InsightsNet GitRepo Manager: User Guide

Welcome to **InsightsNet GitRepo Manager**. This tool, currently in beta, has been developed by the InsightsNet team at the Technical University of Darmstadt. It aims to make Git more accessible to bachelor's and master's students, allowing them to integrate version control into their research workflow without risk of damaging the repository or spending significant time learning command-line Git.

While InsightsNet GitRepo Manager doesn't cover every feature available through the terminal, we still believe the command line is the most powerful way to use Git. Nevertheless, we recognize that many students find the terminal intimidating, which prevents them from incorporating Git into their research. Our tool bridges that gap by providing essential Git functionalities in a friendly interface.

InsightsNet GitRepo Manager is a cross-platform tool built with Python 3.12 or higher, GitPython (3.1.45), Pillow (11.3.0), and Tkinter. A standalone Windows executable is also available as *InsightsNetGitRepoManager.exe*. Note that Git must be installed on your system for the tool to function correctly.

At present, the tool supports three primary functions: **Clone Repository**, **Branch Management**, and **Commit Changes**. It works with both GitHub and GitLab repositories via HTTPS or SSH URLs.

System Requirements:

1. **Operating System:** Windows 11, Linux, or macOS.
2. **Git:** A distributed version control system must be installed.
3. **Python:** Version 3.12 or higher.
4. **gitPython:** Version 3.1.45.
5. **Pillow:** Version 11.3.0.

Installation Guidelines (Windows 11 / Linux / macOS):

1. **Install Git:**
    - **Windows:** Visit git-scm.com and download the latest version of Git (the current release is 2.52.0 as of January 27, 2026). Follow the installation instructions.

- **Windows PowerShell + Winget:**

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

- **Linux and macOS:** These systems often come with Git preinstalled. If not, visit the Git website to download and install it, then open your terminal and follow the installation commands.
  - **Linux:** *apt-get install git*
  - **macOS:** *brew install git*

2. **Verify and Install Python:**
   - Check whether Python 3 is installed by running python --version in your terminal. You should have version 3.12 or higher.
   - If Python is not installed, download it from python.org and follow the installation instructions.
3. **Download GitRepo Manager:**
   - Visit the appropriate download location for the GitRepo Manager project (replace "XXXX" with the actual URL).
   - Download the repository as a .zip file and extract it to a directory of your choice.
4. **Create a Python Virtual Environment:**
   - Open your terminal and navigate to the directory where you wish to set up the virtual environment.
   - Run the following command, substituting YOUR_ENV_NAME with your preferred name:

     python -m venv YOUR_ENV_NAME
   - This creates an isolated environment, so the tool's dependencies won't interfere with your global Python installation.
5. **Set Up the Project:**
   - Copy all files from the extracted "InsightsNet GitRepo Manager" folder into your virtual environment directory.
   - Activate the virtual environment:
     - **Windows:** *YOUR_ENV_NAME\Scripts\activate*
     - **Linux/macOS:** *source YOUR_ENV_NAME/bin/activate*

- Install the required packages by running:

  ```
  pip install -r requirements.txt
  ```

6. **Run the Application:**
   - Execute the application using:

     ```
     python app.py
     ```

   - This command launches the graphical interface for InsightsNet GitRepo Manager.

Note:
If you prefer to use the precompiled Windows executable (**InsightsNetGitRepoManager.exe**), simply download it and double-click to launch the application. Ensure that Git is installed on your system by following the steps in the "Install Git" section above.

Visual Installation Guideline

Please follow each step in order:

Step 1: Install the prerequisites

1. **Install Git**:
   Download the latest version from git-scm.com and complete the installation.

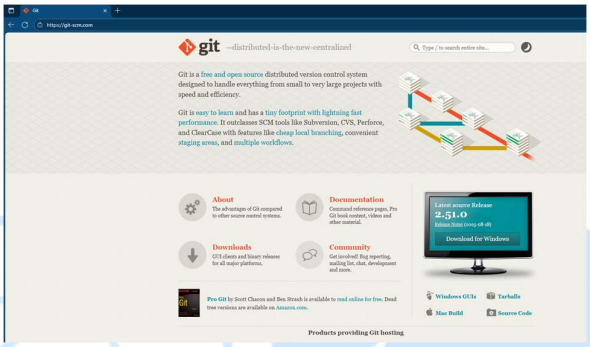


Figure 1: Git Source code management (SCM).

2. **Install Python 3** (version 3.12 or higher):
   Visit the official Python website at python.org and download the installer for your operating system.

Figure 2: Python Official Website

Step 2: Download the project

- Navigate to the InsightsNet GitHub repository (LINK) or the InsightsNet NextHessen Box (LINK) and download the project as a .zip file.
- Extract the contents of the .zip file to a location of your choice.

Step 3: Create a Python virtual environment

1. Open your terminal (PowerShell on Windows; Terminal on Linux or macOS) and change the directory to where you want to create the virtual environment.
2. Run the following command, replacing YOUR_VENV_NAME with a name of your choice:

```
python -m venv YOUR_VENV_NAME
```

This command creates a folder named YOUR_VENV_NAME that contains your isolated virtual environment. Leave this folder intact.

3. Copy all files extracted in Step 2 into the same location that contains your new virtual environment.

Step 4: Activate the virtual environment and install dependencies

1. Change your terminal's directory to the location of the virtual environment.
2. Activate the environment:
   - **Windows**: *YOUR_VENV_NAME\Scripts\activate*
   - **Linux/macOS**: *source YOUR_VENV_NAME/bin/activate*

When the environment is activated, you will see (YOUR_VENV_NAME) appear in your terminal prompt.

3. Install the required Python packages by running:

```
pip install -r requirements.txt
```

This command installs all necessary dependencies for InsightsNet GitRepo Manager. Once completed, the installation and setup are finished.

Step 5: Launch InsightsNet GitRepo Manager

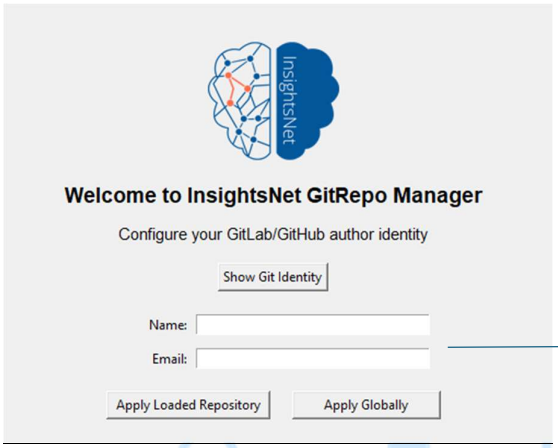
With everything configured, you can start using InsightsNet GitRepo Manager. Run one of the following commands:

```
python app.py
```

or

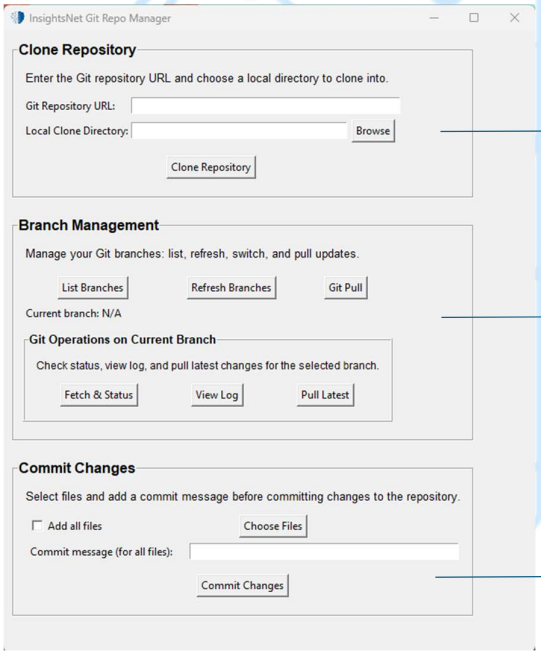
```
python3 app.py
```

This will open the graphical interface for the tool.



Use the name and email exactly as shown in your GitHub or GitLab profile settings.

Figure 3: GitHub or GitLab authentication.



To clone any repository from GitHub or GitLab

To load your already cloned repository.

Manage git brunches, git status and pull new updates from remote git.

Add single or multiple files into git repository and commit.

Figure 4: InsightsNet Git Repo Manager.

Clone Repository

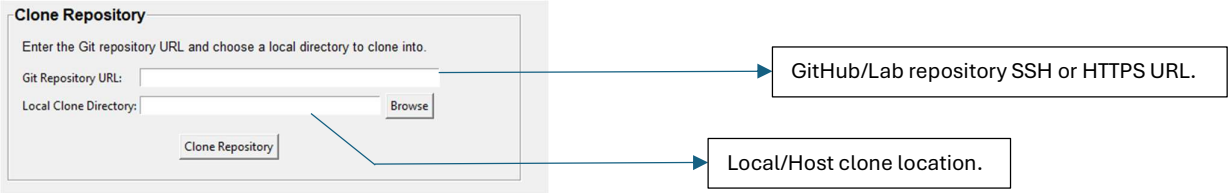


Figure 5: To Clone a remote repository.

To clone a repository from GitHub or GitLab to your local machine:

1. Enter the repository's SSH or HTTPS URL in the **Git Repository URL** field.
2. Select the **Local Clone Directory**, which is the folder on your machine where you want to store the repository.
3. Click **Clone Repository**. The tool will clone the repository to the specified directory.

Branch Management

This section helps you track and manage branches in your project.

- **List Branches** – Displays all branches in your project.
- **Refresh Branches** – Updates the branch list to ensure it reflects the current state of the repository.
- **Git Pull** – Fetches the latest changes from the remote repository into your current branch.
- **Current branch: <branch-name>** – Shows which branch you are currently working on.
- **Branch Dropdown** – Allows you to switch to a different branch.

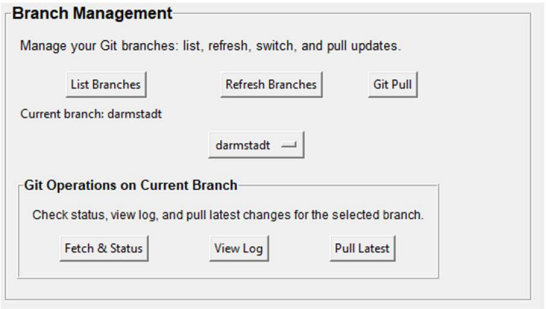


Figure 6: Git branch management, remote repository status and pull the latest update from remote.

Git Operations on Current Branch

- **Fetch & Status** – Checks for updates from the remote repository and shows the status of your files (modified, staged, untracked, etc.).
- **View Log** – Displays the commit history for the current branch.
- **Pull Latest** – Updates your current branch with the newest changes from the remote repository.

Commit Changes

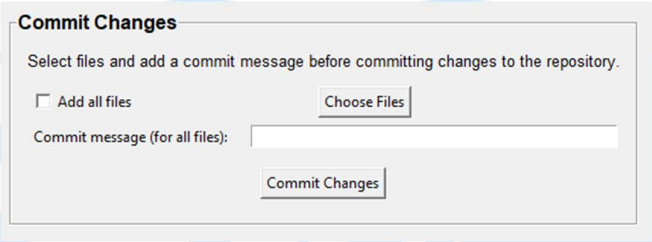


Figure 7: Add and commit file/files into the remote repository.

The **Commit Changes** section lets you prepare and save your work:

- **Add all files** – Selects all modified files to be included in the commit.
- **Choose Files** – Allows you to pick specific files to commit.
- **Commit message (for all files)** – A short note describing what has changed (e.g., "Fixed typo in README").
- **Commit Changes** – Saves your selected changes with the commit message into the project history and pushes the commit to the remote repository (GitHub or GitLab).

Push Changes

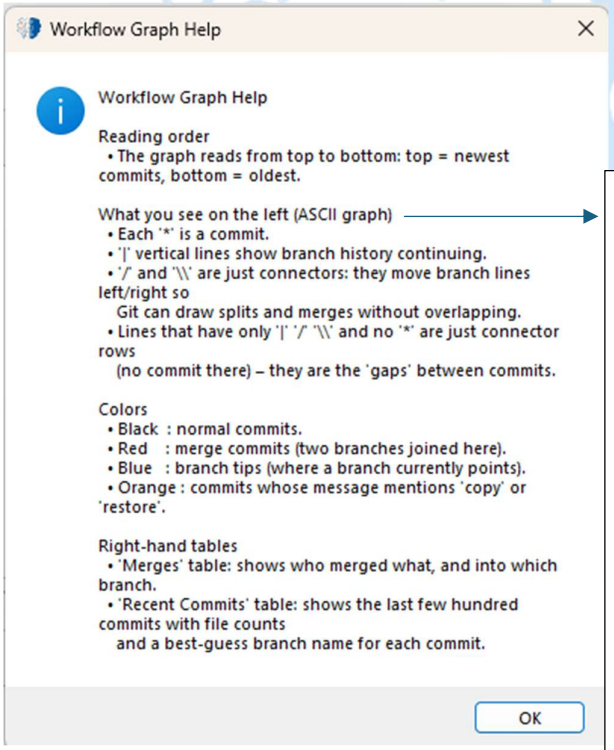
- After commit, due to any technical reason, the local repository is ahead of the remote repository and the commit shows no file changes, using the **Git Push** option will still manually push the commit to the remote repository.

Workflow Graph

Workflow Graph shows,

- What happened
- When it happened
- How branches came together or split apart

- The workflow graph flows from top to bottom.
  - Top = newest commits
  - Bottom = oldest commits



Star (*) → A commit
Each star represents one saved change in the project.

Vertical lines (|) → A branch continuing
These show the history of a branch moving forward.

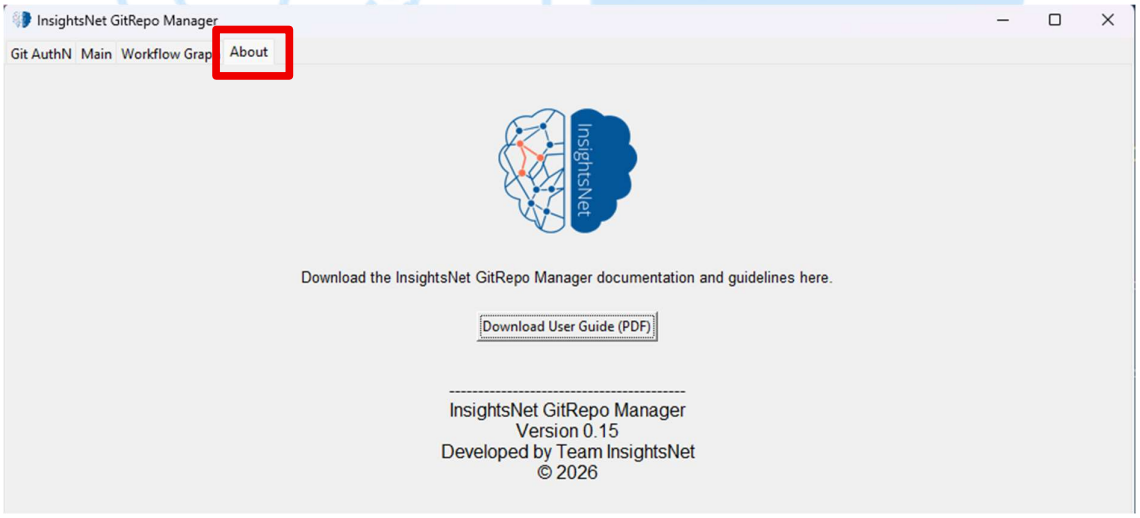
Slanted lines (/) and (\ ) → Branch splits or merges
These connect branches when work splits apart or comes back together.

Empty space (no *) → Just spacing
These gaps help Git draw merges and splits cleanly. No commits are missing.

Commit Colores and Tables

- ⚫ **Black → Normal commits**
- 🔴 **Red → Merge commits**
  (Two branches were combined here)
- 🔵 **Blue → Branch tips**
  (The latest commit on a branch)
- 🟠 **Orange → Special commits**
  (Commits mentioning things like copy or restore)
- **Merges table**
  Shows who merged changes, what was merged, and which branch it went into.
- **Recent Commits table**
  Shows the most recent commits, how many files changed, and Git's best guess at which branch each commit belongs to.

Documentation / User guideline (About)



Appendix:

Remote Tools

- **Show Origin URL**

  📌 What it does: Displays the main remote repository URL for this project (usually called origin).

  💡 When to use it: Use this when you want to check where your code is pushing to or confirm which remote repository you're connected to.

- **Switch Origin to SSH**

  📌 What it does: Converts the current remote URL from HTTPS to SSH format. If it's already using SSH, it'll just let you know.

  💡 When to use it: Use this if you're tired of entering your username/password or want to use SSH keys for smoother authentication.

- **Test GitHub SSH**

  📌 What it does: Checks whether your SSH key is correctly set up for GitHub.

  💡 When to use it: Use this if you're having trouble pushing or cloning from GitHub and want to confirm your SSH access works.

- **Test Origin Host SSH**

  📌 What it does: Detects the host from your current remote (GitHub, GitLab, or self-hosted) and tests SSH access to it.

  💡 When to use it: Use this when you're working with a private or self-hosted Git server and want to verify SSH connectivity.

- **Test GitLab SSH**

  📌 What it does: Checks whether your SSH key is correctly set up for GitLab.

  💡 When to use it: Use this if GitLab pushes or pulls fail and you want to confirm your SSH authentication.

- **Show origin.fetch**

  📌 What it does: Displays the fetch rules (refspec) that define which branches are downloaded from the remote.

  💡 When to use it: Use this if branches aren't appearing locally and you want to see what Git is actually fetching.

- **Show Remote Heads**

  📌 What it does: Lists all branches that currently exist on the remote server.

  💡 When to use it: Use this when you know a branch exists on the server but can't see it locally.

- **Fix Refspec & Fetch All**

  📌 What it does: Fixes the fetch configuration to include all branches, then fetches everything and cleans up deleted branches.

  💡 When to use it: Use this when branches are missing, outdated, or your local repo feels out of sync with the remote.

Branch Management Section

- **List Branches**

  📌 *What it does:* Shows all branches in your project.

  💡 *When to use it:* Use this when you're not sure which branches exist or want to see what branches you can switch to.

- **Refresh Branches**

  📌 *What it does:* Updates the branch list to make sure it's current.

  💡 *When to use it:* Use this after someone else has created a new branch and you want it to appear in your list.

- **Git Pull**

  📌 *What it does:* Brings in the latest changes from the remote repository into your current branch.

  💡 *When to use it:* Use this before starting work, so you're up to date with changes from your teammates.

- **Current branch: <branch-name>**

  📌 *What it does:* Displays which branch you are currently working on.

  💡 *When to use it:* Just to confirm you're working in the correct branch before making changes.

- **Branch Dropdown**
  📌 *What it does:* Lets you switch to a different branch.
  💡 *When to use it:* Use this when you need to move to another branch (for example, to fix a bug or test a feature).

Git Operations on Current Branch

- **Fetch & Status**
  📌 *What it does:* Checks for updates from the remote repository and shows the status of your files (modified, staged, untracked, etc.).
  💡 *When to use it:* Use this if you want to check whether your branch is behind the remote or see if you have local changes.
- **View Log**
  📌 *What it does:* Shows the history of commits made on the branch.
  💡 *When to use it:* Use this to review what changes have been made in the branch over time.
- **Pull Latest**
  📌 *What it does:* Updates your current branch with the newest changes from the remote repository.
  💡 *When to use it:* Use this often, especially before committing or pushing, so you avoid conflicts with others' work.

Commit Changes Section

- **Add all files (checkbox)**
  📌 *What it does:* Selects all the changed files in your project to be included in the commit.
  💡 *When to use it:* Use this if you want to commit every change you've made at once.
- **Choose Files**
  📌 *What it does:* Lets you pick specific files you want to include in the commit.
  💡 *When to use it:* Use this if you only want to commit certain changes, not everything.
- **Commit message (for all files)**
  📌 *What it does:* A short note describing what changes you made.

💡 *When to use it:* Always write a clear and simple message here so others (and future you) know why the change was made. Example: *"Fixed login button issue"*.

- **Commit Changes (button)**

  📌 *What it does:* Saves your selected changes (with the message) into the Git history.

  💡 *When to use it:* Use this after selecting files and writing a commit message, to officially record your changes in the repository.