|  |  |  |
| --- | --- | --- |
| SCHOOL OF INFORMATION AND TECHNOLOGY | | |
| NAME: Shekinah T. Sotero | DATE PERFORMED: 10/23/24 |  |
| Section: IDC2 | DATE SUBMITTED: 10/26/24 |

# SYSADM1 – Git Basics

Answer the following research questions about Git, GitLab desktop and GitHub.

1. What is Git, and why is it important in software development?

Git is a tool that tracks code changes, allowing developers to collaborate, manage versions, and keep a history of project updates. It ensures safe experimentation and teamwork.

1. How does Git track changes in a project?

Git saves snapshots of project files when changes are committed. It uses checksums to identify versions and efficiently stores updates, not entire files.

1. What is the difference between a local repository and a remote repository in Git?

Local repository means that it is stored on local machines for an individual user like your personal computer while remote repository are hosted on a remote in which it could be on the internet or an off-site server.

1. What are the basic Git commands?

|  |  |  |  |
| --- | --- | --- | --- |
| **Commands** | **Description** | **Commands** | **Description** |
| * 1. git init | This creates a repository | 7. git branch | Manage branches. |
| * 1. git add | Stage files for commit | 8. git merge | Combine branches. |
| * 1. git commit | Save changes to the repository | 9. git clone | Copy a remote repository to your computer. |
| * 1. git status | Show the repository’s state | 10. git fetch | Gets the latest changes from the remote repository but doesn’t apply them to your files yet. |
| * 1. git push | Upload changes to a remote repository. | 11. git reset | Undo changes by moving back to a specific point in your project. |
| * 1. git pull | Fetch and merge changes from a remote repository. | 12. git log | Shows a list of all the past changes (commits) in your project. |

1. How do you check the status of a Git repository?

By running the command “git status” to see the untracked, staged, or modified files and the branch’s current state.

1. What is the purpose of branches in Git, and how do you create and switch between them?

Branches in git let’s you work on features without affecting the main code. You have to first enter the command “git branch [branch-name]” to create a branch. The nest step is to switch to a brand in which you can use two commands, “git checkout [branch-name] or git switch [branch-name]” which is the preferred one.

1. What are GitLab Desktop and GitHub, and how are they different from Git?

Git is a version control system in which GitHub and GitLab are platforms for hosting Git repositories, with GitLab offering built-in CI/CD tools. GitLab Desktop is a GUI tool for managing GitLab projects locally.

1. How do you connect a local Git repository to a GitLab or GitHub repository?

1st: Create a repository on GitLab or GitHub.

2nd: Copy the repository URL.

3rd: Run the commands as follows

* git init
* git add .
* git commit -m "Initial commit"
* git remote add origin <repository-URL>
* git push -u origin master

|  |
| --- |
| Example:  git init  git add .  git commit -m "Push existing project to GitLab"  git remote add origin <https://sample>  git push -u origin master |

4th: Ensure the remote repository is empty and matches your branch name.

1. What are the steps to collaborate with others using GitLab or GitHub?

To collaborate using GitLab or GitHub, clone the repository “**git clone repository-url**”, create a branch for your work “**git branch [branch-name]**”, commit and push changes “**git push origin [branch-name]**”, and open a pull/merge request for review and merging.

1. How do you resolve merge conflicts in Git?

You have to edit any conflicting files, and mark the resolved conflicts with “**git add [file-name]**” and commit the resolution using “**git commit**”.

1. What is a pull request, and why is it used in GitHub?

A pull request is a request to merge changes from one branch into another, typically used in GitHub. It facilitates code review, discussions, and collaboration before merging.

1. What are some best practices for writing commit messages?

* Keep it short and clear
* Provide extra details if needed
* It should be consistent and concise
* Capitalize the subject line.
* Write good commit messages.
* Use the imperative mood