

# 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](https://git-scm.com) and download the latest version of Git (the current release is 2.51.0 as of August 22, 2025). Follow the installation instructions.

- **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.
- To check git installation, run the command in terminal,  

```
git --version
```

You should have version 2.51.0 or higher

## 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](https://python.org) and follow the installation instructions.

## 3. Download GitRepo Manager:

- Visit the InsightsNet GitHub for the GitRepo Manager project.



- Download the repository as a .zip file.
- Extract the contents of the .zip file 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.

The executable file **InsightsNetGitRepoManager.exe** is available for download via the TU Darmstadt Nextcloud (Hessen Box): [🔗 Download from Hessen Box](#)

## 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](https://git-scm.com) and complete the installation.

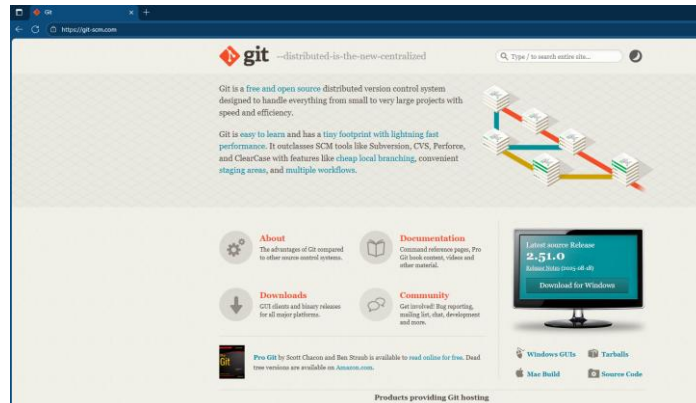


Figure 1: Git Source code management (SCM).

To check git installation, run the command in terminal,

```
git --version
```

#### 2. Install Python 3 (version 3.12 or higher):

Visit the official Python website at [python.org](https://python.org) and download the installer for your operating system.

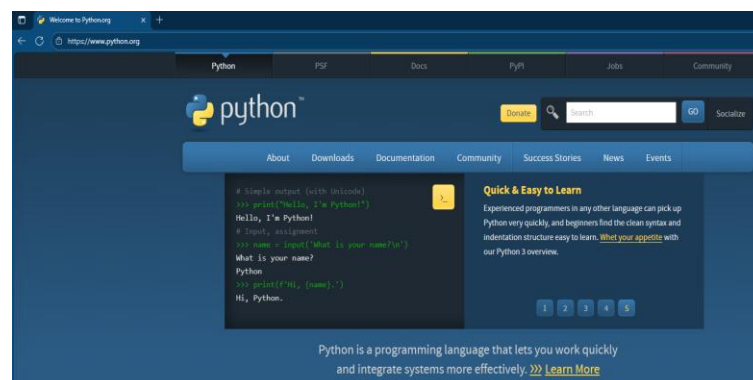


Figure 2: Python Official Website

To check python installation, run the command in terminal,

```
python --version
```

or

```
python3 --version
```

## 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:

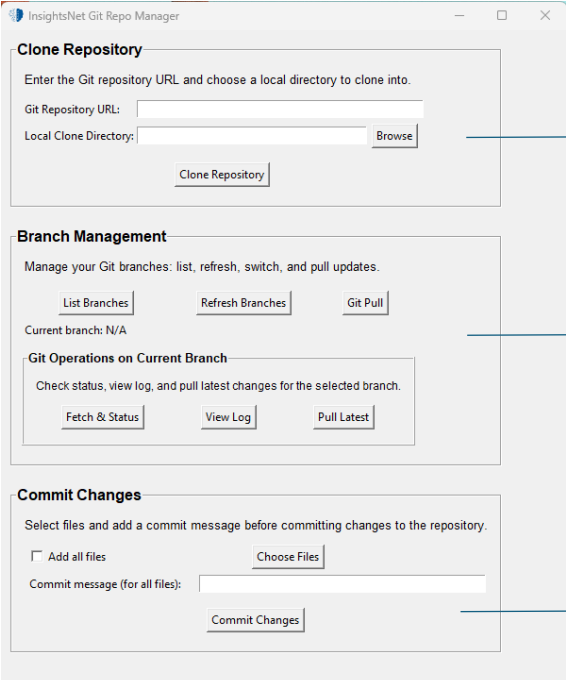
```
python app.py
```

or

```
python3 app.py
```

This will open the graphical interface for the tool.

### InsightsNet Git Repo Manager Tutorial

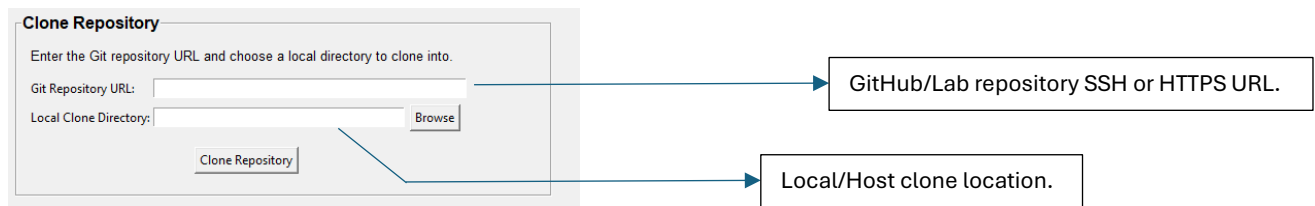


The screenshot shows the InsightsNet Git Repo Manager GUI with three main sections: Clone Repository, Branch Management, and Commit Changes. Annotations with arrows point to specific features:

- Clone Repository:** A text box for "Git Repository URL" and a "Browse" button for "Local Clone Directory". An arrow points to the "Clone Repository" button with the text: "To clone any repository from GitHub or GitLab" and "To load your already cloned repository."
- Branch Management:** Buttons for "List Branches", "Refresh Branches", and "Git Pull". An arrow points to the "Git Pull" button with the text: "Manage git branches, git status and pull new updates from remote git."
- Commit Changes:** A "Choose Files" button and a "Commit Changes" button. An arrow points to the "Commit Changes" button with the text: "Add single or multiple files into git repository and commit."

Figure 3: InsightsNet Git Repo Manager.

## Clone Repository



The diagram shows a 'Clone Repository' form with two input fields: 'Git Repository URL' and 'Local Clone Directory'. The 'Git Repository URL' field is linked to a box labeled 'GitHub/Lab repository SSH or HTTPS URL.'. The 'Local Clone Directory' field is linked to a box labeled 'Local/Host clone location.'. A 'Browse' button is next to the 'Local Clone Directory' field. A 'Clone Repository' button is at the bottom of the form.

Figure 4: To Clone a remote repository.

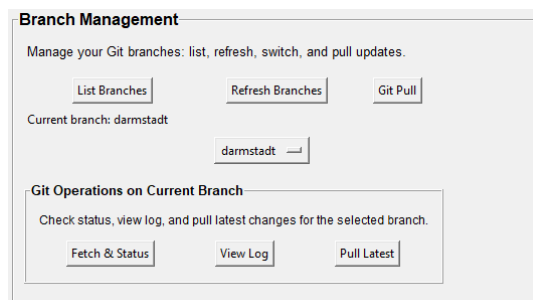
To clone a repository from GitHub or GitLab to your local machine:

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.



The diagram shows a 'Branch Management' form with the following elements:
 

- Buttons: 'List Branches', 'Refresh Branches', and 'Git Pull'.
- Text: 'Current branch: darmstadt'.
- Dropdown menu: A dropdown menu showing 'darmstadt'.
- Section: 'Git Operations on Current Branch'.
- Text: 'Check status, view log, and pull latest changes for the selected branch.'
- Buttons: 'Fetch & Status', 'View Log', and 'Pull Latest'.

Figure 5: 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

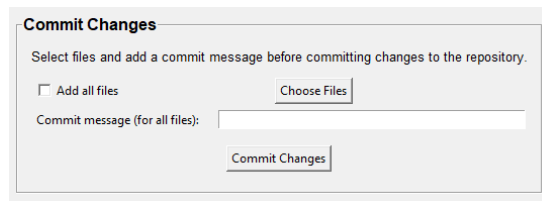


Figure 6: Add and commit file/files into the remote repository.

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).

----- We will add more info soon -----



# Appendix:

## 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.