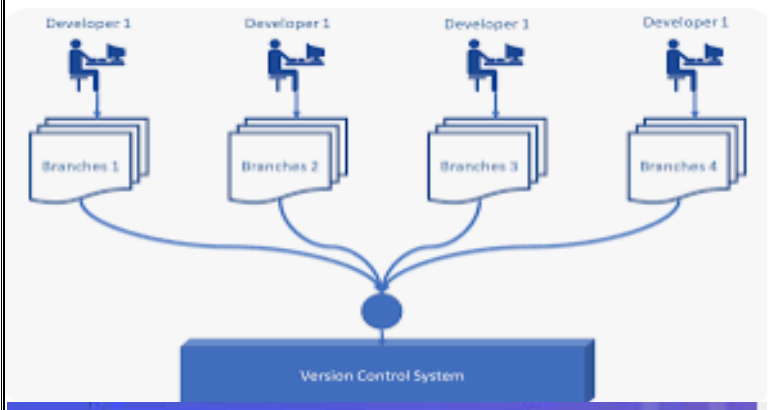
INITIATIVES DON BOSCO
INSTITUT DON BOSCO KABARONDO TSS
TEL: 0788670647/0790909669/0790909612
B.P: 50 Kibungo
Email: dbtikabarondo@gmail.com



SWDVC301. VERSION CONTROL

Competence: CONDUCT VERSION CONTROL

SECTOR: ICT

TRADE: SOFTWARE DEVELOPMENT

MODULE TYPE: SPECIFIC

CREDIT: 7

RQF LEVEL: 3

LEARNING HOURS: 70

Purpose statement: This specific module describes the skills required to apply version control during software development in a team. Upon completion of this module, the learner will be able to: Setup repository, Manipulate files, Ship codes

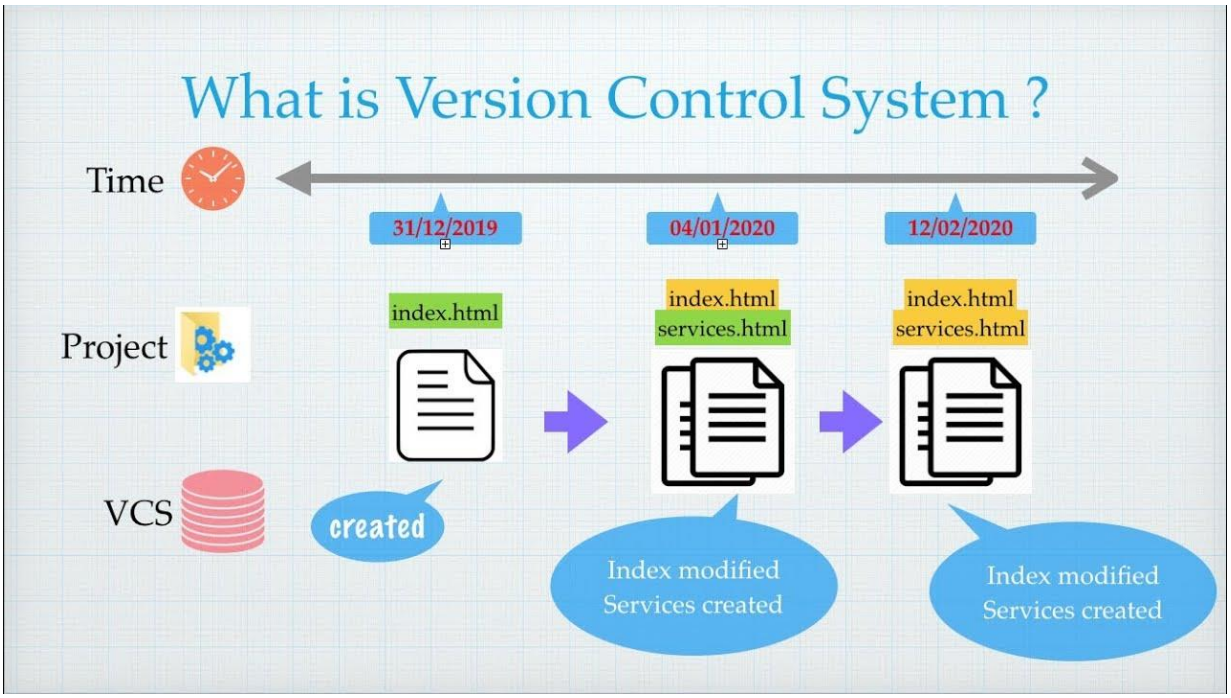
Table of Contents

LEARNING OUTCOME 1: SETUP REPOSITORY

Indicative content 1. Definition of general key terms

Topic1. Version control

It **is a system that allows developers to manage changes to code, documents, and other files over time**. It enables collaboration among team members, facilitates the tracking of changes, and ensures the integrity of code and other files over time.



A repository **is an essential location for storing and managing data or files**.

In the realm of version control systems like **Git**,

A repository **refers to a directory or folder located on a computer or server that contains all the files and folders related to a project**.

It also includes version control that tracks changes to these files over time.

The repository enables multiple individuals to collaborate on the same project simultaneously by providing a means to merge changes made by different contributors into a unified and coherent version of the project.

Repositories can be either **locally hosted** on a computer or **remotely hosted on a server**, such as **GitHub** or **GitLab**.

As the primary component of version control systems, the repository is vital in effectively managing and organizing project files.

Locally hosted on a computer means that **the software or application is installed and runs on a computer's local hard drive or network, and is only accessible on that computer or within the local network**. This means that the software is physically present on the user's machine, and any data or changes made to it are stored locally.

Remotely hosted on a server means that the **software or application is installed and runs on a remote server, which can be accessed over the internet from any device with an internet connection**. This means that the software is not physically present on the user's machine, and any data or changes made to it are stored remotely on the server.

In the case of GitHub or GitLab, these are online platforms for hosting and managing source code repositories.

Users can remotely host their code on these platforms, making it easy for them to collaborate with other developers and to access their code from anywhere with an internet connection

Topic2. Git

Which was created by Linus Torvalds in 2005, is a popular distributed **version control system utilized for tracking changes in files and collaborating on software projects**. It has emerged as one of the most commonly used version control systems in the software development industry.

Git can track changes to a wide variety of file types, including:

- **Source code files** (e.g., C++, Java, Python)
- **Web pages and web application code** (e.g., HTML, CSS, JavaScript)
- **Server-side scripting language code** (e.g., PHP, Ruby on Rails, Node.js)

- **Configuration files** (e.g., YAML, JSON, XML)
- **Database scripts and schema definitions** (e.g., SQL, NoSQL)
- **Documentation and text files** (e.g., README.md, LICENSE.txt)

In general, Git is designed to be a flexible and extensible version control system that can track changes to almost any type of file. This makes it well-suited for managing and collaborating on software projects of all types and sizes.

Where to find:

You can find Git on the official Git **website at https://git-scm.com/.**

The website provides links to download Git for various operating systems, including **Windows**, **macOS**, and **Linux**

Topic3. GitHub

Is a web-based platform that provides hosting for software development projects using the Git version control system. It allows developers to collaborate on code and provides features such as version control, code review, issue tracking, and project management.

Developers can use GitHub **to store** and **manage** their source code **repositories**, track changes to code over time, and collaborate with other developers on projects.

GitHub also offers a range of features that facilitate team communication, such as pull requests and code reviews.

GitHub is widely used by individuals and organizations, including open-source projects, startups, and large companies. It is also used by many developers to showcase their work and build their personal brand in the developer community.

Web-based platform

Heroku

Netlify

Azure App Service

Google Cloud Platform

Step to run project on these platform

The steps to run a project on various platforms such as **Heroku**, **Azure**, and **Google Cloud Platform** are generally similar.

Here are the general steps for deploying a project to these platforms:

1. **Create an account and log in to the platform you want to use**.
2. **Create a new application or project on the platform**.

3. **Set up your project environment, including installing any necessary dependencies, configuring your application settings, and setting up your database**.
4. **Create a production build of your project. This usually involves running a build script or command that generates the necessary files and assets for your application to run**.
5. **Deploy your project to the platform using the platform's deployment tools or command line interface (CLI).**
6. **You may need to connect your local Git repository to the platform's Git repository in order to deploy your project**.
7. **Monitor your application's logs and metrics to ensure it is running correctly on the platform**

Topic4. Terminal

In computing, a terminal (also known as a **console** or **command line interface**) is a text-based interface **that allows users to interact with the operating system and execute commands.** It is typically accessed through a program called a terminal emulator, which **runs on a computer** and **communicates with the operating system through a command shell**.

Some popular terminal emulators include macOS Terminal, Windows Command Prompt, and Linux Terminal (which includes many different terminal programs such as Bash, Zsh, and Fish). Many integrated development environments (IDEs) also include a built-in terminal for convenience.

In this course, we use the terminal to **interact with Git** and to perform various commands and operations related to version control. The terminal provides a command-line interface that allows **us to execute Git commands**, **navigate through directories**, and **create**, **modify,** and **delete** files and **directories**.

To change directory, make a directory, create a file, and open it in a terminal, you can follow these steps:

Change directory:

Use the cd command followed by the directory path to navigate to the directory **where you want to create the new directory and file**. For example, if you want to create the directory and file in your home directory, you can use the following command:

cd ~

Make directory:

Use the **mkdir** command followed by the directory name **to create a new directory**. For example, to create a directory called "**MyFolder**", you can use the following command:

mkdir MyFolder

Change directory:

Navigate into the new directory you just **created using the cd command**. For example:

cd MyFolder

Create file:

Use the touch command followed by the file name to create a new file. For example, to create a file called "**MyFile.txt**", you can use the following command:

touch MyFile.txt

Open file:

Use a text editor to open the file. There are many text editors available for use in the terminal, such as nano or vim. For example, to open the file using nano, use the following command:

nano MyFile.txt

This will open the file in the nano editor, where you can edit and save it.

- Indicative content 2. Introduction to version control

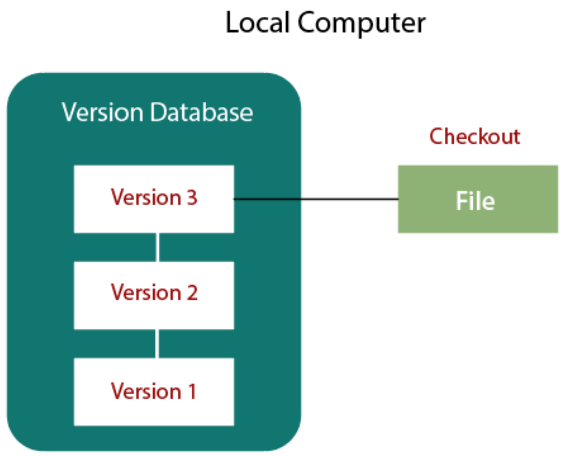
Topic1. Types of version control

Version control, also known as revision control or source control, **is a system that allows developers to manage changes to code, documents, and other files over time.**

It is an essential tool for software development, enabling collaboration among team members, and facilitating the tracking of changes to code and other files.

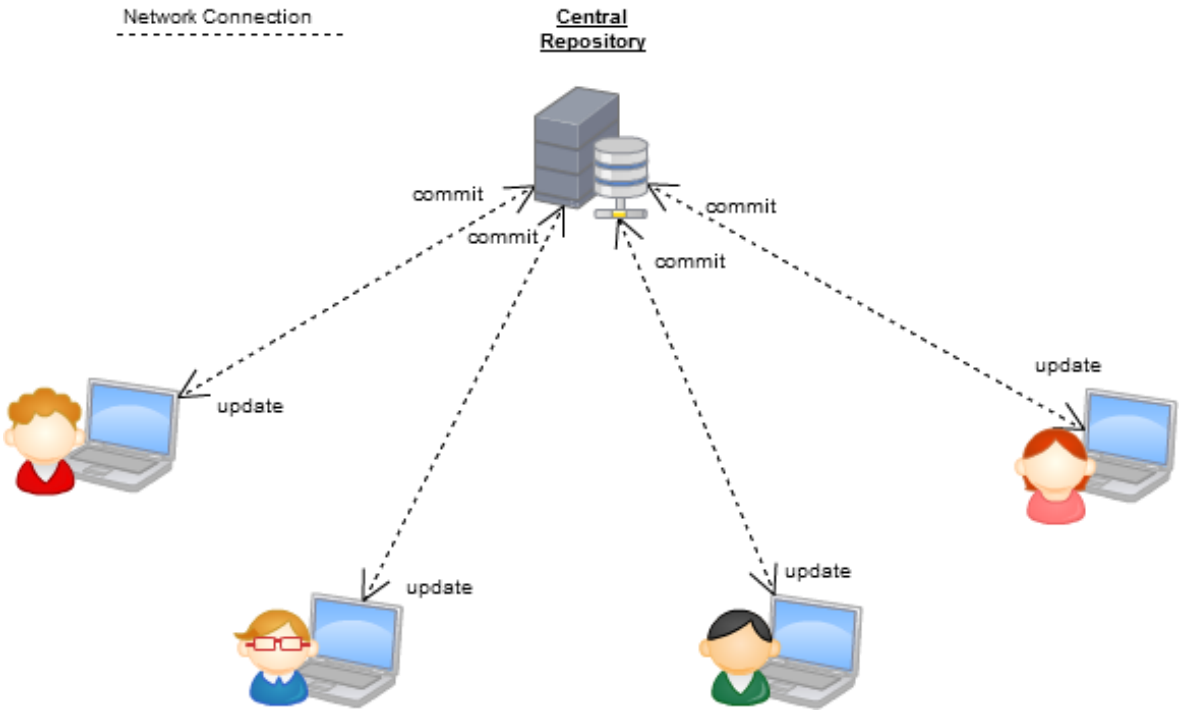
There are three main types of version control systems:

1. Local version control: **This is the simplest form of version control, where changes are tracked on a local machine**. Developers make changes to files on their own computer, and the version control system keeps track of the changes. This type of version control is limited to individual developers and is not suitable for collaborative work.

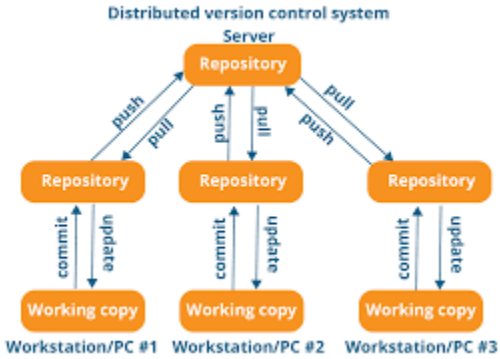


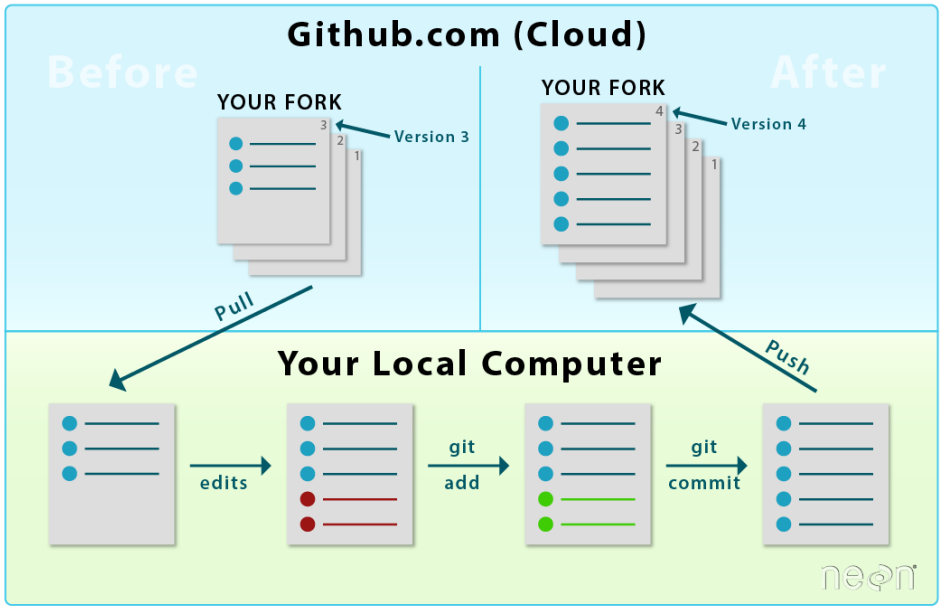
2. **A centralized version control**: **system involves all changes being made to a central repository that is stored on a remote server**. Developers check out files from the repository, make changes locally, and then check the changes back into the repository. **This system promotes collaboration among team**

members and establishes a centralized source of truth for the codebase, which enhances code organization and maintainability..



3. **Distributed version control**: **systems allow each developer to have their own local copy of the entire codebase, including the full history of changes**.
   This enables developers to work independently and facilitates the management of changes in a distributed team.
   Git is a widely used distributed version control system in software development that leverages this approach.





Overall, version control systems are essential tools for software development, enabling collaboration among team members, facilitating the tracking of changes, and ensuring the integrity of code and other files over time.

Topic2. Well Known version control system

Git, **CVS** (Concurrent Version System), **Mercurial**, and **SVN** (Subversion) are all examples of version control systems. They are used to manage changes to source code and other files over time, allowing multiple people to work on the same codebase simultaneously without conflicts.

Git is currently the most popular version control system and is widely **used for software development. It** was created **by Linus Torvalds in 2005** and **is a distributed version control system**,

Which means that each developer has a copy of the entire codebase on their local machine. This makes it fast and efficient, and also allows developers to work on the codebase even when they are offline.

CVS (Concurrent Version System) **is an older version control system that was popular in the 1990s and early 2000s**. It is a centralized system, which means that all code changes are managed on a central server. It is still used by some open source projects, but is largely considered outdated.

Mercurial is another **distributed version control system that is similar to Git**, but is **less popular. It was created in 2005 by Matt Mackall** and is widely **used in the Python community.**

SVN (Subversion) is a centralized version control system that is still used in some organizations. It was created in 2000 by the Apache Software Foundation and **is**

known for its simplicity and ease of use. However, it is considered less powerful than Git and other distributed version control systems.

ASSESSMENT 1

Question1. What is version control and why is it important in software development?

Question2. What is a repository and how is it used in version control systems like Git?

Question3 .what is the difference between locally hosted and remotely hosted repositories?

Question4 What is Git and what kind of files can it track changes to?

Question5 Where can Git be found and downloaded?

Question6 What is GitHub and what features does it offer for software development?

Question7 What are some examples of other web-based platforms similar to GitHub?

Question8 What is a terminal and what is its role in interacting with Git?

Question9 How do you change directory, make a directory, create a file, and open it in a terminal?

Question10 What are some popular text editors used in the terminal for editing files?

Topic3. Benefits of Version Control:

Collaboration: **Version control allows multiple developers to work on the same project simultaneously without interfering with each other's work**.

They can make changes to their own copies of the code and merge them together seamlessly.

Track changes: Version control **helps in tracking changes made to the code**, including **who made the changes**, **when they were made**, and **why they were made**.

This provides an audit trail that can be used to troubleshoot issues and ensure accountability.

Backup and Recovery: Version control **systems keep a copy of each change made to the code**, allowing **developers to recover earlier versions of the code if necessary**. This helps in protecting the code from data loss or damage.

Experimentation or Investigation, Exploration, Research, Analysis, Examination and Study: Version control systems **allow developers to create new branches (refers to a copy of the source code in a repository that can be developed and tested independently from the main codebase) to experiment with new ideas without affecting the main codebase (codebase is the complete set of source code that comprises an application or software system)**.

They can make changes to the branch and merge it with the main codebase if the changes are successful.

Code review: Version control systems **provide a platform for code review**, **where other developers can review the changes made to the code before they are merged into the main codebase**. This helps in ensuring the quality and stability of the code.

Ass

Question 1. What is the benefit of version control for collaboration among multiple developers?

Question 2. How does version control help in tracking changes made to the code?

Question 3. How does version control enable developers to experiment with new ideas without affecting the main codebase?

- Indicative content 3. Description of git

Git architGit is a distributed version control system, which means that each **user has a copy of the entire repository on their local machine**.

This allows developers to work on their own copy of the code and then merge their changes with the changes made by other developers,

Which is known as a distributed workflow.

Topic1. Git Basic concept

Git has a number of **basic concepts that developers need to be familiar with in order to use it effectively**. These include:

Repository: A repository **is a collection of files and directories that make up a project.**

Each repository has its own history and version control.

Commit: A commit **is a snapshot of the repository at a specific point in time**.

It records **changes made to the files and directories in the repository**, along with a **message describing the changes**.

Branch: A branch **is a copy of the repository that can be developed** and **tested independently from the main codebase**. Developers can create new branches to experiment with new ideas without affecting the main codebase.

Merge: Merging is the **process of combining changes from one branch into another**. It allows developers to bring changes made on one branch into the main codebase.

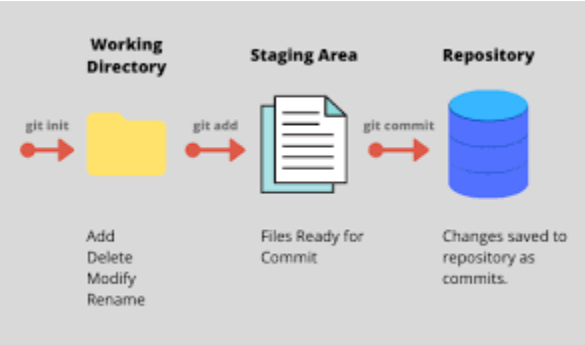
Topic2. Git architecture

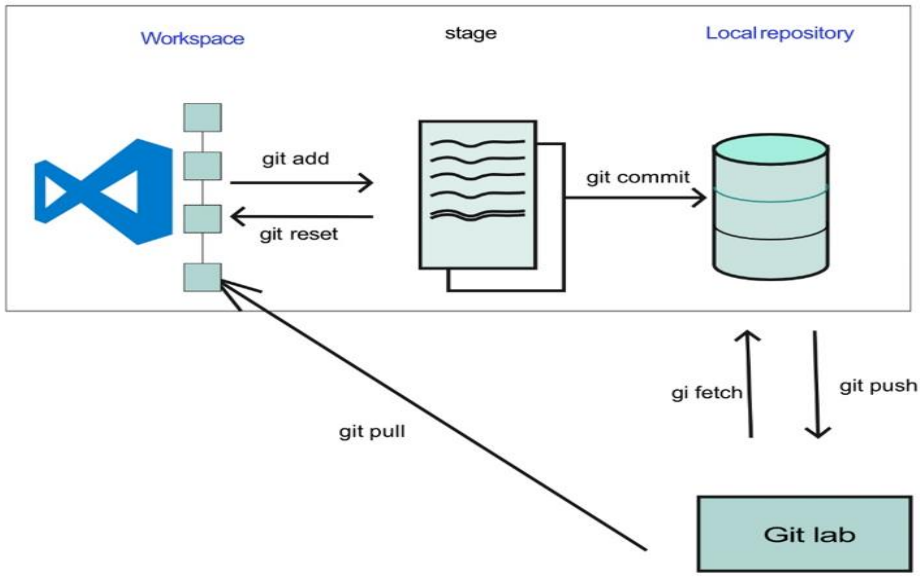
Git architecture consists of three main components:

Working directory: This is **where the actual files are stored** and **where developers make changes.**

Staging area: This is an intermediate **area where changes are prepared before being committed to the repository**.

Repository: This is **where the committed changes are stored**.





Topic3. Git workflow

Git workflow typically involves the following steps:

Create a new branch: Developers **create a new branch to make changes to the code without affecting the main codebase**.

Make changes: Developers **make changes to the code in their branch**.

Commit changes: Developers **commit their changes to their branch**.

Merge changes: Developers **merge their changes into the main codebase.**

Resolve conflicts: If there **are conflicts between the changes made by different developers, they need to be resolved before** the changes can be merged into the main codebase.

Test and deploy: Once the changes **have been merged into the main codebase, they are tested and deployed**.

ASSESSMENT 2

Question 1. What is the main benefit of Git's distributed workflow?

Answer: The main benefit of Git's distributed workflow **is that each user has a copy of the entire repository on their local machine**.

This allows **developers to work on their own copy of the code** and **then merge their changes with the changes made by other developers**, without relying on a central server. This makes collaboration more flexible and decentralized, and allows for faster and more efficient development.

Question 2. What is a commit in Git, and what information does it record?

Question 3. What are the three main components of Git's architecture, and what is the purpose of each one?

Question 4. What are the typical steps involved in Git workflow, from creating a new branch to deploying changes?

Topic 4. Initialization of Git

Refers **to the process of creating a new Git repository in a local** or **remote directory**. **It sets up all the necessary files and data structures required for version control** and **enables tracking of changes made to the files** in that directory

Initialisation of Git:

To initialize Git in a project directory, navigate to the directory in your terminal or command prompt and run the following command:

git init

This will initialize an empty Git repository in the current directory

Topic4.1 Terminal Basic commands:

Here are some basic commands that you can use in the terminal or command prompt:

cd: Change directory

ls or dir: List the contents of the current directory

mkdir: Create a new directory

rm or del: Remove a file or directory

cp or copy: Copy a file or directory

mv or move: Move or rename a file or directory

Topic 4.2. Installation of Git Setup:

You can download Git for your operating system from the official Git website: **https://git-scm.com/downloads**

Once you have downloaded the installer, run it and follow the prompts to install Git on your system.

After installation, **you can access Git from the terminal** or **command prompt by typing git f**ollowed by the command you want to use.

Topic5. Configure Git

Topic5. 1. Git init

It **involves setting up your name**, **email**, and **other preferences that Git will use when you interact with it**.

To initialize Git in a repository, you can use the

git init command in your terminal. This creates a new Git repository in your current working directory.

Topic5. 2. Git configuration

To configure your Git user information, you can use the **git config command**

With the **--global flag to set global settings for all repositories** on your machine, or without the flag to set settings only for the current repository. For example:

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

git config --global user.email "**your.email@example.com**"

You can also check your Git version using the **git --version command in your terminal**.

ASSESSMENT 3

Question1. What is the purpose of initializing Git in a project directory?

Question2. How do you initialize Git in a project directory?

Question3. What are some basic terminal commands that can be used in Git fom marking to directory and create new one ?

Question4. Where can Git be downloaded for your operating system?

Question5. How do you configure your Git user information, and what command can you use to check your Git version?

Question6. What is the purpose of the --global flag in the git config command?

.....

Topic 5.3. Git –version command

You can check your current version of Git by running the git --version command in a terminal (Linux, macOS) or command prompt (Windows). If you don't see a supported version of Git, you'll need to either upgrade Git or perform a fresh install, as described below.

Topic 6. Configure .gitignore file

The .gitignore : file is used to tell Git which files and directories to ignore when you make a commit. This is useful for preventing files such as log files, build artifacts, and other temporary files from being added to your repository.

Topic 7. To configure a .gitignore file you can follow these steps

1.Create a new file in your repository called .**gitignore.**

2.Open the file in a text editor.

3.Add the names of files and directories that you want Git to ignore, each on a new line.

4.Save the file.

For example, if you want to ignore all files with the .log extension and the build/ directory, you can add the following to your .gitignore file:

```
*.log
build/
```

Indicative content 4. Use of GitHub repository:

Topic 1. Description of github

GitHub is a web-based **platform that provides hosting for software development and version control using Git**. It allows **developers to collaborate on projects**, share code, and track changes made to the code.

Topic 2. Creating an account on GitHub:

To use GitHub, you **need to create an account on the website**.

You can sign up for a free account by visiting the GitHub website and following the instructions provided.

Topic 3. Creating a new remote repository:

To create a new remote repository on GitHub, you can follow these steps:

1. **Log in to your GitHub account**.
2. **Click on the "+" icon in the top right corner of the page**.
3. **Select "New repository" from the dropdown menu**.
4. **Enter a name for your repository and select any additional options you need**.
5. **Click on the "Create repository" button to create your new remote repository**.
6. **Applying Git commands related to repositories**:

Once you have created a new remote repository on GitHub, you can use various Git commands to interact with it. Some common Git commands related to repositories include

Topic 4. Applying Git commands related to repositories:

1. **git clone**: This command allows you to create a local copy of a remote repository on your machine.
2. **git remote:** This command allows you to view and manage remote repositories, including adding or removing remote repositories.

3. **Git clone:** The "**git clone**" command allows you to create a local copy of a remote repository on your machine. To use this command, you need to know the URL of the remote repository you want to clone.

For example:

git clone https://github.com/username/repo.git

This command will create a new directory on your machine with the name of the repository and copy all the files and directories from the remote repository to your local machine.

Git remote:

The "**git remote**" command **allows you to view** and **manage remote repositories**. Some common use cases for this command include:

Adding a new remote repository: git remote add <name> <url>

Removing a remote repository: git remote remove <name>

Renaming a remote repository: git remote rename <old-name> <new-name>

Viewing a list of remote repositories: git remote –v

ASSESSMENT 3

Question1. What is the purpose of the .gitignore file?

Question2. How do you create a .gitignore file?

Question3. Can you provide an example of what to add to a .gitignore file?

Question4. What is GitHub and what does it provide for software development?

Question5. How do you create a new remote repository on GitHub?

Question5. What are some common Git commands related to repositories and what do they do?

Answers

git init: Initializes a **new Git repository** in your local directory.

git clone: Creates a **local copy of a remote repository** on your machine.

git add: Adds changes made to a file or files to the staging area.

git commit: Commits the **changes made to a file** or files **to the local repository**.

git push: Pushes the committed **changes from the local repository to a remote repository**.

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

git status: Shows the **status of the local repository, including which files have been modified or staged.**

git log: Displays the **commit history of the local repository**.

git branch: Lists, **creates**, or **deletes** branches in the local repository.

git merge: Merges changes **made in one branch of the local repository with another branch.**

git remote: **Manages remote repositories**, including adding, removing, and renaming them.

git stash: Saves changes made to the working directory without committing them, allowing you to switch to another branch or work on another feature without losing your changes.

LEARNING OUTCOME 2: MANIPULATE FILES

Indicative content1. Definition of general key terms:

Topic1. Status:

The current **state of the files in the Git repository**.

It shows whether files have been **modified**, **deleted**, or **added**, and **whether these changes have been staged for committing**.

Topic2. Branch

Git is a **separate line of development that diverges from the main line of development (usually** called the "master" branch).

Essentially, branches **allow you to work on different versions** or **features of a project simultaneously.**

For example, you could **create a new branch to** work on a specific feature, and **then merge that branch back into the main branch once the feature is complete**. By using branches, you can experiment with new ideas, isolate changes, and collaborate with others without affecting the main codebase.

Topic3. Commit

A **saved version of the changes you have made to the files in the Git repository.** Each commit has a unique identifier, a commit message describing the changes, and a timestamp.

Indicative content 2. Add file change to Git staging area:

Before committing changes to the Git repository, you need **to add them to the staging area.**

This is where you can review the changes and decide which files to include in the next commit.

Topic1. Operation on Git status command:

The "**git status**" command shows you the current status of the files in the repository, including:

New untracked files: files that **have been added to the directory but not yet added to the staging area.**

Modified files: files that have been changed since the last commit.

Deleted files: files that **have been deleted since the last commit.**

Topic 1.1 View new untracked file:

To view a list of new untracked files in the repository, run the command "**git status**". The output will show the names of the files that have not been added to the staging area.

Topic 1.2 View modified file:

To view a list of modified files in the repository, run the command "**git status**". The output will show the names of the files that have been changed since the last commit.

Topic 1.3 View deleted file:

To view a list of deleted files in the repository, run the command "**git status**". The output will show the names of the files that have been deleted since the last commit.

Topic 2. Operation on Git add command:

The "**git add**" command adds changes to the staging area. You can use this command to stage specific files or entire directories.

Topic 2.1 Stage all files:

To stage all files in the repository, run the command **"git add ." (period).** This will add all modified and new files to the staging area.

Topic 2.2 Stage a file:

To stage a specific file, run the command "**git add <filename>". Replace <filename>** with the name of the file you want to stage.

Topic2.3 Stage folder:

To stage all files in a specific folder, run the command "**git add <foldername>/". Replace <foldername> w**ith the name of the folder you want to stage.

Topic 3. Operation on git reset command:

The "**git reset**" command **allows you to undo changes** and **unstage files**. Some common use cases for this command include:

Unstaging a file: **git reset <filename>**

Deleting and staging a file/folder: **git reset <filename/foldername> && git add <filename/foldername>**

Topic 4. Operation on rm command:

The "**rm**" command **allows you to remove files** or **directories from your local file system**. Some common use cases for this command include:

1. Removing and staging a file**: rm <filename> && git add <filename>**
2. Removing and staging a folder: **rm -r <foldername> && git add <foldername>**

Indicative content 3. Commit File changes to git local repository:

After staging your changes, you can **commit them to your local repository**. It's best practice to write a descriptive commit message that summarizes the changes made. Some common operations related to committing include:

Best practice of creating a commit message: Write a summary of the changes made in the first line, followed by a more detailed description if necessary.

Topic 1. Operation on git commit command:

1.1 Commit a file: **git commit -m "Commit message"** <filename>

1.2 Edit commit message: **git commit --amend**

Topic2. Operation on git log command:

The "**git log**" command **allows you to view a list of commits made to a repository**. Some common use cases for this command include:

2.1. To see a simplified list of commit: **git log --oneline**

2.2. To see a list of commits with more detail: git log

Indicative content 4. Manage branches:

Git branches **allow you to work on different versions or features of a project without affecting the main** "master" branch. Some common operations related to branches include:

Topic1. Operations on branches:

Create branch: **git branch <branchname>**

List branch: **git branch**

Delete local: **git branch -d <branchname>** and

Remote branch: **git push origin --delete <branchname>** respectively

Switch branch: **git checkout <branchname>**

Rename branch: **git branch -m <oldbranchname> <newbranchname>**

LEARNING OUTCOME 3: SHIP CODES

Indicative content 1. Definition of general key terms:

1.1. **Pull**: The process of bringing changes from a remote repository to a local repository.

1.2. **Fetch**: The process of retrieving changes from a remote repository to a local repository without merging them.

1.3. **Push:** The process of sending changes from a local repository to a remote repository.

1.4. **Pull request**: A request made by a developer to merge their changes into the main branch of a repository.

1.5. **Merge**: The process of combining changes from different branches into a single branch.

1.6. **Branch**: A separate line of development that diverges from the main line of development (usually called the "master" branch).

Indicative content 2. Fetch file from GitHub repository:

Git fetch command is used to retrieve changes from a remote repository to a local repository without merging them.

Topic 1. Operation on git fetch command

To fetch the remote repository, use the command: **git fetch <remote-name>**

To fetch a specific branch from the remote repository, use the command**: git fetch <remote-name> <branch-name>**

To fetch all the branches simultaneously from the remote repository, use the command: git fetch --all

To synchronize the local repository with the remote repository, use the command: **git fetch <remote-name> && git merge <remote-name>/<branch-name>**

Topic 2. Operation on git pull:

Git pull command is used to retrieve changes from a remote repository to a local repository and merge them.

Default git pull: **git pull**

Git pull remote branch: **git pull <remote-name> <branch-name>**

Git force pull: **git pull --force**

Git pull origin master: **git pull origin master**

Indicative content 2. Push files to remote branch

Git push command is used to send changes from a local repository to a remote repository.

Topic 1. Tags used on git push command

-f: **force push**

-u: **set the upstream branch**

--tags: push tags to the remote repository

Topic 2. Operation on git push

2.1. Push on origin master**: git push origin master**
2.2. Git push force: **git push –force**
2.3. Git push verbose: **git push –verbose**
2.4. Delete a remote branch: **git push <remote-name> --delete <branch-name>**

Indicative content 3. Merge branches on remote repository

Git merge command **is used to combine changes from different branches into a single branch**.

Operation on git rebase command

git rebase <branch-name>: to apply changes from the specified branch to the current active branch

Create pull request: A developer can create a pull request to merge their changes into the main branch Operation on git merge:

of a repository.

Merge the specified commit to current active branch: git merge <commit-hash>

Merge commits into the master branch: git checkout master && git merge <branch-name>

This is a common scenario for large features or when several developers are working on a project simultaneously.

```
Start a new feature
git checkout -b new-feature main
# Edit some files
git add <file>
git commit -m "Start a feature"
# Edit some files
git add <file>
git commit -m "Finish a feature"
# Develop the main branch
git checkout main
# Edit some files
git add <file>
git commit -m "Make some super-stable changes to main"
# Merge in the new-feature branch
git merge new-feature
git branch -d new-feature
```

Practical assessment

UMUHIRE XP is a Senior Developer of Innovate company Ltd located at RUTSIRO District, he assigned a project to 5 developers to design web application that contains different forms such as: login form, Student Registration form, Entertainment form, courses registration form and book registration form, using html language, but due to Covid-19 pandemic developers were not able to work together on the given task and become difficult to control them. Senior developer decided to assign tasks to each developer respectively and remotely. and he recommended them to work individually on a given task.

As one among developpers of that company you are assigned the following task:

- Create an remote repository on GitHub that will be used with the name of your full name.
- Clone that repository to local computer.
- Inside that repository create those required forms to be commited.