|  |  |  |
| --- | --- | --- |
| SCHOOL OF INFORMATION AND TECHNOLOGY | | |
| NAME: Hyra Cayambas | DATE PERFORMED: 11/20 |  |
| Section: IDC1 | DATE SUBMITTED: 11/20 |

# 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 distributed version control system that tracks changes in source code, enabling collaboration and maintaining project history. It is crucial for team-based and distributed development.

1. How does Git track changes in a project?

Git creates snapshots of file changes in a repository and stores them as commits. Taking snapshots means Git records the state of project files at specific points in time. These commits are tracked through Directed Acyclic Graph which is like a family tree for commits in Git. Every commit points to its parent and it never loops back to an earlier commit or “no going in circles”.

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

A local repository is stored on a developer's computer, while a remote repository is hosted on a server or cloud (e.g., GitHub/GitLab) for team access.

1. What are the basic Git commands?

git init: Initialize a repository

git add: Stage changes

git commit: Save changes

git commit –m: Commit message

git status: Check repository state

git push: Send changes to a remote repo

git pull: Fetch and merge changes from a remote repo

git clone: Creates a local copy of a remote repository

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

Use git status to see staged, unstaged, or untracked files and the current branch.

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

Branches enable parallel development. Branches in Git let you work on different versions of your project at the same time without messing up the main version. Think of branches as separate workspaces for different tasks, all based on the same starting point. Use git branch [branch\_name] to create a branch and git checkout [branch\_name] or git switch [branch\_name] to switch.

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

Git is the version control system, while GitHub and GitLab are platforms providing remote repositories and additional collaboration tools like pull requests and CI/CD.

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

Use git remote add origin [repo\_URL], then push changes with git push -u origin [branch\_name]. Or you could manually create a repository in GitHub/GitLab, clone that to your local repository then use git add, git commit and git push.

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

Users have to be added as collaborators before they can make/add changes to a repository. Only then can users: clone the repository, create a branch for your changes, commit and push changes to the branch, open a pull/merge request for review.

1. How do you resolve merge conflicts in Git?

Identify conflicts with git status, manually edit conflicting files, mark them as resolved with git add, and commit the resolution.

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

A pull request proposes changes from one branch to another, enabling code review and approval before merging. It is a way to ask someone to review and approve changes you've made to a project before adding them to the main version. It helps ensure your changes are correct and won't break anything and team members can discuss, suggest improvements, or point out issues.

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

Use clear and descriptive messages. Start with a concise summary, followed by a detailed explanation if necessary. Use imperative language (e.g., "Fix bug" or "Add feature").