `fixme`. This will create a merge conflict.
    b. (Short answer) Investigate the merge conflict. In a sentence or two, explain why it has happened.
    c. Correct the merge conflict and commit your changes.
    d. Rename `fixme` to `fixed`, using `git branch -m fixed`.
4. Saving your work
    a. While still on `fixed`, generate a log and save it to a new file. You can do so with

       ```
       git log --oneline > log.txt
       ```

    b. Commit your changes and return to `main`.
    c. On `main`, checkout *just the log file*. See if you can figure out how to do this one your own. If you're stuck, look at this answer on StackOverflow.
    d. Commit the log to `main`. Once you're finished, zip this whole directory and submit it to GradPathways. To check your work, the directory should contain the following files:

       ```
       answers.txt
       instructions.Rmd
       instructions.pdf
       log.txt
       madlib.txt
       make_madlib.sh
       ```