|  |  |  |
| --- | --- | --- |
| SCHOOL OF INFORMATION AND TECHNOLOGY | | |
| NAME: Dale Matthew R. Boquiren | DATE PERFORMED:11/20/2024 |  |
| Section:IDC1 | DATE SUBMITTED:11/20/2024 |

# 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 versions of files. It is often used to control source code by programmers who are developing software collaboratively. Git is important in software development because it enables teams to maintain a history of changes, work collaboratively without fear of losing code, and manage complex codebases.

1. How does Git track changes in a project?

- Git tracks changes by taking snapshots of the project files at certain points in time, called commits. Each commit records the changes made to the files since the last commit. These changes are stored in a local repository and consist of metadata like the author, timestamp, and a unique commit ID, along with the content of the files. Git uses a system of objects to store the differences (or diffs) between versions, efficiently tracking changes over time.

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

- Local repository: A repository stored on your local machine, where you can make changes, commit them, and view the project history.   
- Remote repository: A repository hosted on a server (like GitHub, GitLab, or Bitbucket), accessible by multiple users. It allows collaboration by enabling others to push and pull changes.

1. What are the basic Git commands?

- git init: Initializes a new Git repository.

- git clone <repository\_url>: Creates a local copy of a remote repository.

- git add <file>: Stages changes to be committed.

- git commit -m "message": Commits staged changes with a message describing the changes.

- git status: Shows the current status of the working directory and staged files.

- git push: Pushes local commits to a remote repository.

- git pull: Pulls changes from a remote repository to the local repository.

- git branch: Lists branches in the repository.

- git checkout <branch>: Switches between branches.

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

* Use the command: “git status”.

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

- Branches allow multiple developers to work on different features or fixes simultaneously without interfering with the main project. The purpose is to isolate changes, making it easier to manage different tasks independently. To create a new branch: “git branch <branch\_name>”. To switch to an existing branch: “git checkout <branch\_name>”.

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

- GitHub: A cloud-based platform for hosting Git repositories, enabling collaboration and sharing of code. It provides tools like issue tracking, pull requests, and project boards.

- GitLab Desktop: GitLab is similar to GitHub but offers a self-hosted or cloud-hosted platform with additional features like continuous integration (CI/CD), security scanning, and issue tracking.

- Git: A version control system used to manage and track changes in code. Git is a tool that works in the background, while GitHub and GitLab are platforms built around Git to offer collaborative features, hosting, and project management tools.

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

- Create a Remote Repository: Set up a new repository on GitLab or GitHub. Copy its URL.

- Initialize Local Repository: Run “git init” in your project folder (if not done already).

- Add and Commit Files: Stage changes with git add . and commit using git commit -m "Initial commit".

- Connect to Remote: Add the remote repository URL using: :git remote add origin <remote-repository-URL>”

- Push to Remote: Push local changes with: “git branch -M main” “git push -u origin main”

1. What are the steps to collaborate with others using GitLab or GitHub?
2. Clone the repository: Get a local copy of the remote repository. git clone <repository\_url>
3. Create a new branch for your changes. git checkout -b <branch\_name>
4. Make changes and commit them locally. git add <file> git commit -m "description of changes"
5. Push your changes to the remote repository. git push origin <branch\_name>
6. Create a pull request (GitHub) or a merge request (GitLab) to request the team to review and merge your changes.
7. Review and merge: After reviewing, the changes are merged into the main branch.
8. How do you resolve merge conflicts in Git?
9. Pull the latest changes from the remote repository: git pull origin <branch\_name>
10. If a conflict arises, Git marks the conflicted files. Open these files to see the conflicting sections.
11. Manually resolve the conflict by choosing one version of the changes or combining them.
12. After resolving the conflict, stage the file: git add <file>
13. Commit the resolved conflict: git commit -m "Resolved merge conflict"
14. Push the resolved changes back to the remote.
15. What is a pull request, and why is it used in GitHub?

- A pull request (PR) is a way to propose changes to a project in GitHub. It allows the author of a branch to request that their changes be reviewed and merged into the main branch. PRs are used for code review, discussion, and ensuring that changes are properly vetted before being incorporated into the project.

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

- Best practices for writing commit messages:

* **Use the imperative mood**: Write commit messages as if you're giving a command
* **Be concise but descriptive**: The message should clearly describe the change in 50-72 characters.
* **Use the body for more details**: If necessary, use the body of the message to explain why the change was made.
* **Reference issues**: If your commit addresses an issue, reference it (e.g., "Fixes #5")
* **Avoid vague messages** like "Update" or "Fix stuff”. Be specific.