



**git**

**CHITKARA**  
UNIVERSITY



# SOURCE CODE MANAGEMENT FILE

**Submitted by:**

**Name:** Aakansha  
Pundir

**Roll No.:** 2310992577

**Group:** 22-B

**Submitted To:**

Dr. Sharad Chauhan ,  
Professor ,CSE,  
Chitkara University,  
Punjab

**Submission of:** Task1.1

**Subject Name:** Source  
Code Management

**Sub Code:** CS181

**Department:** DCSE



# LIST OF PROGRAMS

S. No	Program Title	Page No.
<b>Task 1.1(Week 4)</b>		
1	Setting up of Git Client	3-6
2	Setting up GitHub Account	7-8
3	Generate logs	9-10
4	Create and visualize branches	11-13
5	Git life cycle description	14-16



**CHITKARA**  
**UNIVERSITY**  

---

**PUNJAB**

**Aim:** Setting up of Git Client

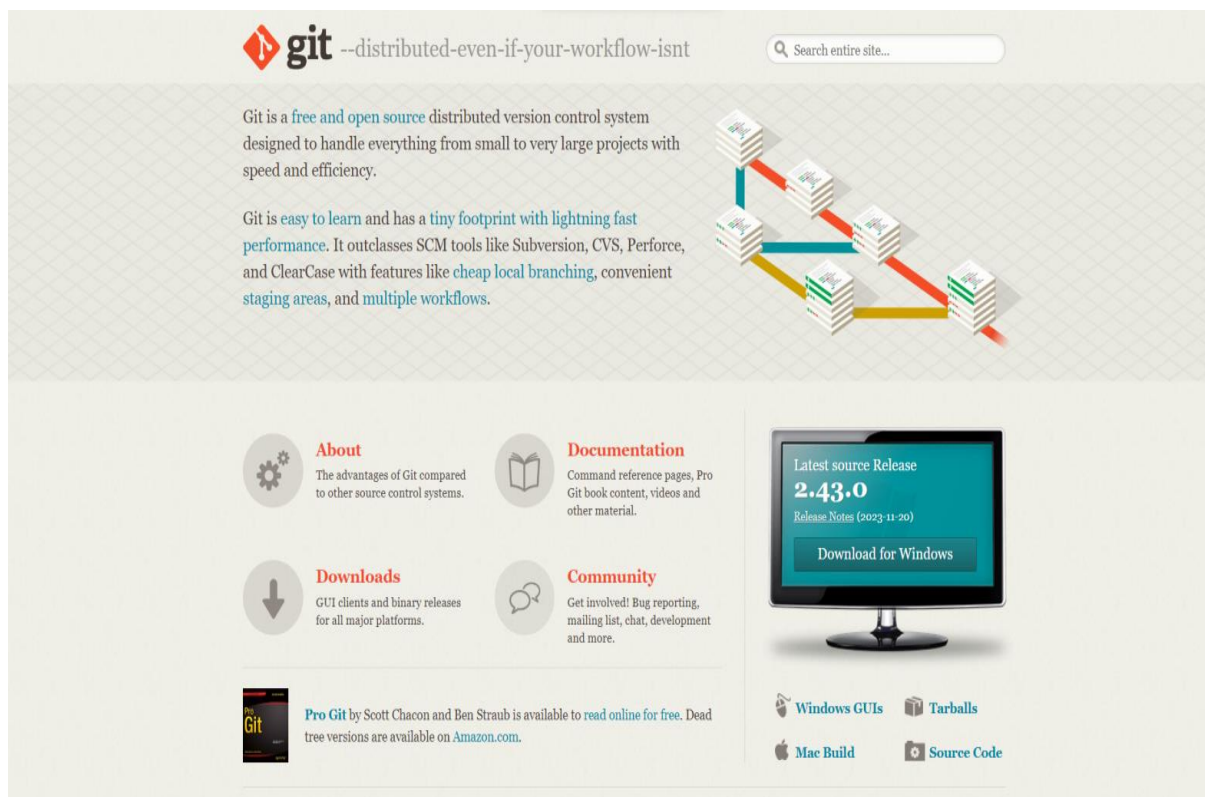
### Theory:

**GIT:** It's a Version Control System (VCS). It is a software or we can say a server by which we are able to track all the previous changes in the code. It is basically used for pushing and pulling of code. We can use git and git-hub parallelly to work with multiple members or individually. We can make, edit, recreate, copy or download any code on git hub using git.

### Procedure:

We can install Git on Windows, using the most official build which is available for download on the GIT's official website or by just typing (scm git) on any search engine. We can go on <https://git-scm.com/download/win> and can select the platform and bit-version to download. And after clicking on your desired bit-version or ios it will start downloading automatically.

### Snapshots of download:



### Installation of GIT

The screenshot shows the Git website's 'Download for Windows' page. The header features the Git logo and the tagline '--distributed-even-if-your-workflow-isnt'. A search bar is located in the top right. The left sidebar contains links for 'About', 'Documentation', 'Downloads' (highlighted), 'GUI Clients', 'Logos', and 'Community'. The main content area is titled 'Download for Windows' and provides instructions for downloading the latest (2.43.0) 64-bit version of Git for Windows. It lists various download options: 'Standalone Installer', '32-bit Git for Windows Setup', '64-bit Git for Windows Setup', 'Portable ("thumbdrive edition")', '32-bit Git for Windows Portable', and '64-bit Git for Windows Portable'. It also includes a section for 'Using winget tool' with a command prompt example and a note about the current source code release. At the bottom, there are three circular icons representing a book, a mouse, and a speech bubble.

**git** --distributed-even-if-your-workflow-isnt

Search entire site...

**About**  
**Documentation**  
**Downloads**  
GUI Clients  
Logos  
**Community**

The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on [Amazon.com](https://amazon.com).

## Download for Windows

[Click here to download](#) the latest (2.43.0) 64-bit version of **Git for Windows**. This is the most recent [maintained build](#). It was released **2 months ago**, on 2023-11-20.

**Other Git for Windows downloads**

**Standalone Installer**  
[32-bit Git for Windows Setup](#).  
[64-bit Git for Windows Setup](#).

**Portable ("thumbdrive edition")**  
[32-bit Git for Windows Portable](#).  
[64-bit Git for Windows Portable](#).

**Using winget tool**  
Install **winget** tool if you don't already have it, then type this command in command prompt or Powershell.  

```
winget install --id Git.Git -e --source winget
```

  
The current source code release is version **2.43.0**. If you want the newer version, you can build it from [the source code](#).

**Now What?**  
Now that you have downloaded Git, it's time to start using it.

Icons: Book, Mouse, Speech bubble

### Download for Windows

The screenshot shows the 'Git 2.43.0 Setup' window. The title bar includes the Git logo and standard window controls. The main heading is 'Select Components' with the question 'Which components should be installed?'. Below this, a list of components is shown with checkboxes. Most components are selected with blue checkmarks. At the bottom, a message states 'Current selection requires at least 320.8 MB of disk space.' and there are 'Back', 'Next', and 'Cancel' buttons.

**Git 2.43.0 Setup**

**Select Components**  
Which components should be installed?

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

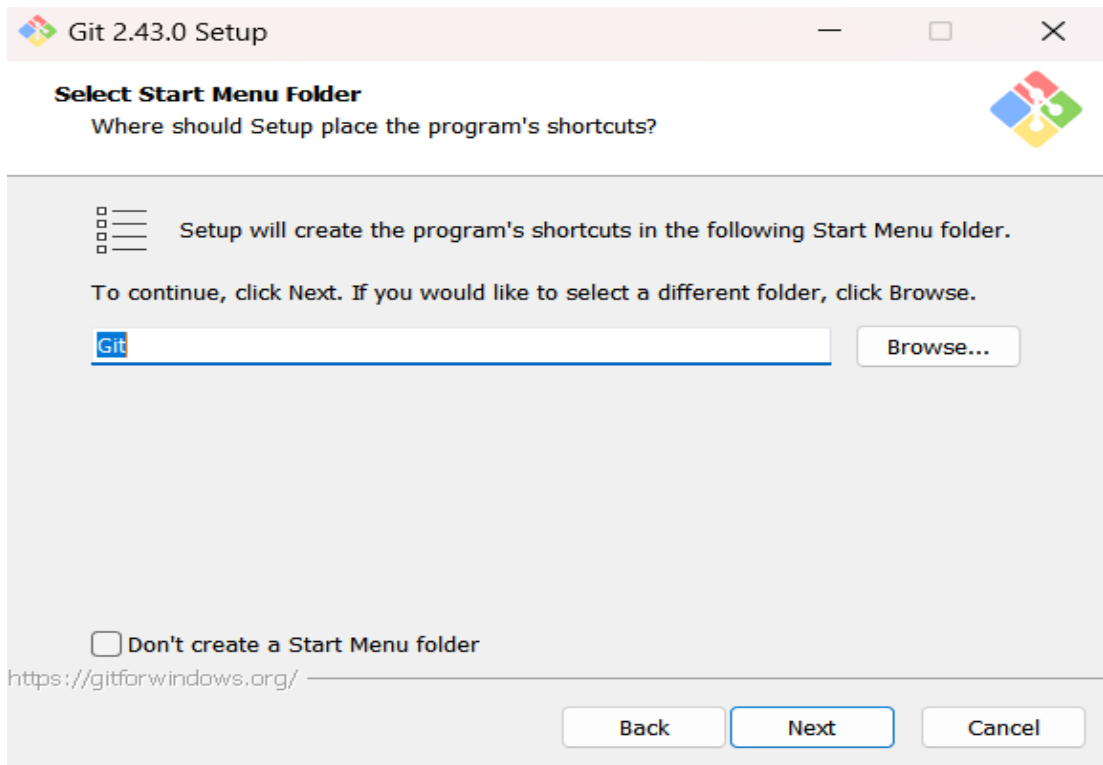
- ☐ Additional icons
  - ☐ On the Desktop
- ☒ Windows Explorer integration
  - ☒ Open Git Bash here
  - ☒ Open Git GUI here
- ☒ Git LFS (Large File Support)
- ☒ Associate .git\* configuration files with the default text editor
- ☒ Associate .sh files to be run with Bash
- ☐ Check daily for Git for Windows updates
- ☐ (NEW!) Add a Git Bash Profile to Windows Terminal

Current selection requires at least 320.8 MB of disk space.

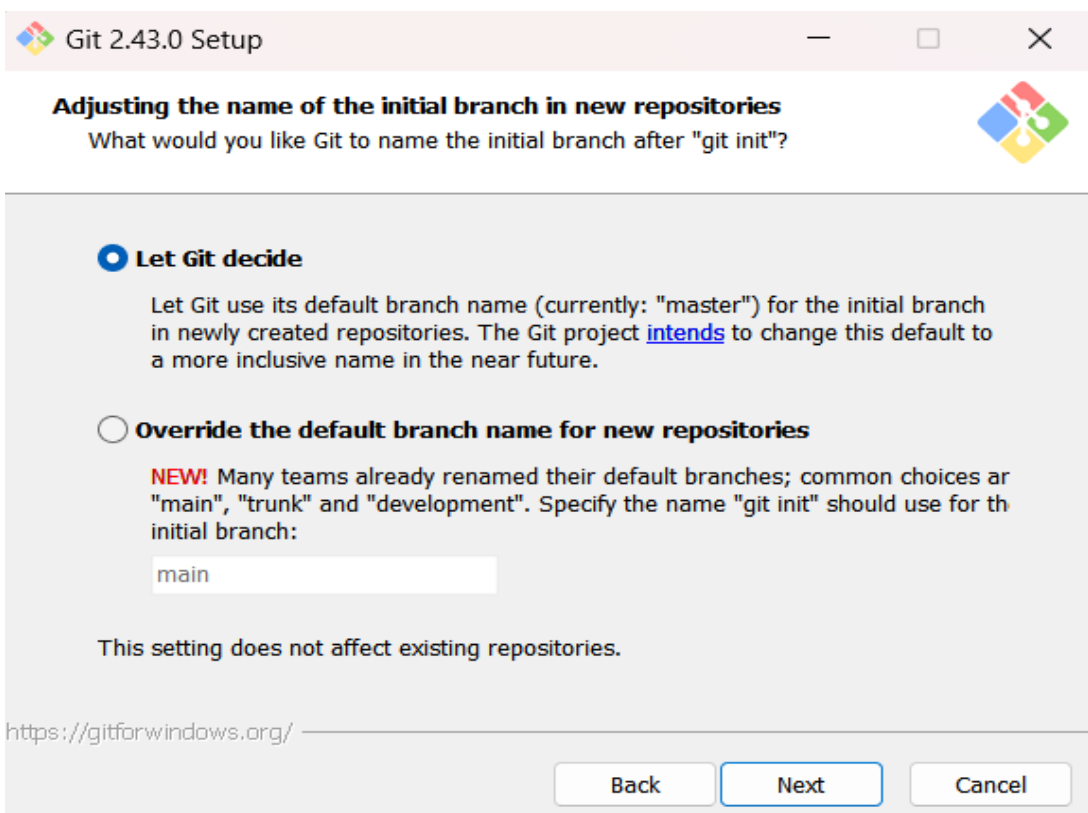
<https://gitforwindows.org/>

Back Next Cancel

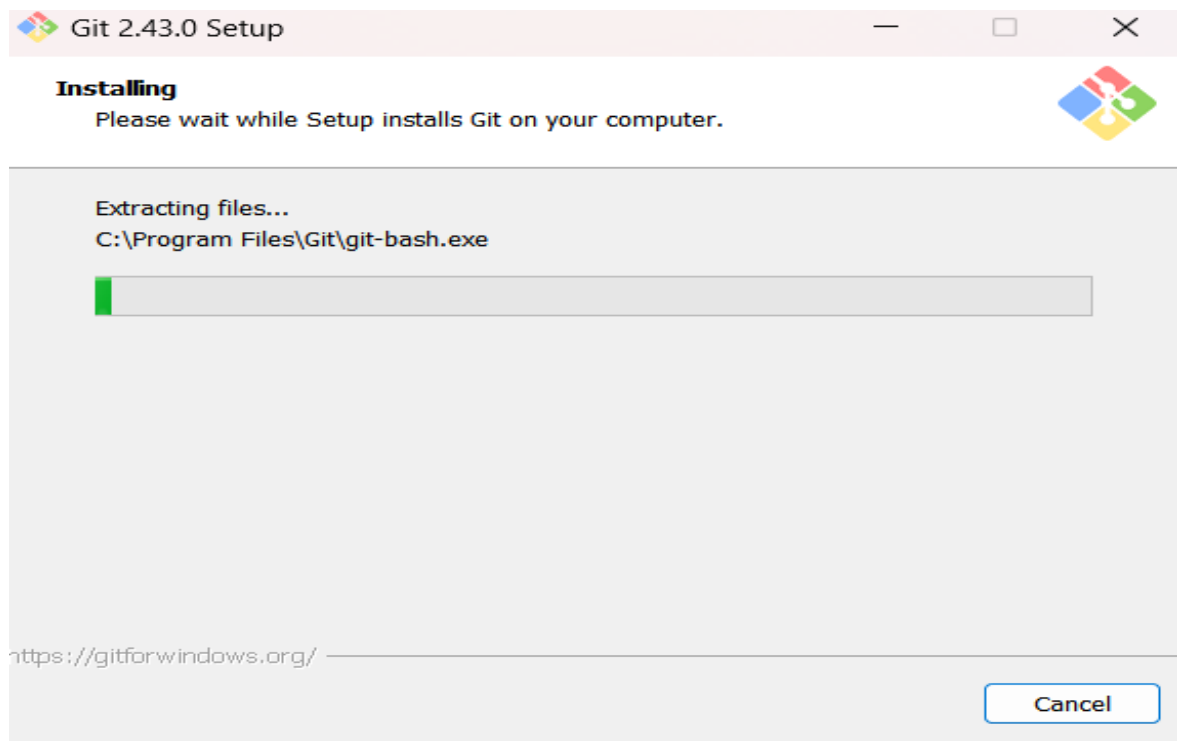
### Select Components



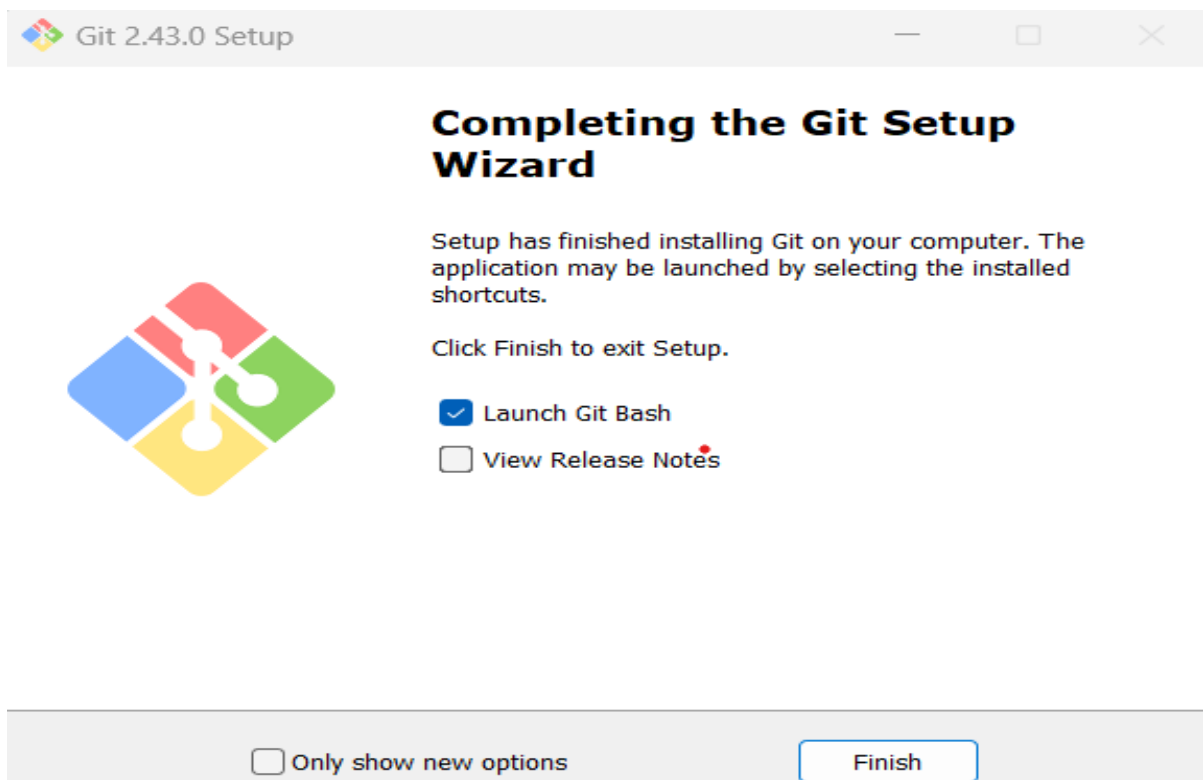
Select Star Menu Folder



Adjusting the name of the initial branch in new repositories



Installing



Completing the Git Setup Wizard

**Aim:** Setting up GitHub Account

**Theory:**

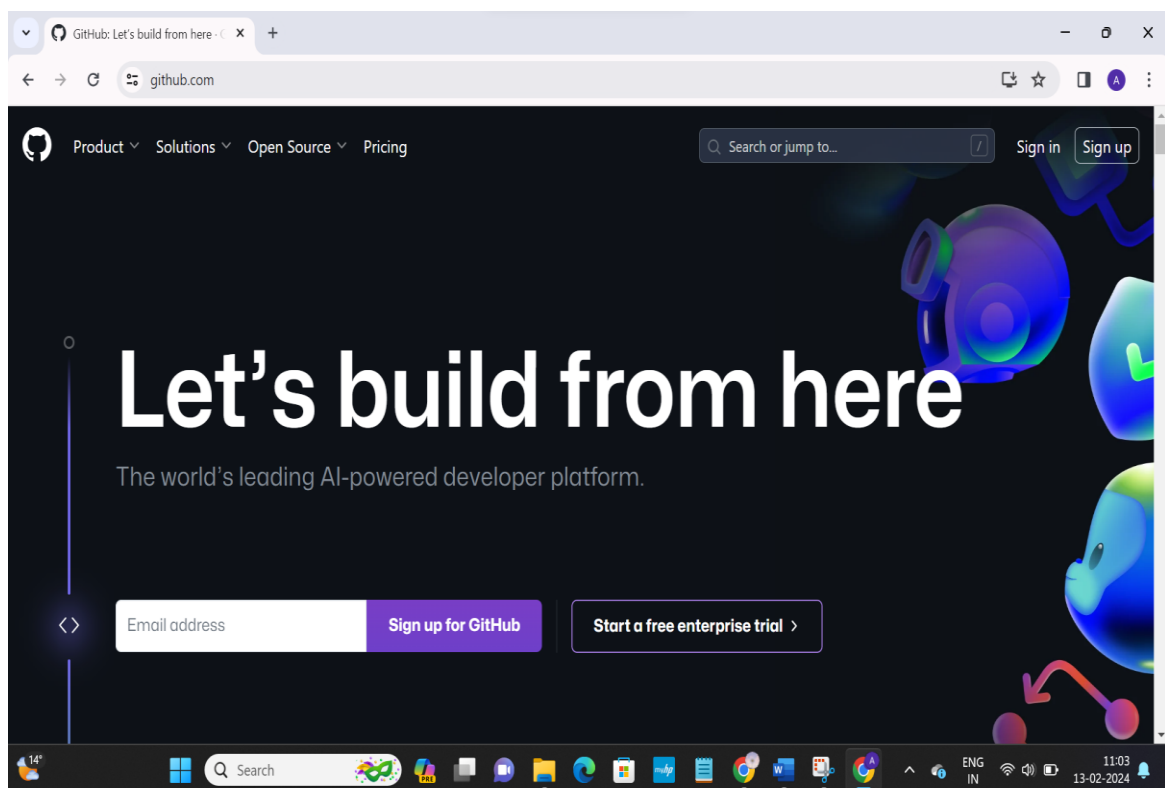
GitHub: GitHub is a website and cloud-based service (client) that helps an individual or developers to store and manage their code. We can also track as well as control changes to our or public code.

Advantages of GitHub: GitHub has a user-friendly interface and is easy to use. We can connect the git-hub and git but using some commands shown below in figure 001. Without GitHub we cannot use Git because it generally requires a host and if we are working for a project, we need to share it with our team members, which can only be done by making a repository. Additionally, anyone can sign up and host a public code repository for free, which makes GitHub especially popular with open-source projects.

**Procedure:**

To make an account on GitHub, we search for GitHub on our browser or visit <https://github.com/signup>. Then, we will enter our mail ID and create a username and password for a GitHub account.

**Snapshots :**



Welcome to GitHub

pundir.aakansha5@gmail.com|

Sign up for GitHub

Sign up for GitHub

The screenshot displays the GitHub Home dashboard. At the top, the navigation bar includes the GitHub logo, the word "Dashboard", a search bar with the placeholder "Type / to search", and several utility icons. The main content area is divided into three columns. The left column, titled "Top Repositories", features a "New" button and a search bar "Find a repository...". Below this, it lists repositories for the user "AakanshaPundir": "pundir", "akansha", and "2577". The "Recent activity" section below lists actions across GitHub. The middle column, titled "Home", contains a "Start writing code" button and a "Send feedback" link. It also features a "Filter" button with a count of 8. The main content of this column is a "Start a new repository for AakanshaPundir" section, which includes a description of a repository, a "Repository name \*" input field, and radio buttons for "Public" and "Private" visibility. A "Create a new repository" button is at the bottom. To the right of this is a "Introduce yourself with a profile README" section, which includes a "Create" button and a list of prompts for a README file. The right column, titled "Latest changes", lists recent updates from the community, such as "Secret scanning adds webhook support for validity checks" and "Enhanced Codespaces Connection". Below this is an "Explore repositories" section featuring the repository "agda / agda". The bottom of the image shows a Windows taskbar with the date "13-02-2024" and time "10:59".

Dashboard

Type / to search

Home

Send feedback

Filter 8

Start writing code

Start a new repository for AakanshaPundir

A repository contains all of your project's files, revision history, and collaborator discussion.

Repository name \*

name your new repository...

☐ Public

Anyone on the internet can see this repository

☒ Private

You choose who can see and commit to this repository

Create a new repository

Introduce yourself with a profile README

Share information about yourself by creating a profile README, which appears at the top of your profile page.

AakanshaPundir / README.md

Create

1 - Hi, I'm @AakanshaPundir

2 - I'm interested in ...

3 - I'm currently learning ...

4 - I'm looking to collaborat

5 - How to reach me ...

6 - Pronouns: ...

7 - Fun fact: ...

8

Latest changes

7 hours ago

Secret scanning adds webhook support for validity checks

9 hours ago

Enhanced Codespaces Connection

11 hours ago

GitHub Issues & Projects - Projects without Limits Private Beta

12 hours ago

VS Code Copilot Chat January 2024 (version 0.12)

View changelog →

Explore repositories

agda / agda

Agda is a dependently typed programming language / interactive theorem prover.

14°C Partly sunny

Search

ENG IN

10:59 13-02-2024

GitHub Interface



## Experiment No. 03

**Aim:** Program to Generate log

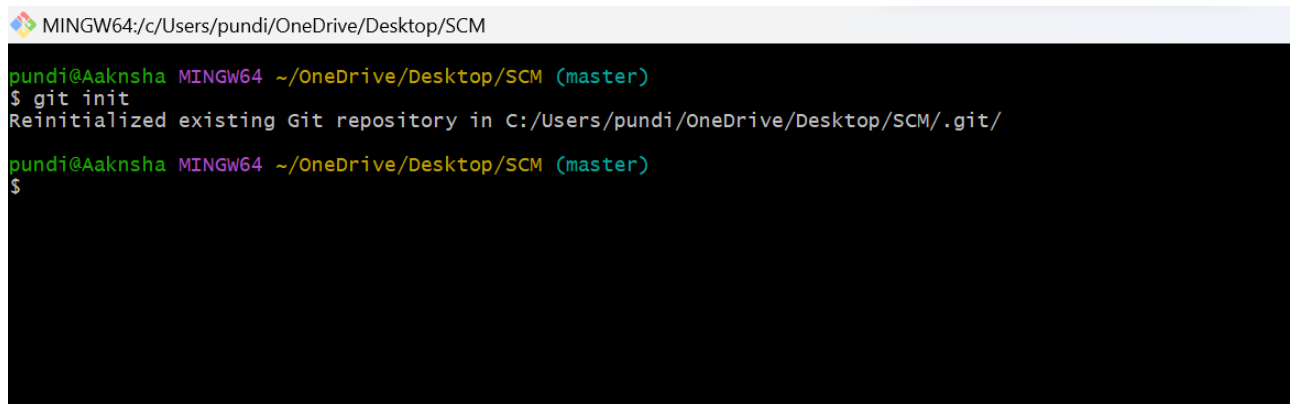
### Theory:

Logs: Logs are nothing but the history which we can see in git by using the code git log.

It contains all the past commits, insertions and deletions in it which we can see any time. Logs helps to check that what were the changes in the code or any other file and by whom. It also contains the number of insertions and deletions including at which time it was changed.

### Procedure:

First of all, create a local repository using Git. For this, you have to make a folder in your device, right click and select “Git Bash Here”. This opens the Git terminal. To create a new local repository, use the command “git init” and it creates a hidden folder “.git”.



```
MINGW64:/c:/Users/pundi/OneDrive/Desktop/SCM
pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git init
Reinitialized existing Git repository in C:/Users/pundi/OneDrive/Desktop/SCM/.git/
pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$
```

When we use GIT for the first time, we have to give the user name and email so that if I am going to change in project, it will be visible to all.

For this, we use command:

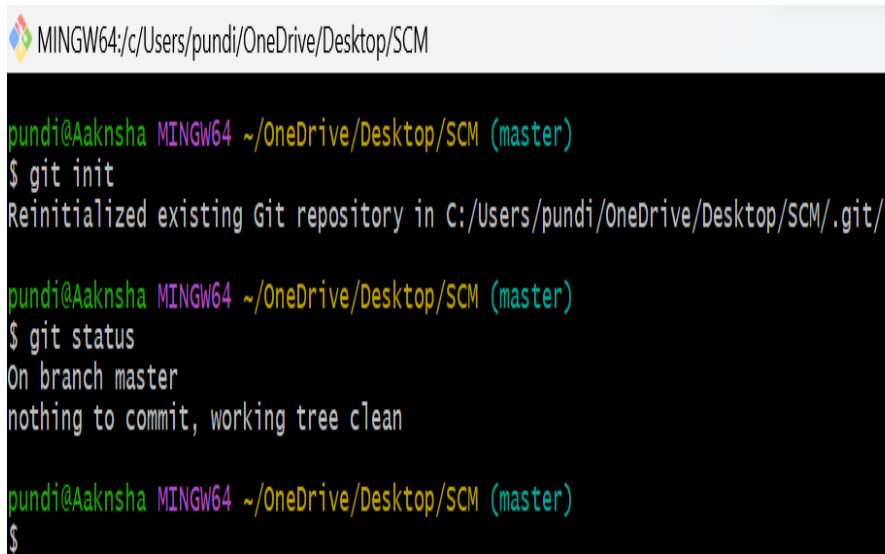
```
“git config --global user.name Name”
“git config --global user.email email”
```

For verifying the user’s name and email, we use:

```
“git config --global user.name”
“git config --global user.email”
```

## SOME IMPORTANT COMMANDS

- Is :- It gives the file names in the folder.
- Is -lart :- Gives the hidden files also.
- Git status :- Displays the status of the working directory and the staged snapshot.
- Touch filename :- This command creates a new file in the repository.
- Clear :- It clears the terminal.
- Rm -rf .git :- It removes the repository.
- Git log :- Displays all of the commits in a repository's history.
- Git diff :- It compares my working tree to staging area.



```
MINGW64:/c/Users/pundi/OneDrive/Desktop/SCM

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git init
Reinitialized existing Git repository in C:/Users/pundi/OneDrive/Desktop/SCM/.git/

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git status
On branch master
nothing to commit, working tree clean

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$
```

Git status

```

MINGW64:/c:/Users/pundi/OneDrive/Desktop/SCM
$ git init
Reinitialized existing Git repository in C:/Users/pundi/OneDrive/Desktop/SCM/.git/

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git status
On branch master
nothing to commit, working tree clean

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git log
commit 18cdb2de66dd37d5f4b20fac71ef90c1bab450b1 (HEAD -> master)
Author: Aakansha Pundir <pundir.aakansha5@gmail.com>
Date: Thu Feb 1 13:48:19 2024 +0530

    commit4:Fourth

commit 9c5387df65a60277e881b6cb479d920183195216
Author: Aakansha Pundir <pundir.aakansha5@gmail.com>
Date: Thu Feb 1 13:47:11 2024 +0530

    commit-3:Third

commit b2925c31c5c0f88f133a6f3a63e23d89f1555ed3
Author: Aakansha Pundir <pundir.aakansha5@gmail.com>
Date: Thu Feb 1 13:45:24 2024 +0530

    commit-2:Second

commit c338dbcc5c73bda65f44ece9bceb81cfe54c56bd
Author: Aakansha Pundir <pundir.aakansha5@gmail.com>
Date: Thu Feb 1 13:38:11 2024 +0530

    commit-1:Initial

commit 7a4f8f295835f0234feedba54785cdac3dbb97aa
Author: Aakansha Pundir <pundir.aakansha5@gmail.com>
Date: Wed Jan 31 11:22:00 2024 +0530

    commit-2:Second phase

commit df246dc8eec4535a8d21f14cee133fdadf231577
Author: Aakansha Pundir <pundir.aakansha5@gmail.com>
Date: Wed Jan 31 11:15:20 2024 +0530

    commit-1: Initial

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$

```

## Git log

The git log command displays a record of the commits in a Git repository. By default, the git log command displays a commit hash, the commit message and other commit metadata

**Aim:** Create and visualize branches

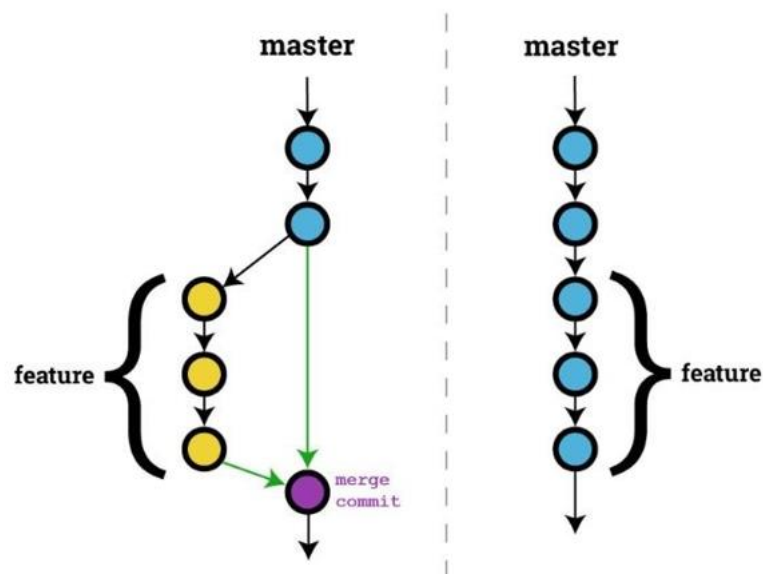
### Theory:

Branching: A branch in Git is an independent line of work (a pointer to a specific commit). It allows users to create a branch from the original code (master branch) and isolate their work. Branches allow you to work on different parts of a project without impacting the main branch.

Create branches: The main branch in git is called as master branch. But we can make branches out of this main master branch. All the files present in master can be shown in branch but the file which are created in branch are not shown in master branch. We can also merge both the parent (master) and child (other branches).

Syntax:

For creating a new branch, git branch name by default is master branch.



## Snapshots:

```
MINGW64:/c/Users/pundi/OneDrive/Desktop/SCM

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git branch
* master

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$
```

Default branch is master branch

```
MINGW64:/c/Users/pundi/OneDrive/Desktop/SCM

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git branch
* master

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git branch feature

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git branch
feature
* master

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$
```

Adding a feature branch

```
MINGW64:/c/Users/pundi/OneDrive/Desktop/SCM

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git branch
* master

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git branch feature

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git branch
feature
* master

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git checkout feature
Switched to branch 'feature'

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (feature)
$ git branch
feature
* master

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (feature)
$ |
```

Switching to feature branch

```
MINGW64:/c/Users/pundi/OneDrive/Desktop/SCM

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git branch
* master

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git branch feature

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git branch
  feature
* master

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git checkout feature
Switched to branch 'feature'

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (feature)
$ git branch
* feature
  master

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (feature)
$ git checkout master
Switched to branch 'master'

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ |
```

Switching to master branch

```
pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (feature)
$ git branch
* feature
  master

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (feature)
$ git checkout master
Switched to branch 'master'

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git log --oneline
18cdb2d (HEAD -> master, feature) commit4:Fourth
9c5387d commit-3:Third
b2925c3 commit-2:Second
c338dbc commit-1:Initial
7a4f8f2 commit-2:Second phase
df246dc commit-1: Initial

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$
```

Checking commits via log

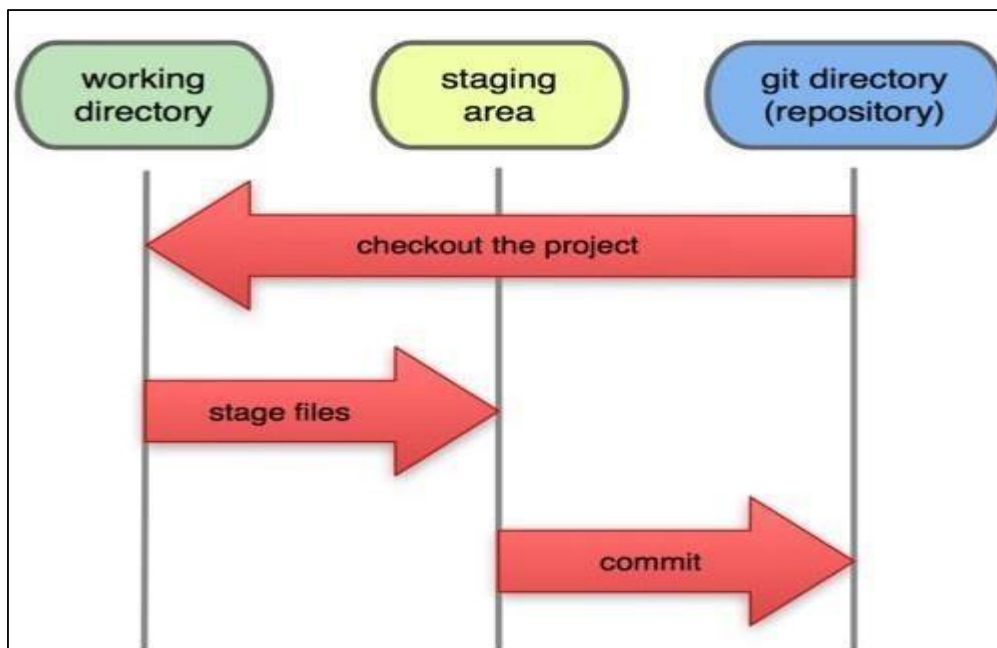


**Aim:** Git lifecycle description

**Theory:**

Stages in GIT Life Cycle: Files in a Git project have various stages like Creation, Modification, Refactoring, and Deletion and so on. Irrespective of whether this project is tracked by Git or not, these phases are still prevalent. However, when a project is under Git version control system, they are present in three major Git states in addition to these basic ones. Here are the three Git states:

- Working directory
- Staging area
- Git directory



**Working Directory:**

Consider a project residing in your local system. This project may or may not be tracked by Git. To track the files in the directory we need to initialize the repository by using '*git init*' command. In either case, this project directory is called your Working directory.

**Staging Area:**

Staging area is the playground where you group, add and organize the files to be committed to Git for tracking their versions. Files are added into the staging area by using '*Git add "filename"*' command.

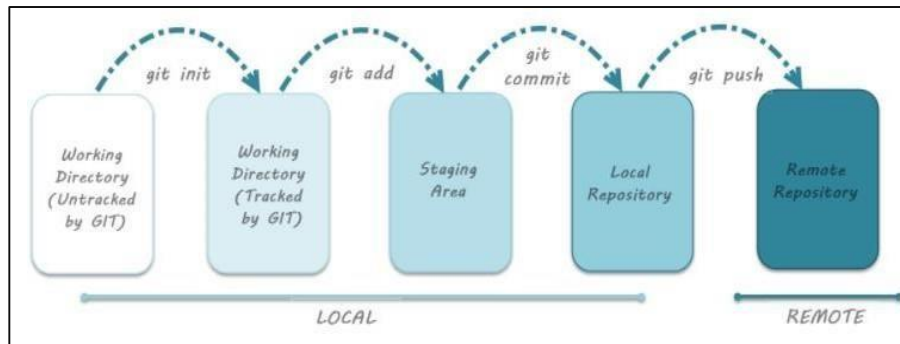


## Git Directory:

Now that the files to be committed are grouped and ready in the staging area, we can commit these files. So, we commit this group of files along with a commit message explaining what is the commit about. Apart from commit message, this step also records the author and time of the commit. Now, a snapshot of the files in the commit is recorded by Git. The information related to this commit is stored in the Git directory. Files are added into directory in unmodified state by using '*git commit*' command, this will open the VIM editor where user can enter the commit message by using a few key combinations mainly '*i*' enter the message and then '*:wq*' to exit the VIM editor.

## Remote Repository:

It means mirror or clone of the local Git repository in GitHub. And pushing means uploading the commits from local Git repository to remote repository hosted in GitHub.



## File Status Lifecycle

