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

A: Git creates the .git (wich is hidden) folder and starts tracking changes via commits. You need to do it to let Git track your files and changes made to them and allow you to compare them.

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

A: The staging area is where you commit all the changes made to your files (you can decide wich ones you will add) and are ready to be sent to your local repository. The working directory is a copy of your project intended to be modified by you (or someone else).

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

A: Git gives you the possibility to decide what files you want to add to the repository, you have to be sure to only commit changes that make sense and are important to your project.

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

A: When you want to introduce a new experimental feature but you are still working on it and do not know if it will work out or you are not sure these changes will be on the final version.

Branches would help you in making this changes you are working on without affecting the master file.

Q: How do the diagrams help you visualize the branch structure?

A: It is a way to look graphically all the commits you have made to your project and which ones belong to which branch.

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

A: The changes made to the file either by yourself or by someone else can be brought together and now be part of the master branch.

Q: What are the pros and cons of Git’s automatic merging vs. always doing merges manually?

Commands:

git add | Add a file to the staging area

git commit | Commit the files from the staging area to the repository

git status | Show which files are in the staging area, and which files from the working directory are not part of the repository

git diff | Git diff without arguments is comparing the working directory with the staging area

git diff --staged | Compare the staging area with the repository

git reset --hard | Undo all changes! There is no rollback for this. So, be careful!

git checkout master | Restore the "Head" to the latest check-in (this will be part of a later lesson/module)

git log --graph --oneline <branch1> <branch2> | Show a visual representation of the commit history within different branches

git checkout -b new\_branch\_name | Creates a new branch and do a checkout on this branch in one call instead of git branch new\_branch\_name and git checkout new\_branch\_name

git merge branch1 branch2 | Merge two branches, branch2 get merged into branch1

git branch -d name | Remove the branch with the specified name

git show commitId | Show the changes made in this commit compared to the previous version. This is working even after merging.