**2. GIT Basics - Dulana Nuwanjith - 06/08/2023**

1. Create a repository “training” and share with your trainer
2. Get a copy of this repository to your local machine
3. Create a new branch ‘oop-concepts’
4. Create a directory ‘oop-exercise’
5. Add your oop exercise answer to directory ‘oop-exercise’ and push
6. Create a pull request from your branch to master and request approval from trainer
7. After pull request is accepted, delete branch ‘oop-exercise’ from both local and remote
8. List down git command you used in the exercise

* git clone <repository\_url>
* git checkout -b oop-concepts
* git add .
* git commit -m "Add OOP exercise answer"
* git push origin oop-concepts
* git checkout master
* git branch -d oop-concepts
* git push --delete origin oop-concepts

1. Describe about following commands and practice

|  |  |
| --- | --- |
| git fetch | To retrieve the latest changes from a remote repository without automatically merging them into your local branches. It is a useful command for keeping your local repository up-to-date with changes made by other contributors in the remote repository. |
| git branch | To manage and view branches in a repository. It allows you to list existing branches, create new branches, delete branches, and see which branch you are currently on. The git branch command is versatile and has multiple options depending on how you want to use it. |
| git pull | To fetch and integrate changes from a remote repository into your local branch. It is a combination of two commands: git fetch and git merge. When you run git pull, Git will automatically fetch the latest changes from the remote repository and merge them into your current branch. |
| git log | To view the commit history of a repository. It displays a list of commits in reverse chronological order, meaning the most recent commits are shown first. The git log command is helpful for understanding the project's history, reviewing changes made by collaborators, and identifying specific commits. |
| git status | To show the current status of your working directory and staging area. It provides valuable information about the changes you've made since your last commit and the current state of your branch. The git status command is often the first command you run to get an overview of your repository's status. |
| git diff | To show the differences or changes between different states of a Git repository. It is a powerful tool for comparing various aspects of your code, such as changes between commits, changes between branches, or changes within individual files. |
| git stash | To save the current changes in your working directory that are not ready for committing but you want to temporarily set aside. It allows you to save your work on a "stack" of changes, known as the "stash," so that you can switch to a different branch or apply another patch without committing your current changes. |
| git stash pop | To apply the most recent stash from the stash stack and remove it from the stack in one step. It allows you to reapply the changes that were previously stashed and brings your working directory and staging area back to the state they were when you created the stash. The git stash pop command is a convenient way to retrieve your stashed changes and continue working with them without keeping the stash in the stash stack. |

1. What is a merge conflict?

A merge conflict occurs in Git when two or more branches have made conflicting changes to the same part of a file or files. When you attempt to merge these branches, Git is unable to automatically determine which changes should take precedence, as the changes conflict with each other. As a result, Git marks the affected files as having a conflict, and it requires manual intervention from the user to resolve the conflicts.

1. Create a new branch “my-branch” from master
2. Checkout to master branch
3. create a new text file “example.txt”
4. Add text “this is from master” and push to master branch
5. Checkout to “my-branch”
6. create a new text file “example.txt”
7. Add text “this is from my-branch” and push to my-branch
8. Merge “my-branch” with master
9. Mention the steps followed to resolve the merge conflict

* Identify the conflict
* Open the conflicted file
* Choose the correct version
* Save the resolved file
* Stage the resolved file
* Commit the merge
* Push the merged changes

1. What is the .gitignore file?

a special file in a Git repository that is used to specify which files and directories should be ignored by Git. It allows you to tell Git to exclude certain files and patterns from being tracked and included in the version control system.

When you add files to a Git repository, Git will track them by default, which means changes to these files will be detected, and you can stage and commit them. However, there are certain files and directories that you typically don't want to include in the version control, such as temporary files, build artifacts, log files, and sensitive information like API keys or passwords. The .gitignore file helps you specify these files and patterns to be excluded.

1. Rename the answer sheet as “git-basics-<your-name>-<date>” and submit