RQF LEVEL 3

TRADE: SOFTWARE DEVELOPMENT

MODULE CODE:SWDVC301

TEACHER'S GUIDE

Module name: CONDUCT VERSION CONTROL

MODULE NAME:CONDUCT VERSION CONTROL

2022

Table of content

Contents

Acronyms

CLI: Command-Line Interface

VCS: Version Control System

Bash: Bourne Again Shell

CMD is an acronym for Command

MS DOS: Microsoft Disk Operating System

CVCS: Centralized Version Control System

DVCS: Distributed Version Control Systems

CVS: Concurrent Version System
VS: Visual Studio
URL: Uniform Resource Locator
HTTP: Hyper Text Transfer Protocal
HTTPS: Hyper Text Transfer Protocal Secure
SSH: Secure Shell

Introduction

Let's imagine a company operating in many countries with many employees, in order to work they need to come to work in different departments. What happened during the COVID 19 pandemic? Are employees still working as usual? We all know it's not possible, because many companies were forced to close their doors because of the stay-at-home order. How would the company's employees continue to work and be aware of the changes in their company? How would they know about the changes in their institution? How would they keep up with the past?

There are different problems that may be raised, among them we can cite the followings:

- ✓ Lack of collaboration
- ✓ Storing versions
- ✓ Restoring previous version once needed
- ✓ Figuring out what happened in different branches
- ✓ Problem of backup

In summary Is the reason why we need a version control system as developers in order to overcome those cited issues.

Module Code and Title: SWDVC301 CONDUCT VERSION CONTROL

Learning Outcome 1: Setup repository
Learning Outcome 2: Manipulate files
Learning Outcome 3: Ship codes

Learning outcome 1: Setup repository

Picture/s reflecting the Learning outcome 1

Learning outcome 1. Setup Repository

Indicative contents

1.1 Introduction to version control

1.2 Description of git

1.3 Use of GitHub repository

Duration: 25 hrs

Learning outcome 1 objectives

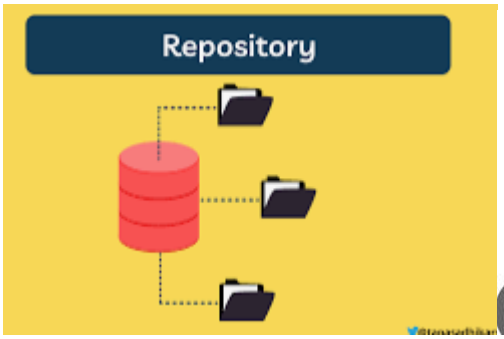
By the end of the learning outcome, the trainees will be able to:

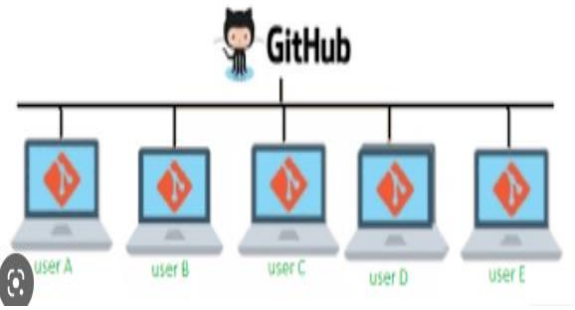
1. Git is introduced based on version control
2. Git is properly initiated based on Git commands
3. Repository is properly created based on the project.
1. 4. Remote URL is properly set in accordance with Git commands

Resources

Equipment	Tools	Materials
• Computer • Projector • White board	• Git • GitHub • Text editor (vs code) • Terminal (CMD, Gitbash).	• Internet • Electricity • Flipchart • Marker pen

Advance preparation:





Indicative content 1.1: Introduction to version control

Summary for the trainer related to the indicative content

Definition of general key terms

Repository

A **Git repository** tracks and saves the history of all changes made to the files in a Git project. It saves this data in a directory called**.git**, also known as the repository folder.

Git uses a version control system to track all changes made to the project and save them in the repository. Users can then delete or copy existing repositories or create new ones for ongoing projects.

Types of Git Repository

There are two types of Git repositories, based on user permissions:

Bare Repositories

Software development teams use **bare repositories** to share changes made by team members. Individual users aren't allowed to modify or create new versions of the repository.

Non-Bare Repositories

With **non-bare repositories**, users can modify the existing repository and create new versions. By default, the cloning process creates a non-bare repository.

Version control

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

For the examples in this book, you will use software source code as the files being version controlled, though in reality you can do this with nearly any type of file on a computer.

If you are a graphic or web designer and want to keep every version of an image or layout (which you would most certainly want to), a Version Control System (VCS) is a very wise thing to use. It allows you to revert selected files back to a previous state, revert the entire project back to a previous state, compare changes over time, see who last modified something that might be causing a problem, who introduced an issue and when, and more. Using a VCS also generally means that if you screw things up or lose files, you can easily recover. In addition, you get all this for very little overhead.

Git

Git is the most commonly used version control system.

Git tracks the changes you make to files, so you have a record of what has been done, and you can revert to specific versions should you ever need to.

Git also makes collaboration easier, allowing changes by multiple people to all be merged into one source.

So regardless of whether you write code that only you will see, or work as part of a team, Git will be useful for you.

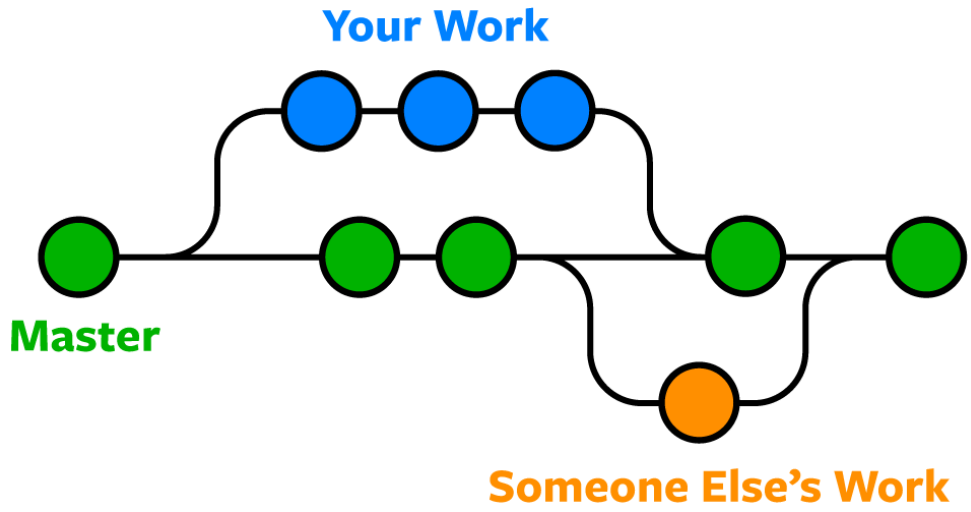


Figure 1 Git

GitHub:

GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere.

GitHub is *an online software development platform used for storing, tracking, and collaborating on software projects.*

GitHub is *a Git repository hosting service that provides a web-based graphical interface.*

Terminal:

The terminal *is an interface that lets you access the command line(Bash and CMD).*

Bash (Bourne Again Shell) is the free and enhanced version of the Bourne shell distributed with Linux and GNU operating systems. Bash is similar to the original, but has added features such as command-line editing.

CMD is an acronym for Command. Command prompt, or CMD, is the command-line interpreter of Windows operating systems. It is similar to Command.com used in DOS and Windows 9x systems called "MS-DOS Prompt". It is analogous to Unix Shells used on Unix like system.

The command prompt is a native application of the Windows operating system and gives the user an option to perform operations using commands.

INTRODUCTION TO VERSION CONTROL

TYPES OF VERSION CONTROL

Centralized version control

With centralized version control systems, you have a single “central” copy of your project on a server and commit your changes to this central copy.

A centralized version control system offers software development teams a way to collaborate using a central server. In a centralized version control system (CVCS), a server acts as the main repository which stores every version of code.

You pull the files that you need, but you never have a full copy of your project locally. Some of the most common version control systems are centralized, including Subversion (SVN) and Perforce.

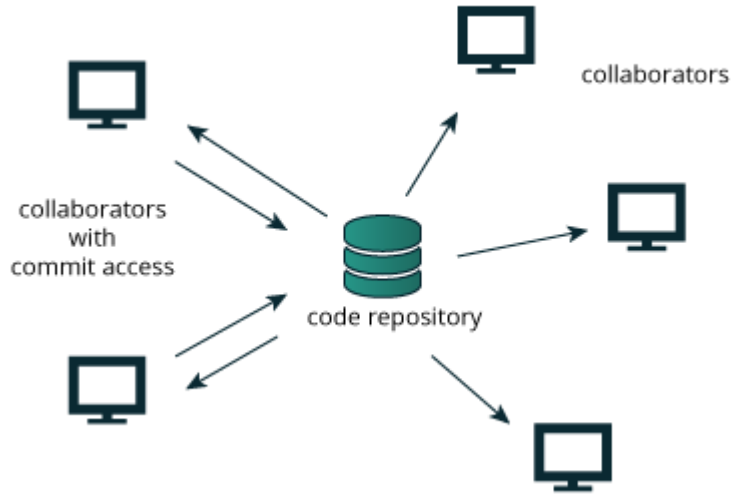


Figure 2 Centralised Version Control

Distributed version control

With distributed version control systems (DVCS), you don't rely on a central server to store all the versions of a project’s files. Instead, you clone a copy of a repository locally so that you have the full history of the project. Two common distributed version control systems are Git and Mercurial.

While you don't have to have a central repository for your files, you may want one "central" place to keep your code so that you can share and collaborate on your project with others. That's where Bitbucket comes in. Keep a copy of your code in a repository on Bitbucket so that you and your teammates can use Git or Mercurial locally and to push and pull code.

A distributed version control system (DVCS) is a type of version control where the complete codebase — including its full version history — is mirrored on every developer's computer. It's abbreviated DVCS. Changes to files are tracked between computers.

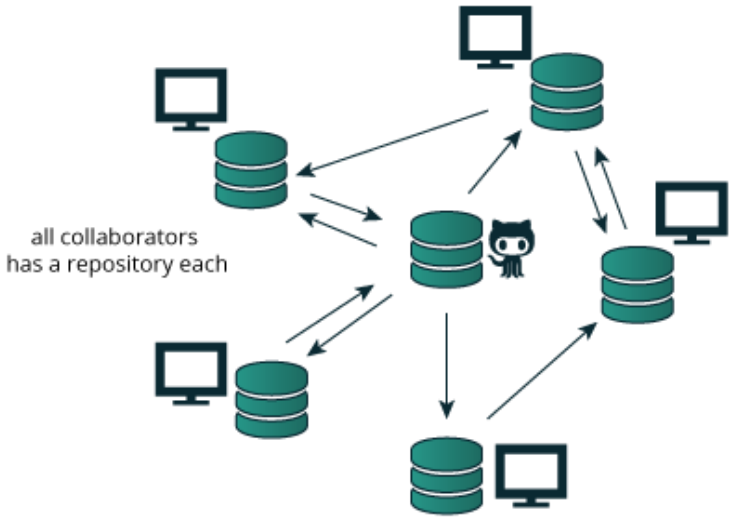


Figure 3 Distrubuted Vesrion Control

Local version control

A local version control system is a local database located on your local computer, in which every file change is stored as a patch.

Well Known version control system

1. Git
2. CVS (Concurrent Version System)
3. Mercurial
4. SVN (subversion)
5. GitLab
6. AWS Code Commit
7. Perforce
8. Beanstalk
9. Team Foundation Server
10. Bitbucket

Benefits of Version control

- ✓ Help in managing and protecting the source code
- ✓ Keep track of all the modifications made to the code
- ✓ Comparing earlier versions of the code
- ✓ Supports developer’s workflow and not any rigid way of working

Theoretical learning Activity

- ✓ Brainstorm why version control is needed as front end developers within groups of three trainees.
- ✓ Discuss on the main difference between remote version control (distributed and centralised) and local version control system and developers.
- ✓ Brainstorm on benefits of version control in our daily life as programmers

Practical learning Activity

- ✓ In group of three open and test if the command prompt of your computer is working.

Points to Remember

- ✓ Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.
- ✓ We have two main types of version control system local and remote.
- ✓ For local version control we use git weather for remote version control we use GitHub or GitLab.
- ✓ You can run git commands by using CMD.

Indicative content 1.2: Description of git

Summary for the trainer related to the indicative content

Git Basic concept

Branch

A *branch represents an independent line of development*. Branches serve as an abstraction for the edit/stage/commit process.

Clone

The git clone command is *used to create a copy of a specific repository or branch within a repository*.

Pull

The git pull command is *used to fetch and download content from a remote repository and immediately update the local repository to match that content*.

Push

The git push command is **used to upload local repository content to a remote repository**. Pushing is how you transfer commits from your local repository to a remote repo. It's the counterpart to git fetch , but whereas fetching imports commits to local branches, pushing exports commits to remote branches.

Commit

It is used to record the changes in the repository. It is the next command after the git add. Every commit contains the index data and the commit message

Initialisation

The git init command **creates a new Git repository**. It can be used to convert an existing, unversioned project to a Git repository or initialize a new, empty repository. Most other Git commands are not available outside of an initialized repository, so this is usually the first command you'll run in a new project.

The git init command is the *first command* that you will run on Git. The git init command is used to create a new blank repository.

Git architecture

Many VCS's use a two-tier architecture i.e a repository and a working copy. Git uses three-tier architecture i.e a working directory, staging area and local repository. The three stages of git can store different(or the same) states of the same code in each stage.

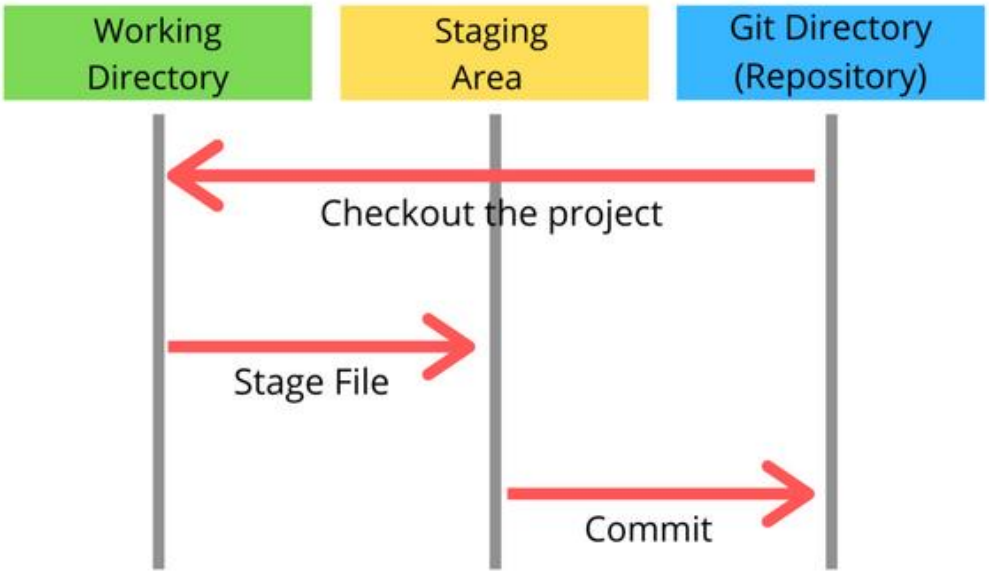


Figure 4 Git Architecture

Git workflow

A Git workflow is **a recipe or recommendation for how to use Git to accomplish work in a consistent and productive manner**. Git workflows encourage developers and DevOps teams to leverage Git effectively and consistently. Git offers a lot of flexibility in how users manage changes.

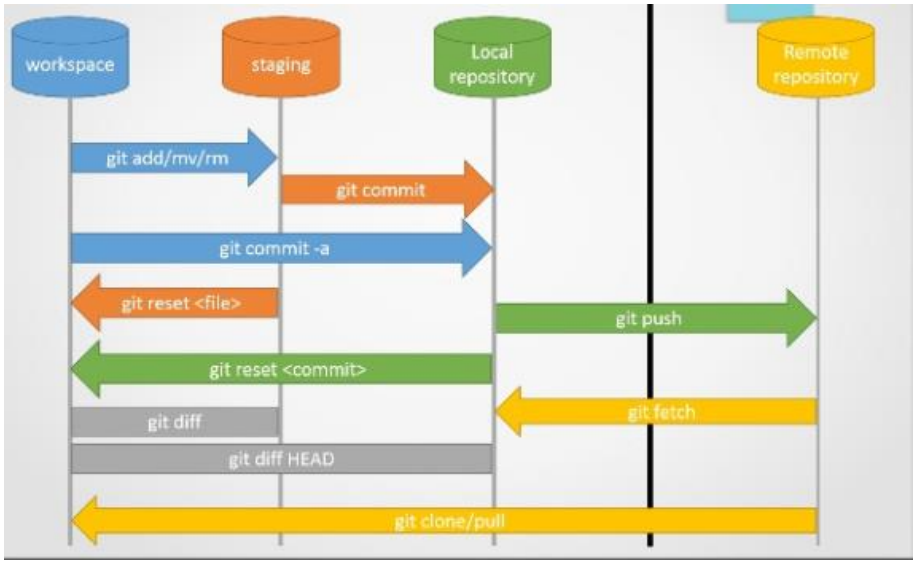


Figure 5 Git workflow in detail

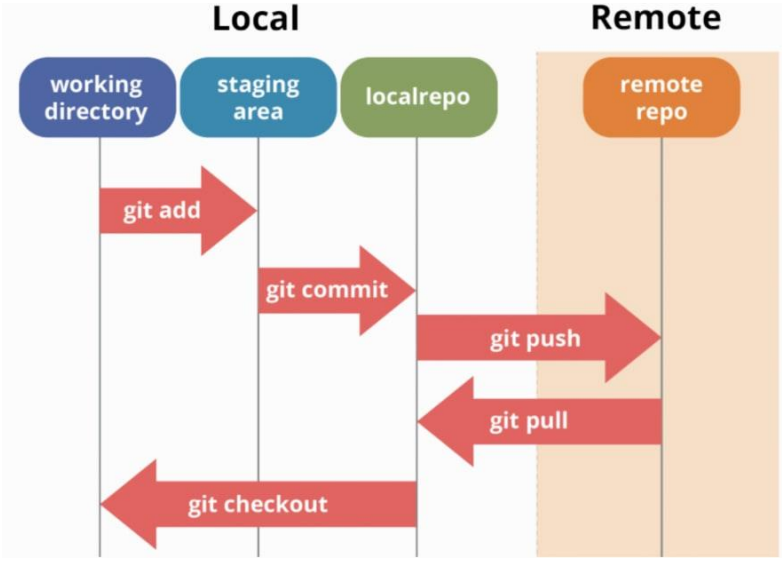


Figure 6 Git workflow simplified

INITIALISATION OF GIT

TERMINAL BASIC COMMANDS

git init

This command turns a directory into an empty Git repository. This is the first step in creating a repository. After running git init, adding and committing files/directories is possible.

git add

Adds files in the to the staging area for Git. Before a file is available to commit to a repository, the file needs to be added to the Git index (staging area). There are a few different ways to use git add, by adding entire directories, specific files, or all unstaged files.

git commit

Record the changes made to the files to a local repository. For easy reference, each commit has a unique ID.

git status

This command returns the current state of the repository.

git status will return the current working branch. If a file is in the staging area, but not committed, it shows with *git status*. Or, if there are no changes it'll return *nothing to commit, working directory clean.*

git config

With Git, there are many configurations and settings possible. *git config* is how to assign these settings. Two important settings are user user.name and user.email. These values set what email address and name commits will be from on a local computer. With *git config,* a --*global* flag is used to write the settings to all repositories on a computer. Without a --*global* flag settings will only apply to the current repository that you are currently in.

git branch

To determine what branch the local repository is on, add a new branch, or delete a branch.

git checkout

To start working in a different branch, use *git checkout* to switch branches.

git merge

Integrate branches together. *git merge* combines the changes from one branch to another branch. For example, merge the changes made in a staging branch into the stable branch.

git remote

To connect a local repository with a remote repository. A remote repository can have a name set to avoid having to remember the URL of the repository.

git clone

To create a local working copy of an existing remote repository, use *git clone* to copy and download the repository to a computer. Cloning is the equivalent of *git init* when working with a remote repository. Git will create a directory locally with all files and repository history.

git pull

To get the latest version of a repository run *git pull*. This pulls the changes from the remote repository to the local computer.

git push

Sends local commits to the remote repository. *git push* requires two parameters: the remote repository and the branch that the push is for.

git log

To show the chronological commit history for a repository. This helps give context and history for a repository. *git log* is available immediately on a recently cloned repository to see history.

git rm

Remove files or directories from the working index (staging area). With *git rm*, there are two options to keep in mind: force and cached. Running the command with force deletes the file. The cached command removes the file from the working index. When removing an entire directory, a recursive command is necessary.

INSTALLATION OF GIT SETUP

To use Git, you have to install it on your computer. Even if you have already installed Git, it's probably a good idea to upgrade it to the latest version. You can either install it as a package or via another installer or download it from its official site.

Now the question arises that how to download the Git installer package. Below is the stepwise installation process that helps you to download and install the Git.

Step1

To download the Git installer, visit the Git's official site and go to download page. The link for the download page is https://git-scm.com/downloads. The page looks like as

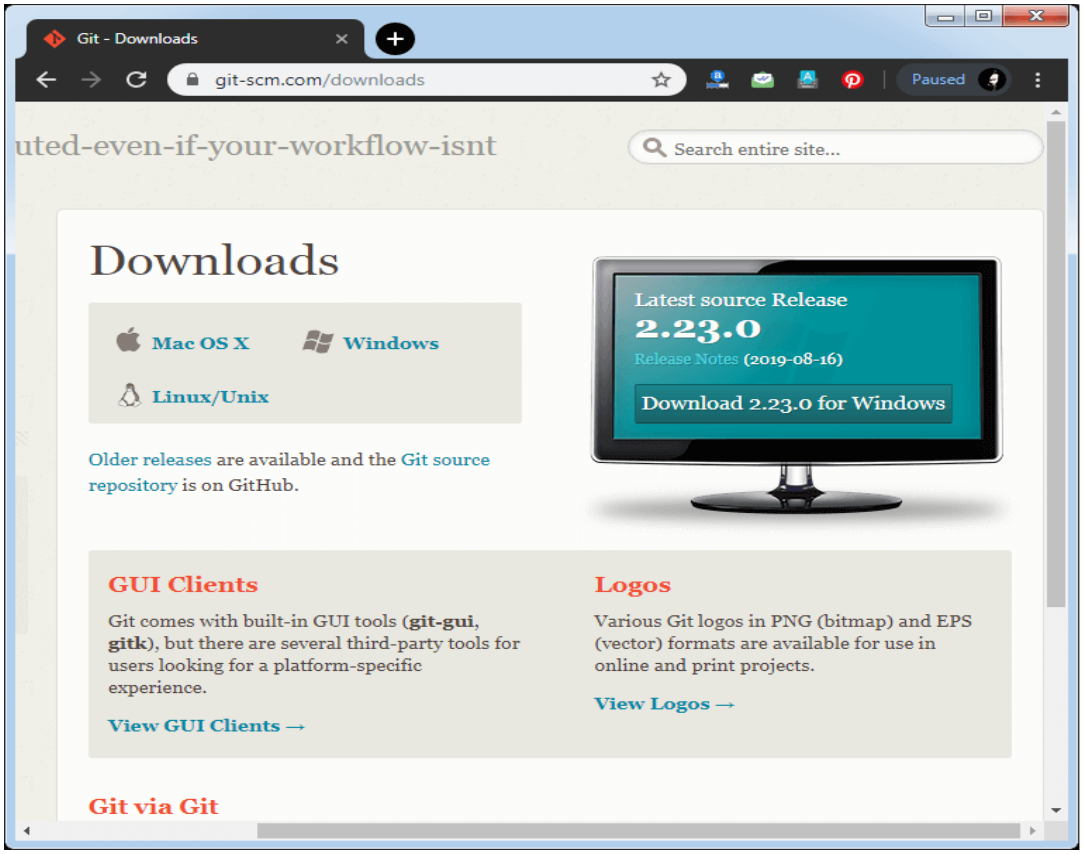


Figure 7 Installation of git step 1

Click on the package given on the page as **download 2.23.0 for windows**. The download will start after selecting the package.

Now, the Git installer package has been downloaded.

Step2

Click on the downloaded installer file and select **yes** to continue. After the selecting **yes** the installation begins, and the screen will look like as

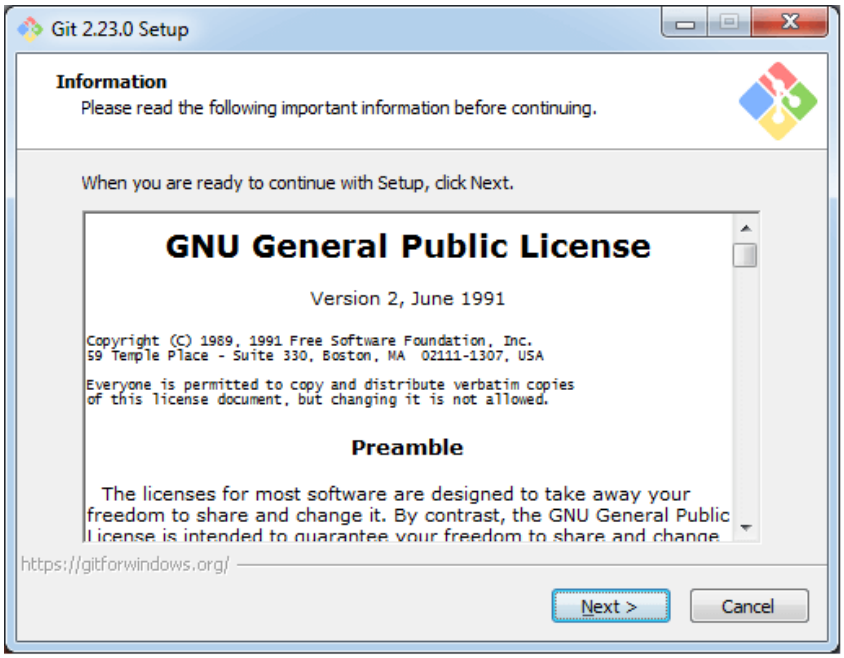
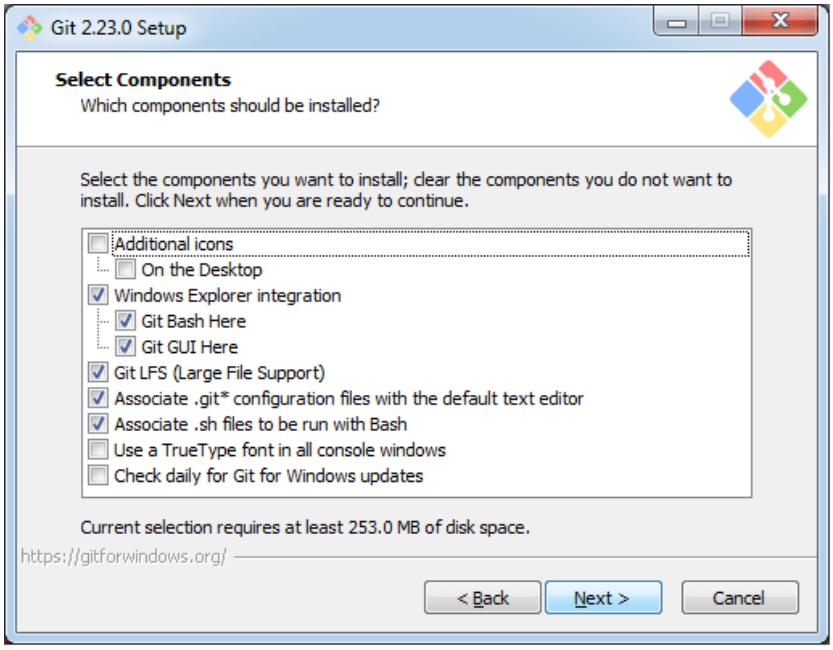


Figure 8 Install Git Step 2

Click on **next** to continue.

Step3

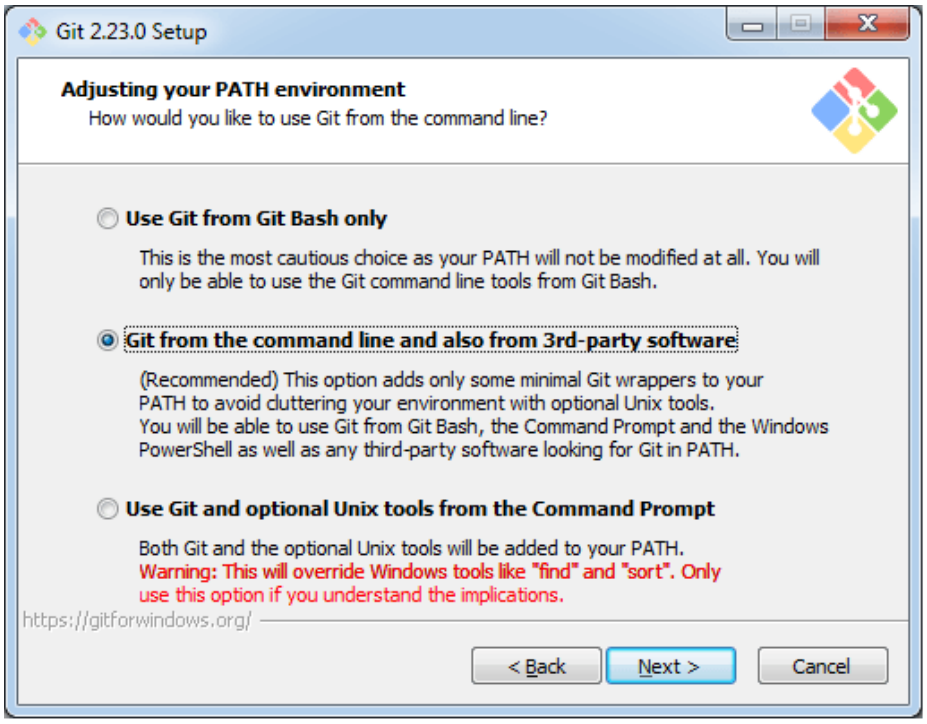
Default components are automatically selected in this step. You can also choose your required part.



Click next to continue.

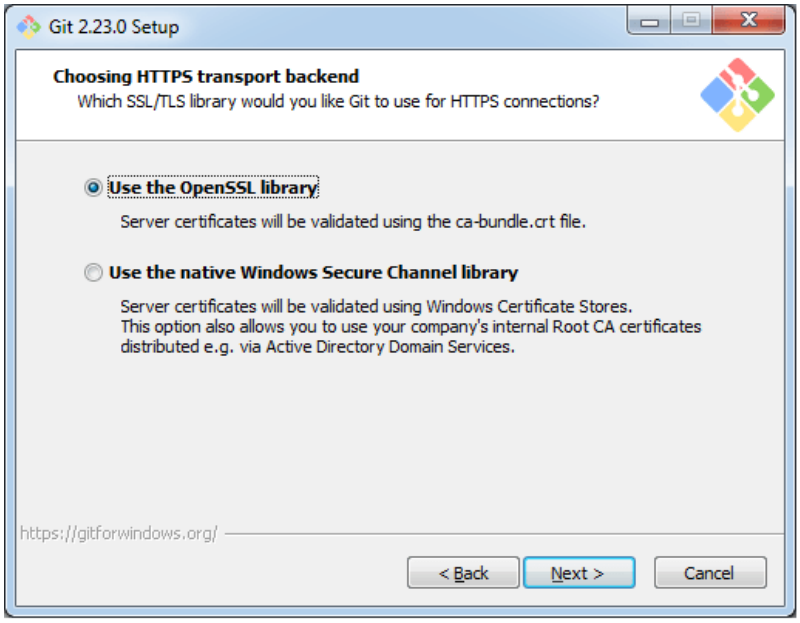
Step4

The default Git command-line options are selected automatically. You can choose your preferred choice. Click **next** to continue.



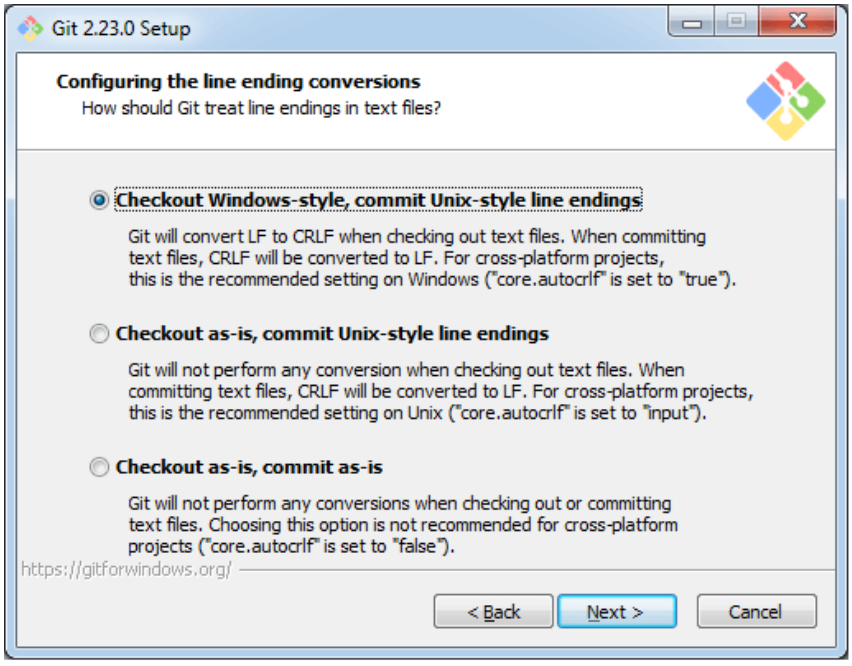
Step5

The default transport backend options are selected in this step. Click **next** to continue.



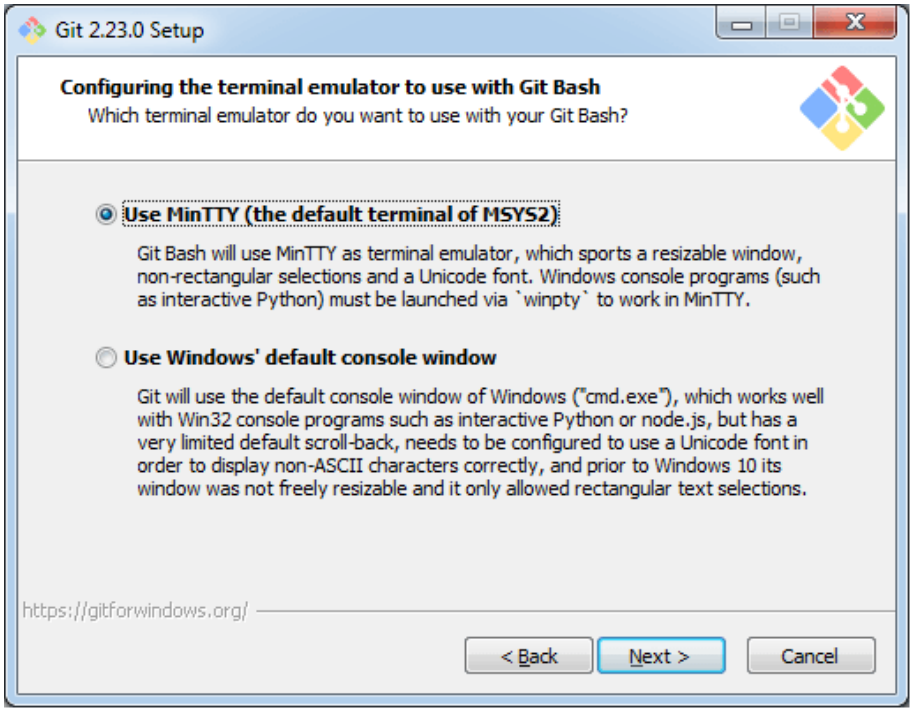
Step6

Select your required line ending option and click next to continue.



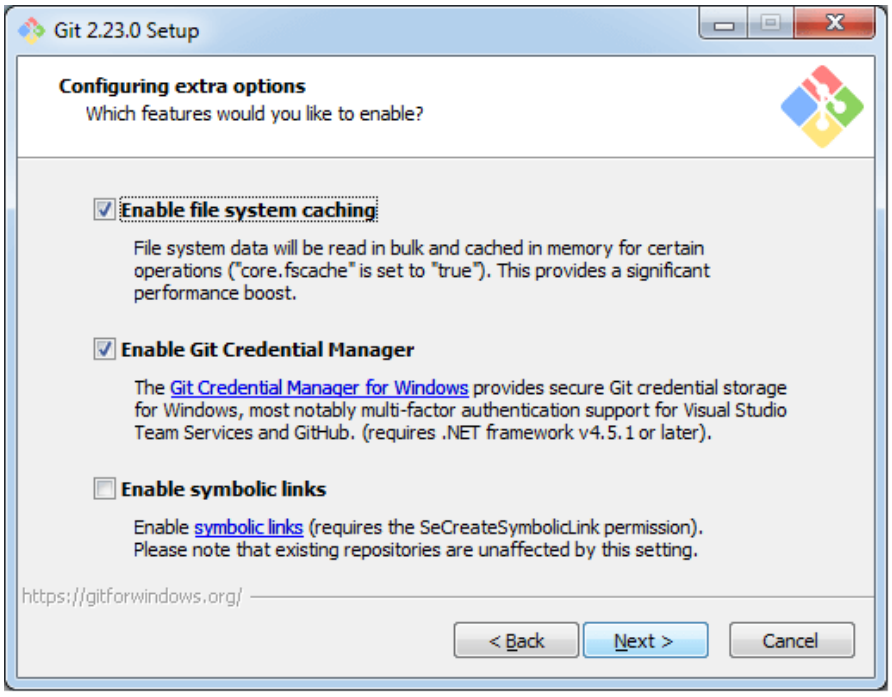
Step7

Select preferred terminal emulator clicks on the **next** to continue.



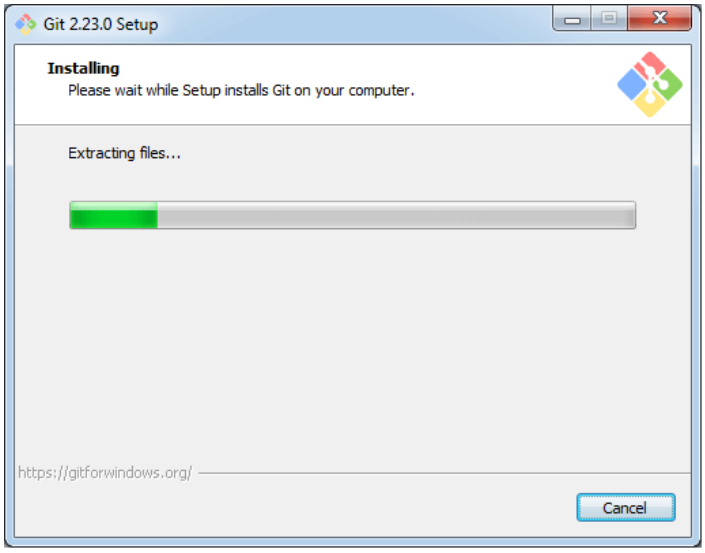
Step8

This is the last step that provides some extra features like system caching, credential management and symbolic link. Select the required features and click on the **next** option.



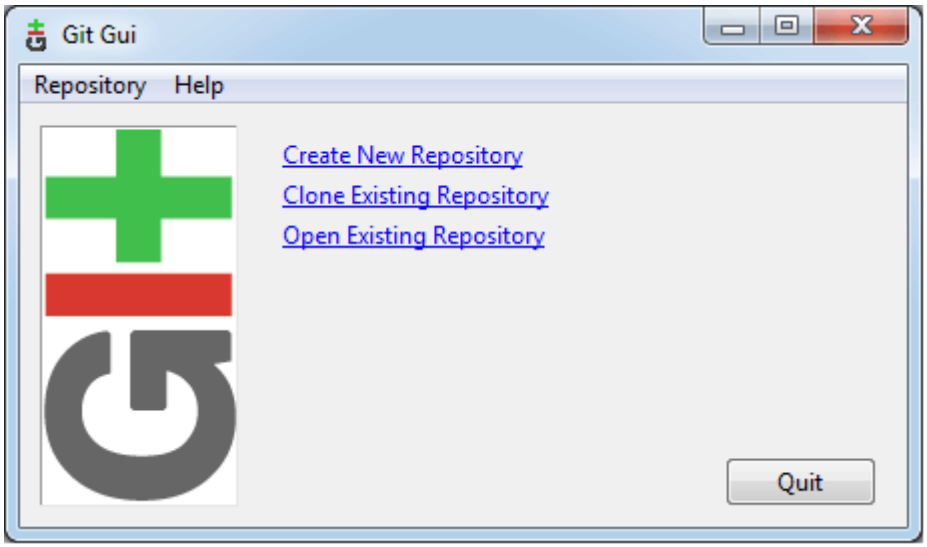
Step9

The files are being extracted in this step.



Therefore, The Git installation is completed. Now you can access the **Git Gui** and **Git Bash**.

The **Git Gui** looks like as



It facilitates with three features.

- Create New Repository
- Clone Existing Repository
- Open Existing Repository

The **Git Bash** looks like as

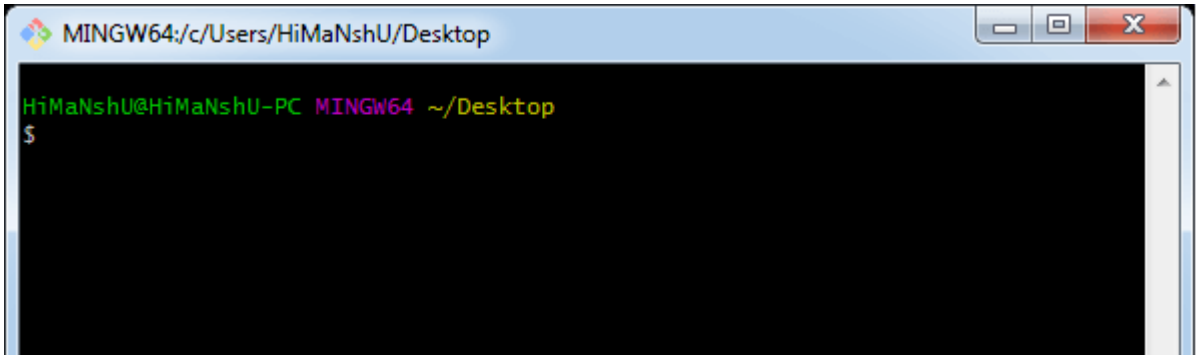


Figure 9 GitBash GUI

CONFIGURE GIT

Git init command

This command is used to create a local repository.

Syntax

$ git init Demo

The init command will initialize an empty repository. See the below screenshot.

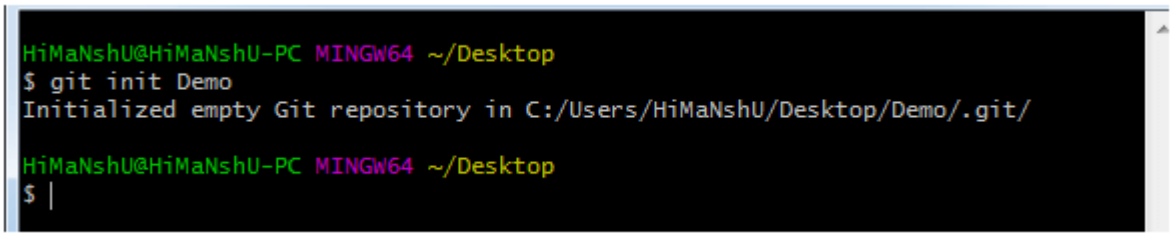


Figure 10 Git init example

Git config command

This command configures the user. The Git config command is the first and necessary command used on the Git command line. This command sets the author name and email address to be used with your commits. Git config is also used in other scenarios.

```
$ git config --global user.name "ImDwivedi1"

$ git config --global user.email "Himanshudubey481@gmail.com"
```

GIT – VERSION COMMAND

Open the *command* prompt "terminal" and type *git version* to verify *Git* was installed.

Configure .git ignore file

Theoretical learning Activity

- ✓ In group of three brainstorm about terminal basics commands and their usage and syntax.
- ✓ Discuss all steps that are included in git architecture.
- ✓ Explain git workflow.

Practical learning Activity

- ✓ In group of two install git on your computer and compare it interface with CMD.
- ✓ Configure the installed git to your first_name as username and your email as user_Email.
- ✓ By using git create a repository and name it by using your lastname. Inside that repository initialise an empty git repository.

Points to Remember (Take home message)

- ✓ Git can be installed on computer
- ✓ Once running git commands you can use CMD or git bash
- ✓ We can use different git commands in order to perform different tasks such as create a repository, clone a repository, push local repository to remote VC and others.
- ✓ Git workflow consist of the following:
  - Staging are
  - Working area
  - Local repository
  - Remote repository

USE OF GITHUB REPOSITORY

Description of GitHub

GitHub is a Git repository hosting service. GitHub also facilitates with many of its features, such as access control and collaboration. It provides a Web-based graphical interface.

GitHub is an American company. It hosts source code of your project in the form of different programming languages and keeps track of the various changes made by programmers.

It offers both **distributed version control and source code management (SCM)** functionality of Git. It also facilitates with some collaboration features such as bug tracking, feature requests, task management for every project.

Some of its significant features are as follows.

- Collaboration
- Integrated issue and bug tracking
- Graphical representation of branches
- Git repositories hosting
- Project management
- Team management
- Code hosting
- Track and assign tasks
- Conversations

Benefits of GitHub

GitHub can be separated as the Git and the Hub. GitHub service includes access controls as well as collaboration features like task management, repository hosting, and team management.

The key benefits of GitHub are as follows.

- It is easy to contribute to open source projects via GitHub.
- It helps to create an excellent document.
- You can attract recruiter by showing off your work. If you have a profile on GitHub, you will have a higher chance of being recruited.
- It allows your work to get out there in front of the public.
- You can track changes in your code across versions.
  Create account on GitHub

If you don't already have a GitHub account, here's how to create one.

1. Open https://github.com in a web browser, and then select **Sign up**.
2. **Enter your email** address.
3. **Create a password** for your new GitHub account, and **Enter a username**, too. Next, choose whether you want to receive updates and announcements via email, and then select **Continue**.
4. **Verify your account** by solving a puzzle. Select the **Start Puzzle** button to do so, and then follow the prompts.
5. After you verify your account, select the **Create account** button.
6. Next, GitHub sends a launch code to your email address. Type that launch code in the **Enter code** dialog, and then press **Enter**.
7. GitHub asks you some questions to help tailor your experience. Choose the answers that apply to you in the following dialogs:
    - **How many team members will be working with you?**
    - **What specific features are you interested in using?**
8. On the **Where teams collaborate and ship** screen, you can choose whether you want to use the Free account or the Team account. To choose the **Free** account, select the **Skip personalization** button.

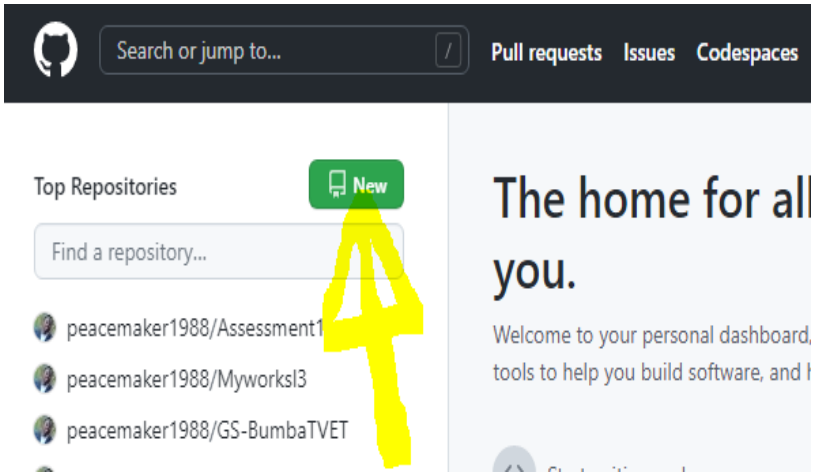
Congratulations! You've successfully created your GitHub account.

Create new remote repository

You have two options to create a Git repo. You can create one from the code in a folder on a computer, or clone one from an existing repo. If working with code that's just on the local computer, create a local repo using the code in that folder. But most of the time the code is already shared in a Git repo, so cloning the existing repo to the local computer is the recommended way to go.

Once creating a remote repository, you can use GitHub interface by following those steps.

Step 1



Or click on the icon located near the profile picture as shown on that image.

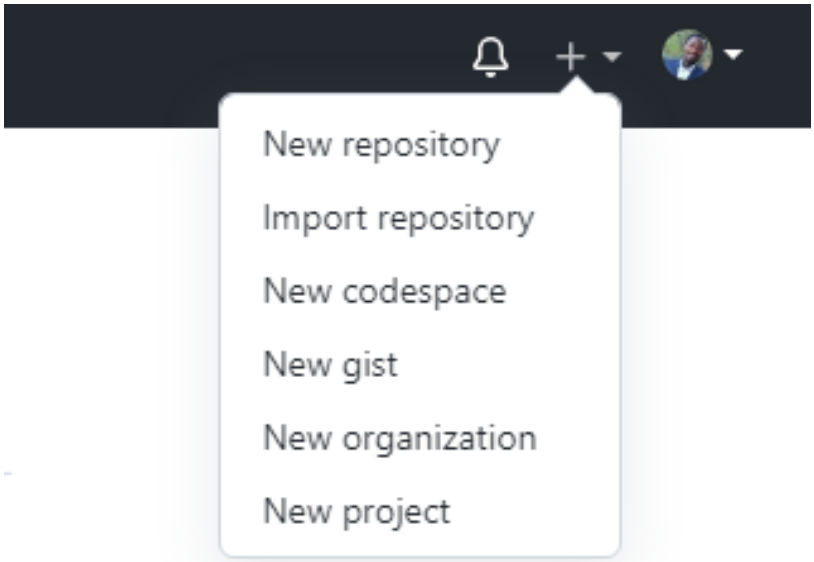


Figure 11 Creation of Remote repository

Step 2

Fill required information such as Repository name and Description as shown on the next figure and click on create repository.

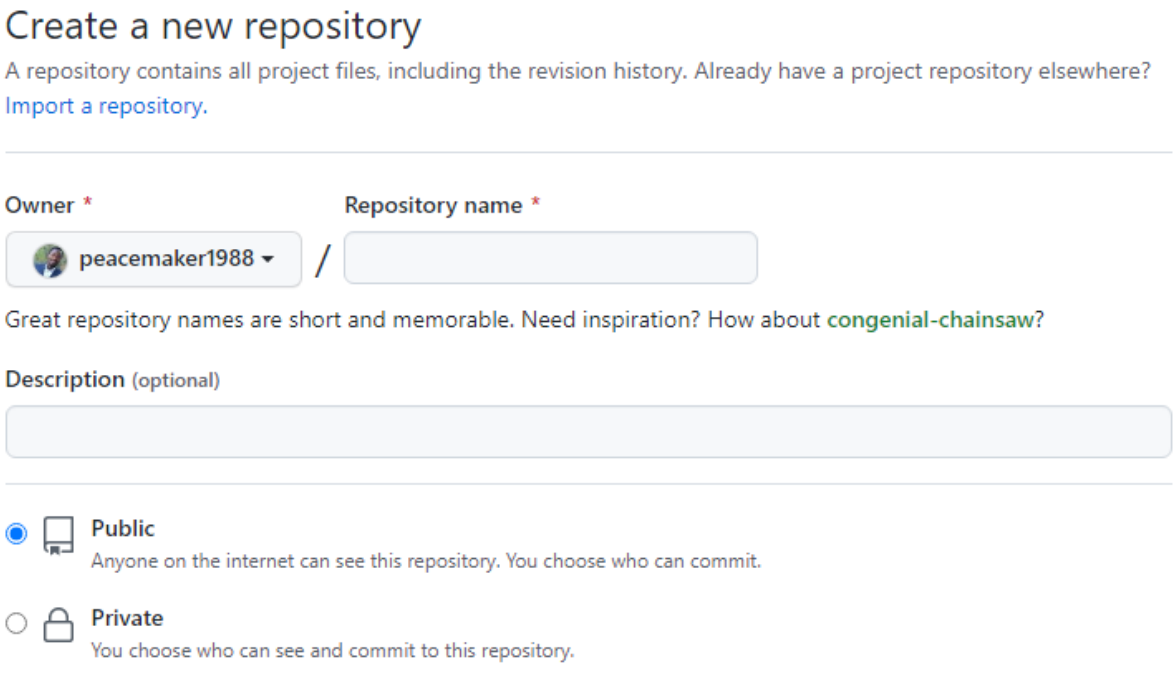
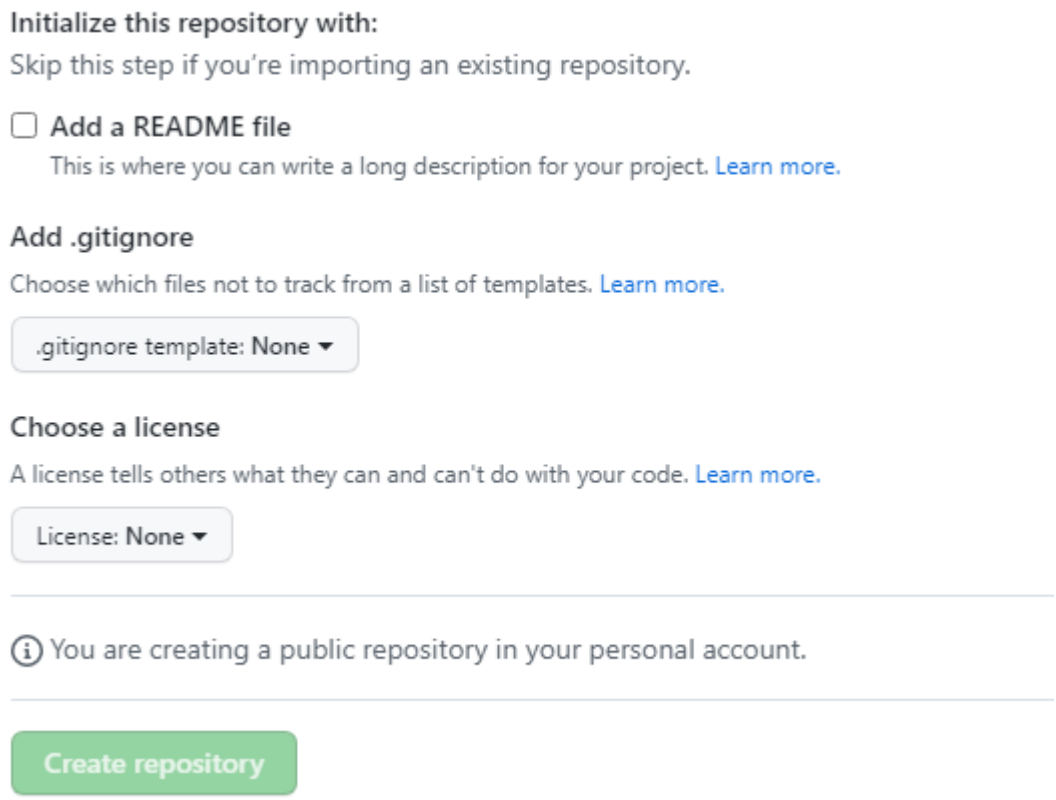


Figure 12 New Repository Name and Description



Apply git commands related to repository
Git clone

The "clone" command downloads an existing Git repository to your local computer.

You will then have a full-blown, *local* version of that Git repo and can start working on the project.

Typically, the "original" repository is located on a remote server, often from a service like GitHub, Bitbucket, or GitLab). That remote repository's URL is then later referred to as the "origin".

<repository>

Specifies the URL of the remote repository. Usually, this will point to a remote server, using a protocol like HTTP, HTTPS, SSH, or GIT.

<directory>

The name of the folder on your local machine where the repository will be downloaded into. If this option is not specified, Git will simply create a new folder named after the remote repository.

--recurse-submodules

Clones and initializes all contained submodules. If your project contains submodules, using this parameter will make sure that all submodules will both be cloned *and* initialized once the main project has been cloned. This saves you from having to manually initialize and update the submodules later.

Usage Examples

In its simplest (and most common) form, only the repository URL is specified:

```
cd folder/to/clone-into/
git clone https://github.com/gittower/git-crash-course.git
```

This will download the project to a folder named after the Git repository ("git-crash-course" in this case). If you want a different folder name, simply specify it as the last parameter:

```
git clone https://github.com/gittower/git-crash-course.git other-name
```

Git remote

In Git, the term remote is concerned with the remote repository. It is a shared repository that all team members use to exchange their changes. A remote repository is stored on a code hosting service like an internal server, GitHub, Subversion, and more. In the case of a local repository, a remote typically does not provide a file tree of the project's current state; as an alternative, it only consists of the .git versioning data.

The developers can perform many operations with the remote server. These operations can be a clone, fetch, push, pull, and more. Consider the below image:

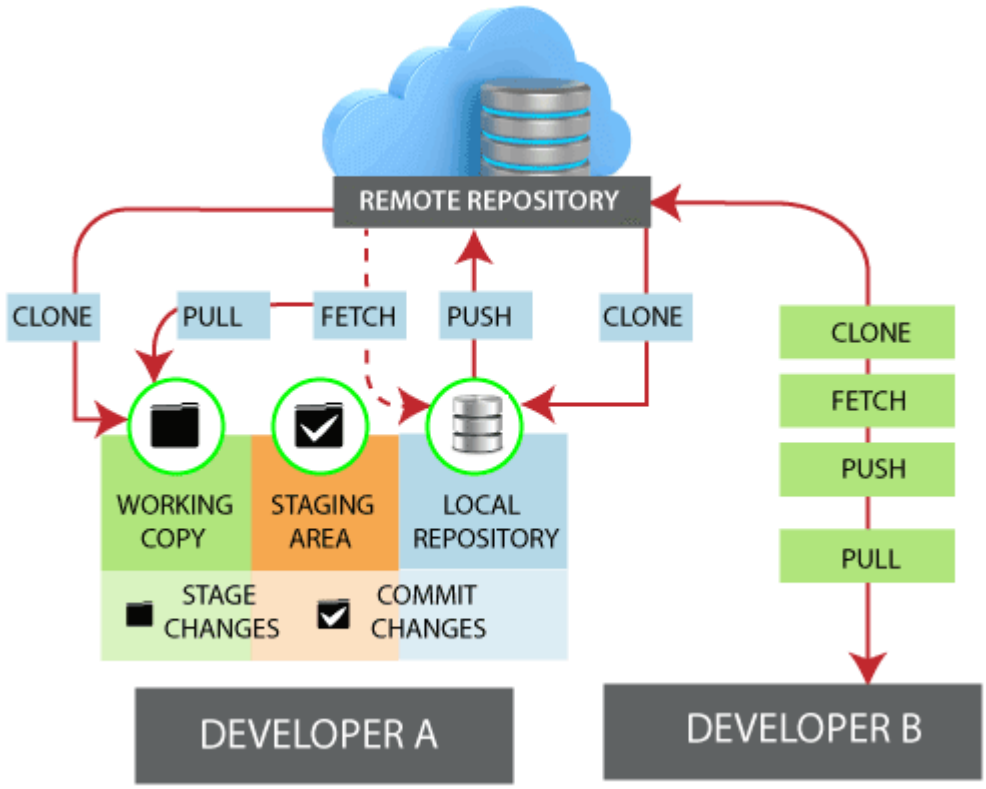


Figure 13 Git Remote

Check your Remote

To check the configuration of the remote server, run the **git remote** command. The git remote command allows accessing the connection between remote and local. If you want to see the original existence of your cloned repository, use the git remote command. It can be used as:

Syntax:

1. $ git remote

Output:

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git remote
origin
```

The given command is providing the remote name as **the origin**. Origin is the default name for the remote server, which is given by Git.

Git remote -v:

Git remote supports a specific option -v to show the URLs that Git has stored as a short name. These short names are used during the reading and write operation. Here, -v stands for **verbose**. We can use **--verbose** in place of -v. It is used as:

Syntax:

1. $ git remote -v

Or

1. $ git remote --verbose

Output:

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git remote -v
origin  https://github.com/ImDwivedi1/GitExample2.git (fetch)
origin  https://github.com/ImDwivedi1/GitExample2.git (push)
```

The above output is providing available remote connections. If a repository contains more than one remote connection, this command will list them all.

Git Remote Add

When we fetch a repository implicitly, git adds a remote for the repository. Also, we can explicitly add a remote for a repository. We can add a remote as a shot nickname or short name. To add remote as a short name, follow the below command:

Syntax:

1. $ git remote add <short name><remote URL>

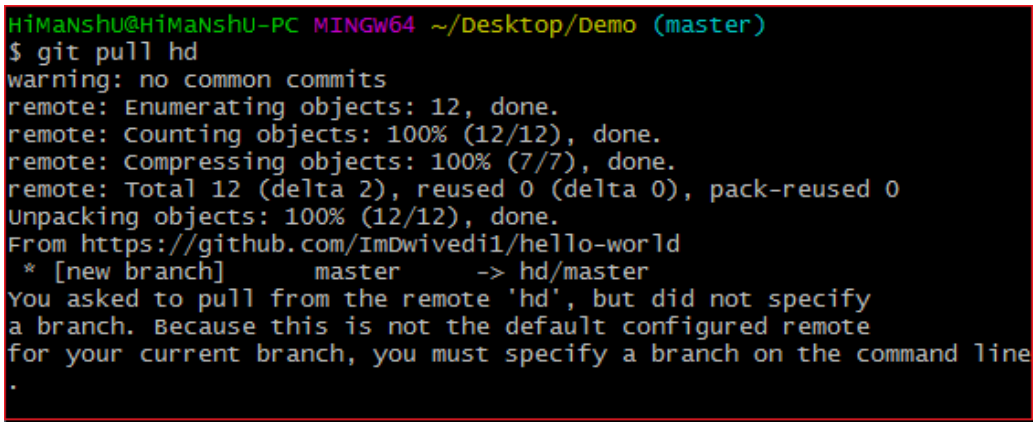
Output:

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/Demo (master)
$ git remote add hd https://github.com/ImDwivedi1/hello-world

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/Demo (master)
$ git remote -v
hd      https://github.com/ImDwivedi1/hello-world (fetch)
hd      https://github.com/ImDwivedi1/hello-world (push)

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/Demo (master)
$
```

In the above output, I have added a remote repository with an existing repository as a short name "**hd**". Now, you can use "**hd**" on the command line in place of the whole URL. For example, you want to pull the repository, consider below output:



I have pulled a repository using its short name instead of its remote URL. Now, the repository master branch can be accessed through a short name.

Fetching and Pulling Remote Branch

You can fetch and pull data from the remote repository. The fetch and pull command goes out to that remote server, and fetch all the data from that remote project that you don't have yet. These commands let us fetch the references to all the branches from that remote.

To fetch the data from your remote projects, run the below command:

1. $ git fetch <remote>

To clone the remote repository from your remote projects, run the below command:

1. $ git clone<remote>

When we clone a repository, the remote repository is added by a default name "**origin**." So, mostly, the command is used as git fetch origin.

The git fetch origin fetches the updates that have been made to the remote server since you cloned it. The git fetch command only downloads the data to the local repository; it doesn't merge or modify the data until you don't operate. You have to merge it manually into your repository when you want.

To pull the repository, run the below command:

1. $ git pull <remote>

The git pull command automatically fetches and then merges the remote data into your current branch. Pulling is an easier and comfortable workflow than fetching. Because the git clone command sets up your local master branch to track the remote master branch on the server you cloned.

Pushing to Remote Branch

If you want to share your project, you have to push it upstream. The git push command is used to share a project or send updates to the remote server. It is used as:

1. $ git push <remote><branch>

To update the main branch of the project, use the below command:

1. $ git push origin master

It is a special command-line utility that specifies the remote branch and directory. When you have multiple branches on a remote server, then this command assists you to specify your main branch and repository.

Generally, the term **origin** stands for the remote repository, and master is considered as the main branch. So, the entire statement "**git push origin master**" pushed the local content on the master branch of the remote location.

Git Remove Remote

You can remove a remote connection from a repository. To remove a connection, perform the git remote command with **remove** or **rm** option. It can be done as:

Syntax:

1. $ git remote rm <destination>

Or

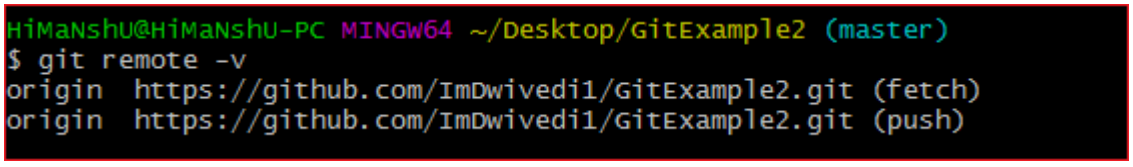
1. $ git remote remove <destination>

Consider the below example:

Suppose you are connected with a default remote server "**origin**." To check the remote verbosely, perform the below command:

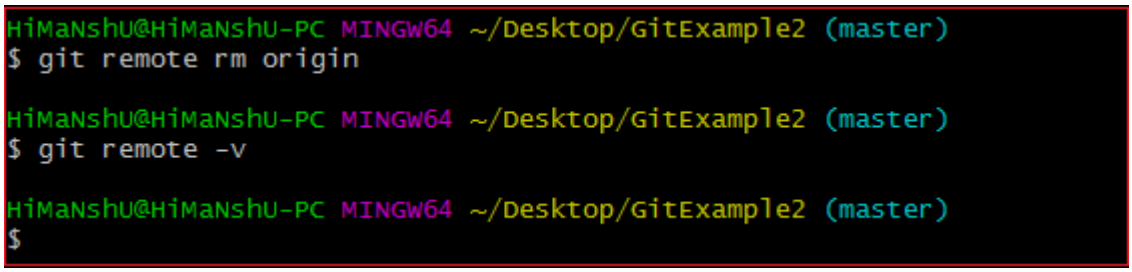
1. $ git remote -v

Output:



The above output will list the available remote server. Now, perform the remove operation as

mentioned above. Consider the below output:



In the above output, I have removed remote server "origin" from my repository.

Git Remote Rename

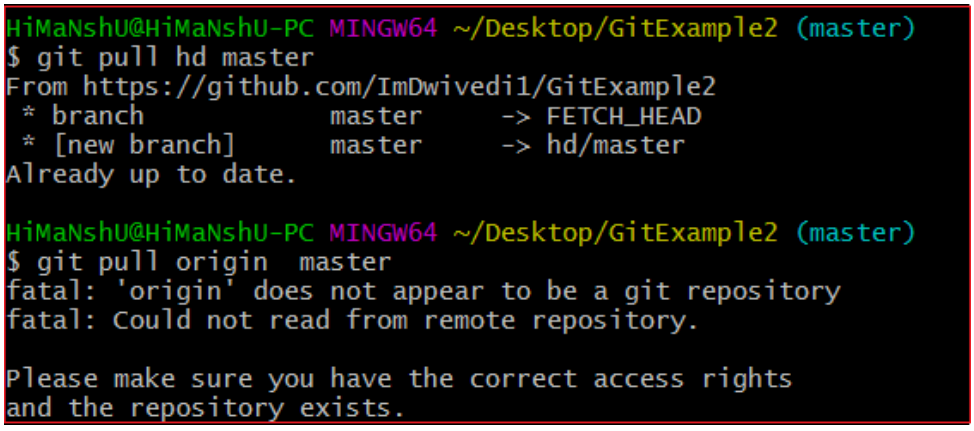
Git allows renaming the remote server name so that you can use a short name in place of the remote server name. Below command is used to rename the remote server:

Syntax:

1. $ git remote rename <old name><new name>

Output:

In the above output, I have renamed my default server name origin to hd. Now, I can operate using this name in place of origin. Consider the below output:



In the above output, I have pulled the remote repository using the server name hd. But, when I am using the old server name, it is throwing an error with the message "**'origin' does not**

appear to be a git repository." It means Git is not identifying the old name, so all the operations will be performed by a new name.

Git Show Remote

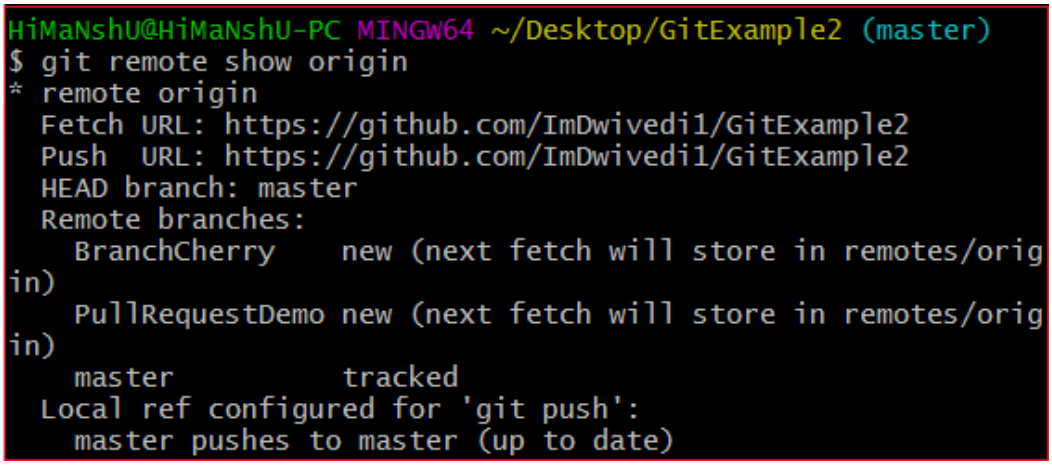
To see additional information about a particular remote, use the git remote command along with show sub-command. It is used as:

Syntax:

1. $ git remote show <remote>

It will result in information about the remote server. It contains a list of branches related to the remote and also the endpoints attached for fetching and pushing.

Output:



The above output is listing the URLs for the remote repository as well as the tracking branch information. This information will be helpful in various cases.

Git Change Remote (Changing a Remote's URL)

We can change the URL of a remote repository. The git remote set command is used to change the URL of the repository. It changes an existing remote repository URL.

Git Remote Set:

We can change the remote URL simply by using the git remote set command. Suppose we want to make a unique name for our project to specify it. Git allows us to do so. It is a simple process. To change the remote URL, use the below command:

1. $ git remote set-url <remote name><newURL>

The **remote set-url** command takes two types of arguments. The first one is <remote name >, it is your current server name for the repository. The second argument is <newURL>, it is

your new URL name for the repository. The <new URL> should be in below format: **https://github.com/URLChanged**

Consider the below image:

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git remote set-url origin  https://github.com/URLChanged

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git remote -v
origin  https://github.com/URLChanged (fetch)
origin  https://github.com/URLChanged (push)

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$
```

In the above output, I have changed my existing repository URL as **https://github.com/URLChanged** from **https://github.com/ImDwivedi1/GitExample2**. It can be understood by my URL name that I have changed this. To check the latest URL, perform the below command:

1. $ git remote -v

 **Theoretical learning Activity**

- ✓ In group of two describe GitHub as remote version control system
- ✓ Brainstorm about commands related to remote version control systems

 Practical learning Activity

- ✓ Create account on GitHub by using your email and First_name followed by Username as username on GitHub.
- ✓ Create a new remote repository on GitHub by respecting the following Repository name by using your Lastname followed by the name of your class(ex. JeandelapaixL3SWDB)
- ✓ By using Git Clone copy your remote repository to your local machine

Points to Remember (Take home message)

- Remember that we use GitHub as remote version control for managing changes
- Before creating account on GitHub you have to create an email
- On GitHub you can create a remote repository that you can pull or clone to local repository once needed.
- Git Clone and Git Remote are two commands that are used on remote version control

Learning out come 1.1 : formative assessment

Written assessment

1. Describe version control
2. Why do you need version control as developer? Justify your answer by exact proof
3. Differentiate local repository from remote repository
4. What do you need in order to create an account on GitHub? Justify your answer
5. Is it possible to copy the remote repository to your local machine? If yes Justify it by correct steps.
6. Differentiate Git Clone from Git Remote? Which one can be used once creating a backup from remote repo to local computer?
7. Respond by True or False from the followings:
   a) Git bash is one omong version control system that exist
   b) Git init is used to clonea remote repository
   c) Git status is used to initialise an empty repository
   d) In git you can view untracked files by using git version command
9. What will be the output of the folloing git command
   Git init L3SWDA
10. When do we use the following git commands?
    a) Git Commit
    b) Git Clone
    c) Git push
    d) Git Pull
    e) Git remote
    f) Git Status
    g) Git config
    h) Git config
    i) Git rm

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.

Resources

Tools	• Git • Terminal • Web browser • Text editor (sublime text, notepad, notepad++, vscode) • GitHub
Equipment	• Computer • Network devices
Materials/ Consumables	• Internet • Electricity

Assessable outcomes	Assessment criteria (Based on performance criteria)	Indicator	Observation		Marks allocation
			Yes	No	
.Setup repositor	Git is properly initiated based on Git commands	Indicator 1. Git setup is installed			5
		Indicator 2. Git is configured			5
	Repository is properly created based on the project.	Indicator 1. GitHub account is created			5
		Indicator 2. Remote Repository is created			5
		Remote repository is cloned			5
		Untracked files are displayed			5
	Remote URL is properly set in accordance with Git commands	Indicator 1. Remote URL is generated			5
		Indicator 2. URL is configured			5

References:

https://www.nobledesktop.com/learn/git/stage-commit-files
https://www.javatpoint.com/git-branch

https://www.javatpoint.com/git-fetch

https://www.javatpoint.com/gitpull#:~:text=Git%20Pull%20Request&text=It%20allows%20reviewi ng%20commits%20before,branch%20or%20an%20existing%20bra nch.

https://www.javatpoint.com/gitpull#:~:text=Git%20Pull%20Request&text=It%20allows%20reviewi ng%20commits%20before,branch%20or%20an%20existing%20bra nch.

https://www.slideshare.net/bcbbslides/introduction-to-git-andgithub-72514916