Lesson 2 Reflections

Reflection 1

What happens when you initialize a repository? Why do you need to do it?

When you initialize a repository, it begins to be tracked by git. Git stores information about the files in the repository in a hidden folder called .git.

You need to do it because git will not automatically do it for you. Git allows you to choose which files to commit by letting you "stage" them for the commit.

Reflection 2

How is the staging area different from the working directory and the repository? What value do you think it offers?

The repository is what gets tracked and shared with git. The working directory is where you are making changes to files, and the staging area is where you can choose which changes to commit together.

This allows the user the control of being able to group changes that are logically associated and commit them together.

Reflection 3

How can you use the staging area to make sure you have one commit per logical change?

You can add and remove changed files from the staging area and tailor the contents of commits to only have files that are logically related.

Reflection 4

What are some situations when branches would be helpful in keeping your history organized? How would branches help?

Branches can keep your history organized by allowing you to experiment with changes without fear of breaking the working part of the project.

Branches help by providing a space where this experimentation can take place.

Reflection 5

How do the diagrams help you visualize the brach structure?

The diagrams help me visualize the structure because each node in the graph represents a commit and the edges form the paths that the head can travel.

Reflection 6

What is the result of merging two branches together? Why do we represent it in the diagram the way that we do?

The result of merging two branches together is a single branch that comtains all the commits of both branches, using parts of each to create a new version of the file or project. Git represents it as two branches coming together and forming a linear commit path. This representations helps me visualize the history of the branches and how they merge.

Reflection 7

What are the pros and cons of Git's automatic merging vs. always doing merged manually?

One of the cons of Git's automatic merging is that it could automatically include code that passes the merging criteria, but you do not want to include in the merge. On the other hand, Git's automatic merge seems like it will perform the desired operation most of the time and that could save a lot of time.