RWANDA POLYTECNIC

NGOMA COLLEGE

B-TECH IT

MODULE: DevOps

ASSIGNMENT OF DevOps

TOPIC: GIT AND GITHUB

DATE: 18 January 2025

**GROUP MEMBERS**

SHYAKA VINCENT 24RP03809

BANANIYE STEVEN 24RP04326

MUSHIMIYIMANA CLAUDINE 23RP00498

NDUWAYEZU JEAN BAPTISTE 23RP00470

PATRICE TUYIZERE 23RP00987

MUTESI ANITHA 24RP14272

INTRODUCTION

**What is Git ? what is GitHub ?**

1. Git is a version control system (VCS) that is used for a file to:

* Tracking code changes
* Tracking who made changes
* Coding collaboration

What does Git do?

* Manage projects with **Repositories**
* **Clone** a project to work on a local copy
* Control and track changes with **Staging** and **Committing**
* **Branch** and **Merge** to allow for work on different parts and versions of a project
* **Pull** the latest version of the project to a local copy
* **Push** local updates to the main project

### Working with Git

* Initialize Git on a folder, making it a **Repository**
* Git now creates a hidden folder to keep track of changes in that folder
* When a file is changed, added or deleted, it is considered **modified**
* You select the modified files you want to **Stage**
* The **Staged** files are **Committed**, which prompts Git to store a **permanent** snapshot of the files
* Git allows you to see the full history of every commit.
* You can revert back to any previous commit.
* Git does not store a separate copy of every file in every commit, but keeps track of changes made in each commit!

### Why Git?

* Over 70% of developers use Git!
* Developers can work together from anywhere in the world.
* Developers can see the full history of the project.
* Developers can revert to earlier versions of a project.

## **Git Staging Environment**

One of the core functions of Git is the concepts of the Staging Environment, and the Commit.

As you are working, you may be adding, editing and removing files. But whenever you hit a milestone or finish a part of the work, you should add the files to a Staging Environment.

**Staged** files are files that are ready to be **committed** to the repository you are working on. You will learn more about commit shortly.

## **Git Commit**

Since we have finished our work, we are ready move from stage to commit for our repo.

Adding commits keep track of our progress and changes as we work. Git considers each commit change point or "save point". It is a point in the project you can go back to if you find a bug, or want to make a change.

When we commit, we should **always** include a **message**.

By adding clear messages to each commit, it is easy for yourself (and others) to see what has changed and when.

## **Git Help**

If you are having trouble remembering commands or options for commands, you can use Git help.

There are a couple of different ways you can use the help command in command line:

* git command -help -  See all the available options for the specific command
* git help --all -  See all possible commands

# **Git Branch**

## **Working with Git Branches**

In Git, a branch is a new/separate version of the main repository.

Let's say you have a large project, and you need to update the design on it.

How would that work without and with Git:

Without Git:

* Make copies of all the relevant files to avoid impacting the live version
* Start working with the design and find that code depend on code in other files, that also need to be changed!
* Make copies of the dependant files as well. Making sure that every file dependency references the correct file name
* EMERGENCY! There is an unrelated error somewhere else in the project that needs to be fixed ASAP!
* Save all your files, making a note of the names of the copies you were working on
* Work on the unrelated error and update the code to fix it
* Go back to the design, and finish the work there
* Copy the code or rename the files, so the updated design is on the live version
* (2 weeks later, you realize that the unrelated error was not fixed in the new design version because you copied the files before the fix)

With Git:

* With a new branch called new-design, edit the code directly without impacting the main branch
* EMERGENCY! There is an unrelated error somewhere else in the project that needs to be fixed ASAP!
* Create a new branch from the main project called small-error-fix
* Fix the unrelated error and merge the small-error-fix branch with the main branch
* You go back to the new-design branch, and finish the work there
* Merge the new-design branch with main (getting alerted to the small error fix that you were missing)

Branches allow you to work on different parts of a project without impacting the main branch.

When the work is complete, a branch can be merged with the main project.

You can even switch between branches and work on different projects without them interfering with each other.

Branching in Git is very lightweight and fast!

## **New Git Branch**

Let add some new features to our index.html page.

We are working in our local repository, and we do not want to disturb or possibly wreck the main project.

So we create a new branch:

## **Merge Branches**

We have the emergency fix ready, and so let's merge the master and emergency-fix branches.

First, we need to change to the master branch:

### Example

git checkout master

Switched to branch 'master'

Now we merge the current branch (master) with emergency-fix:

### Example

git merge emergency-fix

Updating 09f4acd..dfa79db

Fast-forward

index.html | 2 +-

1 file changed, 1 insertion(+), 1 deletion(-)

Since the emergency-fix branch came directly from master, and no other changes had been made to master while we were working, Git sees this as a continuation of master. So it can "Fast-forward", just pointing both master and emergency-fix to the same commit.

As master and emergency-fix are essentially the same now, we can delete emergency-fix, as it is no longer needed:

### Example

git branch -d emergency-fix

Deleted branch emergency-fix (was dfa79db).

## **Merge Conflict**

Now we can move over to hello-world-images and keep working. Add another image file (img\_hello\_git.jpg) and change index.html, so it shows it:

### Example

git checkout hello-world-images

Switched to branch 'hello-world-images'

### What is GitHub?

* Git is not the same as GitHub.
* GitHub makes tools that use Git.
* GitHub is the largest host of source code in the world, and has been owned by Microsoft since 2018.

**GitHub** is a web based platform that allows developers to create, store, manage, and share their codes. It uses Git, an open source version control system that allows multiple people to work on codes at the same time.