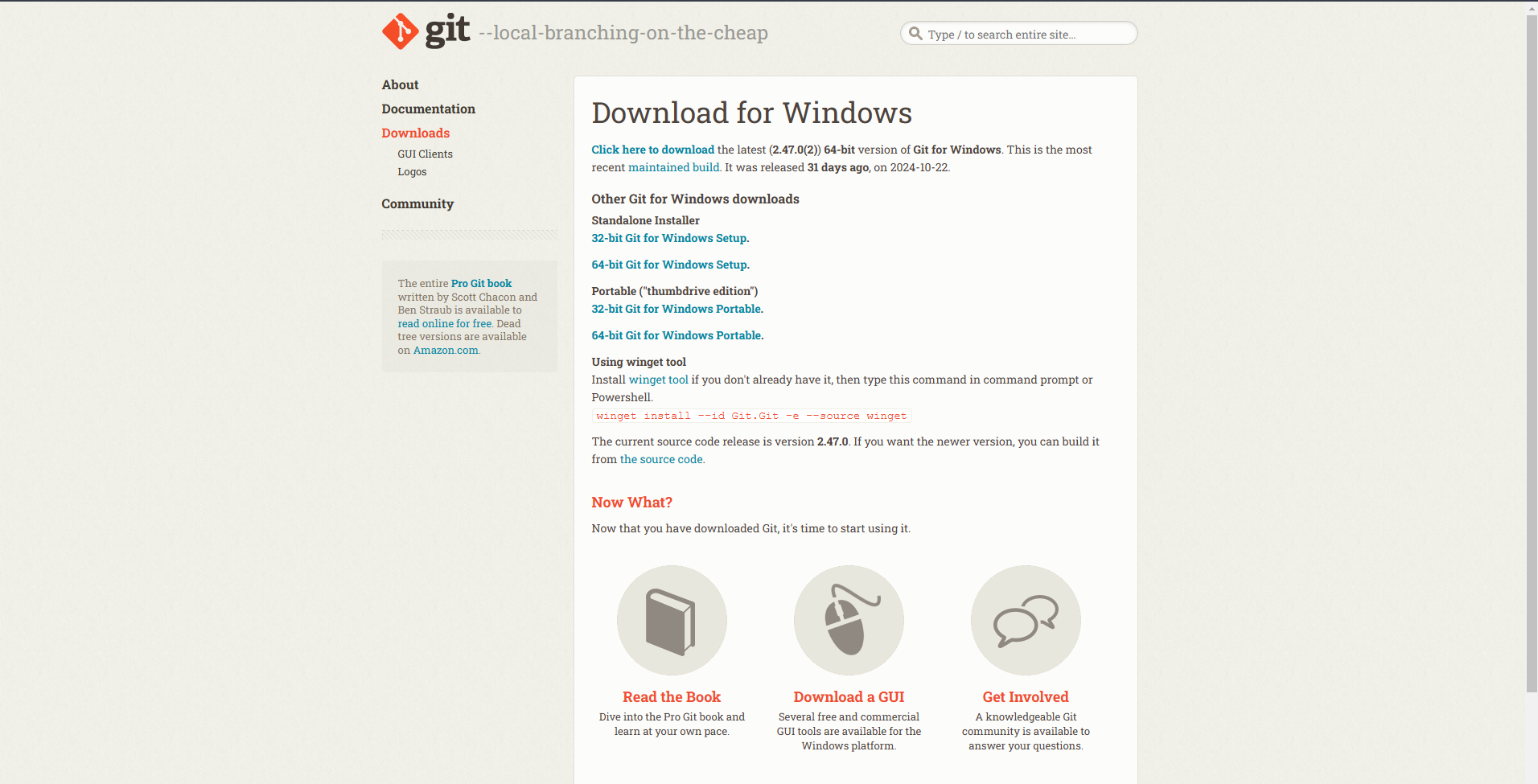
Version Control and Git Basics

# **Version Control and Git Basics**

# **Explaining the installation steps**

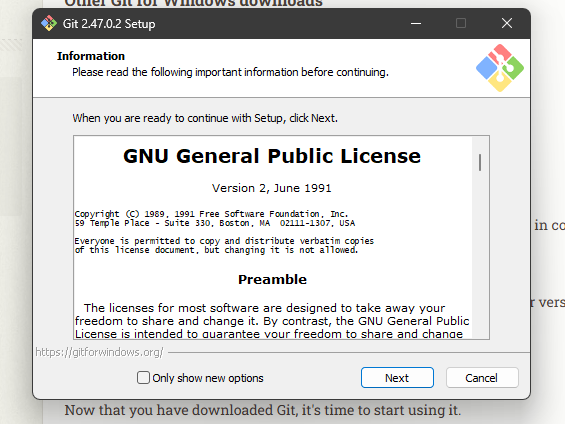


Site link : <https://git-scm.com/downloads/win>

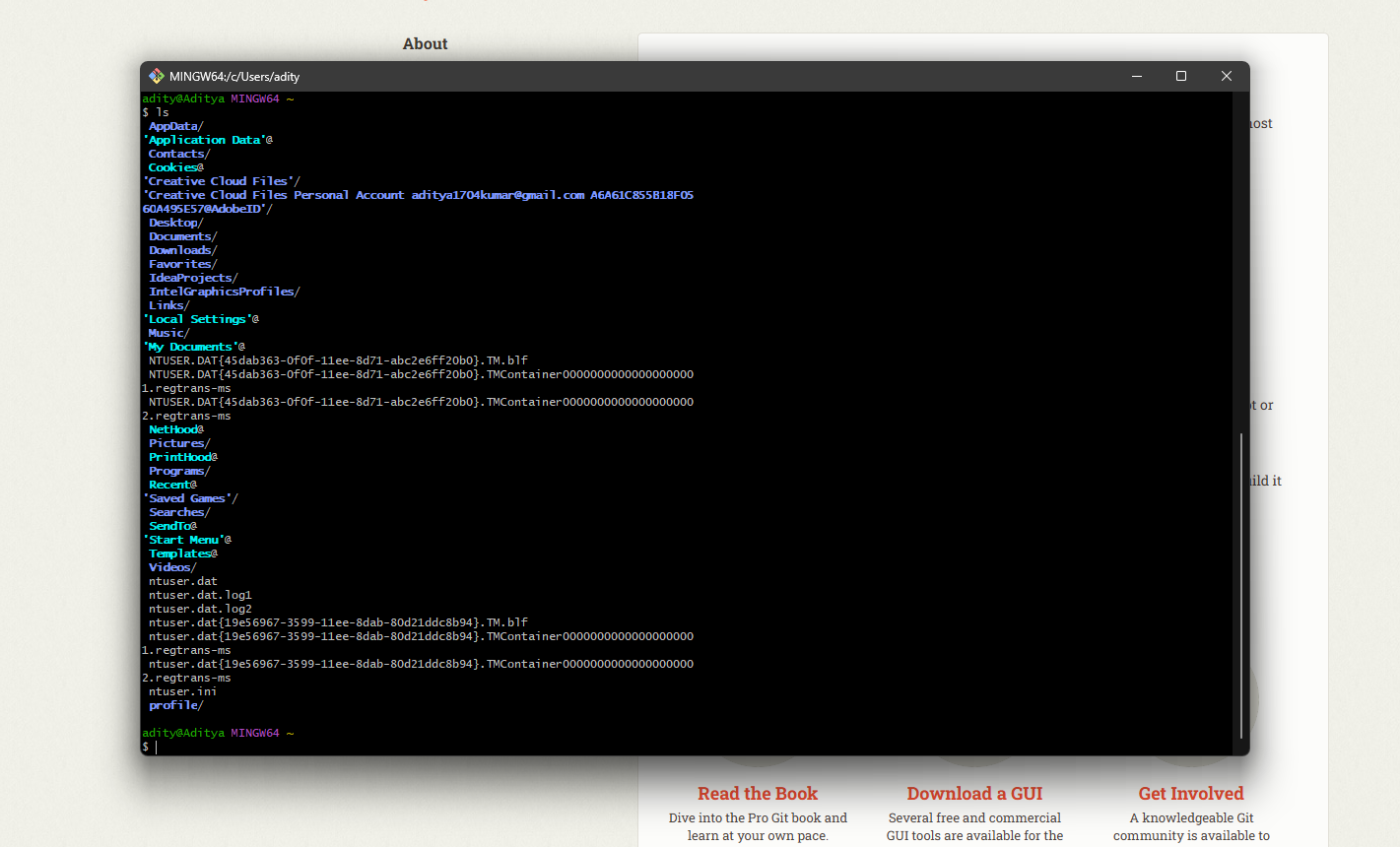
* **Go and visit this site and Download git**
* **Check your os ( Windows , Ios , Linux )**
* **Standalone Installer**
* [**32-bit Git for Windows Setup**](https://github.com/git-for-windows/git/releases/download/v2.47.0.windows.2/Git-2.47.0.2-32-bit.exe)**.** [**64-bit Git for Windows Setup**](https://github.com/git-for-windows/git/releases/download/v2.47.0.windows.2/Git-2.47.0.2-64-bit.exe)**.**

#### **Using winget tool**

* **Install** [**winget tool**](https://docs.microsoft.com/en-us/windows/package-manager/winget) **if you don't already have it, then type this command in command prompt or Powershell.  
  winget install --id Git.Git -e --source winget**

****

* **Click next Button And Install**

****

* **Check version ( git -–version )**

**Git common commands:**

* **Git add - I created a new file in the repository folder**
* **Git init - I navigated to a project folder**
* **git commit - After staging the file, I committed the changes using**

**—-----------------------------------------------------------**

# **Written by chat gpt 📥 in next line** —----------------------------------------------------------------

### **Key Features of Git:**

1. **Version Control**:
   * Git keeps track of changes made to files over time. You can easily go back to previous versions of your files if needed.
2. **Distributed**:
   * Unlike centralised version control systems, Git is distributed. This means every user has a full copy of the repository (including its history) on their local machine. This allows for work to be done offline and ensures redundancy.
3. **Branching and Merging**:
   * Git allows you to create branches (separate lines of development) to work on new features or bug fixes without affecting the main project. Once you’re done with the work, you can merge the branch back into the main project.
4. **Speed and Efficiency**:
   * Git is designed to be fast. Most operations (like commits, branching, merging) are performed locally, which means they don’t rely on a central server.
5. **Collaboration**:
   * Git is ideal for collaboration, as it allows multiple developers to work on the same project simultaneously. Changes made by different people can be merged together in a controlled way.

### **Key Git Concepts:**

1. **Repository**:
   * A Git repository (or repo) is a place where all the files and their history are stored. It can either be local (on your computer) or remote (on a platform like GitHub, GitLab, etc.).
2. **Commit**:
   * A commit is like a snapshot of your project at a specific point in time. Each commit has a unique ID and includes changes made to the files, along with a commit message describing those changes.
3. **Branch**:
   * A branch is a parallel version of your repository. You can work on a branch without affecting the main (or master) branch. After making your changes, you can merge them back into the main branch.
4. **Clone**:
   * Cloning a repository means creating a local copy of a remote Git repository. After cloning, you can make changes to the local copy and sync those changes with the remote repository.
5. **Push and Pull**:
   * **Push**: When you push, you upload your local changes (commits) to a remote repository.
   * **Pull**: When you pull, you download changes from the remote repository to your local copy.
6. **Staging Area**:
   * The staging area (or index) is where changes are prepared before committing them. You add changes to the staging area using the git add command.
7. **Merge**:
   * Merging is the process of integrating changes from different branches into one branch. Git will automatically try to merge changes, but in case of conflicting changes, you may need to resolve them manually.

### **Basic Git Commands:**

1. **git init**: Initializes a new Git repository.
2. **git clone <repository>**: Creates a local copy of a remote repository.
3. **git add <file>**: Adds changes to the staging area.
4. **git commit -m "message"**: Records changes in the repository with a commit message.
5. **git status**: Displays the current state of the repository (e.g., changes made, staged, etc.).
6. **git push**: Uploads local commits to a remote repository.
7. **git pull**: Downloads updates from a remote repository.
8. **git log**: Shows the commit history.

### **Why Use Git?**

* **Tracking Changes**: Git allows you to track every change in your code, providing a full history of a project.
* **Collaboration**: Multiple developers can work on different parts of a project simultaneously, using Git to merge their work.
* **Backup**: Since Git is distributed, you have a full copy of your project on your machine, minimizing the risk of losing your work.

In summary, Git is an essential tool for modern software development, helping developers manage code changes, collaborate with others, and keep a record of their project’s history.