RWANDA POLYTECHNIC

NGOMA COLLEGE

GROUP 1: GIT AND GITHUB

MODULE CODE AND TITLE: ITLDO801- DEVELOPMENT OPERATIONS

LECTURE NAME: Mr. BONAVENTURE UWAMAHORO

RQF LEVEL: 8

PROGRAM: INFORMATION TECHNOLOGY

Group names & Regno: 1. Vincent SHYAKA 24RP03809

2. Steven BANANIYE 24RP04326

3. Claudine MUSHIMIYIMANA 23RP00498

4. Jean Baptiste NDUWAYEZU 23RP00470

5. Patrice TUYIZERE 23RP00987

6. Anitha MUTESI 24RP14272

1. Git: A Version Control System

Git is an open-source distributed version control system created by Linus Torvalds in 2005. It is designed to handle everything from small to very large projects with speed and efficiency.

Git helps developers track changes in their codebase, collaborate with others, and manage versions of their project files.

Importance of using Git

Collaboration: Enables multiple developers to work simultaneously.

History: Provides a detailed history of project changes.

Branching: Supports branching and merging for feature development

Key Features of Git:

- **Version Tracking:** Git tracks changes made to files over time, allowing users to revert to previous versions if needed.
- **Branching and Merging:** Developers can create branches to work on specific features or fixes independently and later merge them into the main branch.
- **Distributed Nature:** Each developer has a full copy of the repository, enabling offline access and ensuring redundancy.
- **Speed and Efficiency:** Git performs operations like commits, diffs, and merges locally, making it faster than centralized systems.

Basic Git Commands:

- git init: Initializes a new Git repository.
- git add: Stages changes to be committed.
- git commit: Saves staged changes to the repository with a message.
- git push: Uploads local repository changes to a remote repository.
- git pull: Fetches and integrates changes from a remote repository.
- git clone: Copies a remote repository to your local machine.

1. **Setting Up Git**

- **Installation:** Download and install Git based on your operating system (Windows, macOS, or Linux).
- **Configuration**

git config --global user.name "Your Name"

git config --global user.email your_email@example.com

2. **Verify Configuration:** git config --list
3. **Initializing a Repository**

Use **git init** to start tracking a project in the current directory, turning it into a Git repository.

Example: cd my-project, git init

4. **Managing Changes**

- Check Status**:** Use **git status** to see which files have changed.
- Listing file: **list**
- Deleting file **: git rm <file name>**
- Staging Changes**:** Add specific files to the staging area with **git add [file]** or all changes with **git add.**
- Committing Changes**:** Record changes in the repository with: **git commit -m "Descriptive message"**

5. **Remote Repositories**

- Cloning: Copy a remote repository to your local system with: **git clone [repository URL]**
- git remote add origin : Used to **link** a local Git repository to a remote repository.
- Pushing Changes**:** Upload local commits to the remote repository: **git push origin main**
- Pulling Changes**:** Synchronize your local repository with updates from the remote: **git pull**

6. **Branching and Merging**

- Branching: Create separate branches for new features or changes:

Create a Branch: **git branch feature-branch**

Switch to a Branch: **git checkout feature-branch**

Combine these steps**: git checkout -b feature-branch**

- **Merging:** Integrate changes from one branch into another:

git checkout main

git merge feature-branch

- **Deleting a branch:** git branch -d <branch_name>

7. **Stashing Changes**

Temporarily save unfinished work with: **git stash**

apply the stashed changes: **git stash apply**

8. **Advanced Features**

Tags: Use tags to mark specific points in history, such as a version release:

git tag -a v1.0 -m "Release version 1.0"

git push origin v1.0

- **SSH Setup:** Generate and add SSH keys for secure access to GitHub repositories.

  ssh-keygen -t ed25519 -C your_email@example.com

- Start the ssh-agent in the background.

  eval "$(ssh-agent -s)"

- Add your SSH private key to the ssh-agent.

  ssh-add ~/.ssh/id_ed25519

  Adding a new SSH key to your GitHub account

  Copy the SSH public key to your clipboard.

  cat ~/.ssh/id_ed25519.pub

Git revert

Undo a specific commit by creating a new commit.

How it Works

- Does **not** remove history; instead, it creates a new commit that reverses the changes of a previous commit.
- Safe to use in public/shared branches.

Git Reset

Reset is the command we use when we want to move the repository back to a previous commit, discarding any changes made after that commit.

2. GitHub: A Platform for Collaboration

GitHub is a web-based hosting service for Git repositories. It provides a graphical interface, collaboration tools, and additional features like issue tracking, project management, and code review. GitHub makes it easier for developers to share their work, collaborate with others, and manage projects.

Key Features of GitHub:

- **Remote Hosting:** GitHub acts as a remote repository, enabling developers to share code and collaborate.
- **Collaboration Tools:** Users can work together on repositories, track issues, and discuss changes using pull requests.
- **Version Control Visualization:** GitHub provides an intuitive interface to visualize commits, branches, and changes.
- **Integration with Tools:** GitHub integrates with CI/CD pipelines, testing frameworks, and third-party services.
- **Open Source Projects:** Many open-source projects are hosted on GitHub, making it a hub for learning and contribution.

Advantages of GitHub

1. Version Control: Tracks changes in code, allowing developers to manage and revert to earlier versions easily.

2. Collaboration and Teamwork: Facilitates collaboration among multiple developers using features like branches, pull requests, and code reviews.

3. Open Source Contributions: Provides a platform for sharing and contributing to open-source projects, fostering innovation and learning.

4. Integration with Tools: Seamlessly integrates with CI/CD pipelines, project management tools, and third-party applications like Jenkins, Jira, and Slack.

5. Accessibility and Hosting: Provides a centralized location for remote repository hosting, accessible from anywhere with an internet connection.

6. Documentation Support: Supports README files, wikis, and other forms of documentation, improving project clarity and usability.

7. Portfolio Building: Acts as an online portfolio for developers, showcasing skills and projects to potential employers or collaborators.

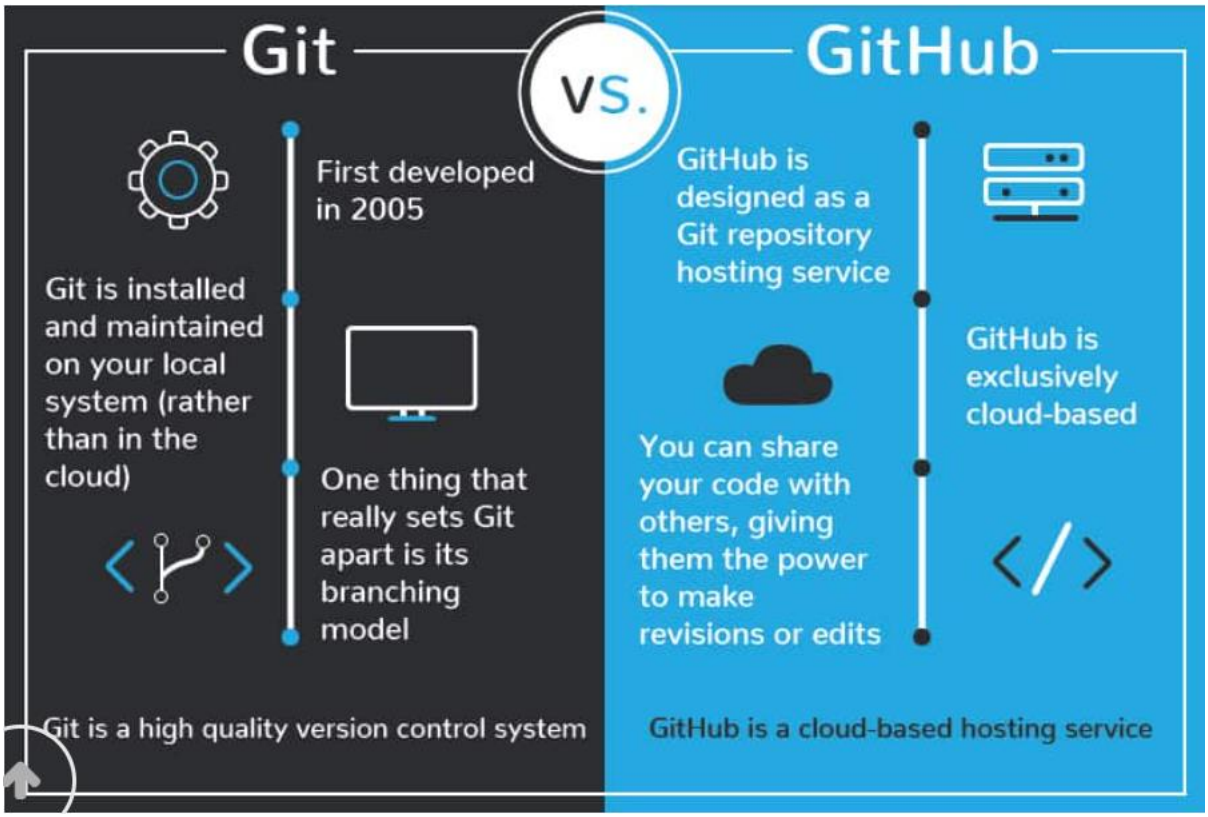
Disadvantages of GitHub

1. Cost for Private Repositories: While public repositories are free, some advanced features and private repositories require paid plans for organizations.

2. Internet Dependency: Requires an active internet connection for most interactions, limiting usability in offline scenarios.

3. Limited Large File Support: GitHub is not optimized for handling large files or repositories, and additional tools like Git LFS may be required.

The difference between Git and GitHub



Feature	Git	GitHub
Type	Version control system (software/tool).	Cloud-based platform/service for hosting Git repositories.
Purpose	Tracks changes in files, manages versions, and facilitates collaboration.	Allows sharing, collaboration, and remote management of Git repositories.
Functionality	Works locally on your computer.	Works online to host repositories and provide collaboration tools.
Internet Requirement	Can work offline.	Requires an internet connection for accessing and managing repositories.
Key Features	Version control, branching, merging, and tracking file history.	Hosting Git repositories, pull requests, issue tracking, and team collaboration tools.
Usage	Operated through the command line or tools like Git GUI.	Accessible via a web interface, desktop app, or integration with IDEs.
Scope	Focuses on version control.	Focuses on repository hosting, team collaboration, and project management.
Examples	git init, git add, git commit, git merge, etc.	Create repositories, manage pull requests, review code, and collaborate on projects.

Git GitHub Flow

Working using the GitHub Flow

The GitHub flow is a workflow designed to work well with Git and GitHub.

It focuses on branching and makes it possible for teams to experiment freely, and make deployments regularly.

The GitHub flow works like this:

- Create a new Branch
- Make changes and add Commits
- Open a Pull Request
- Review
- Deploy
- Merge

Open a Pull Request

Pull requests are a key part of GitHub. A Pull Request notifies people you have changes ready for them to consider or review.

When a Pull Request is made, it can be reviewed by whoever has the proper access to the branch. This is where good discussions and review of the changes happen

If you receive feedback and continue to improve your changes, you can push your changes with new commits, making further reviews possible.

Deploy

When the pull request has been reviewed and everything looks good, it is time for the final testing. GitHub allows you to deploy from a branch for final testing in production before merging with the master branch.

Fork a Repository

A fork is a copy of a repository. This is useful when you want to contribute to someone else's project or start your own project based on theirs.

Fork is not a command in Git, but something offered in GitHub and other repository hosts.