CHANDAN MUKHERJEE

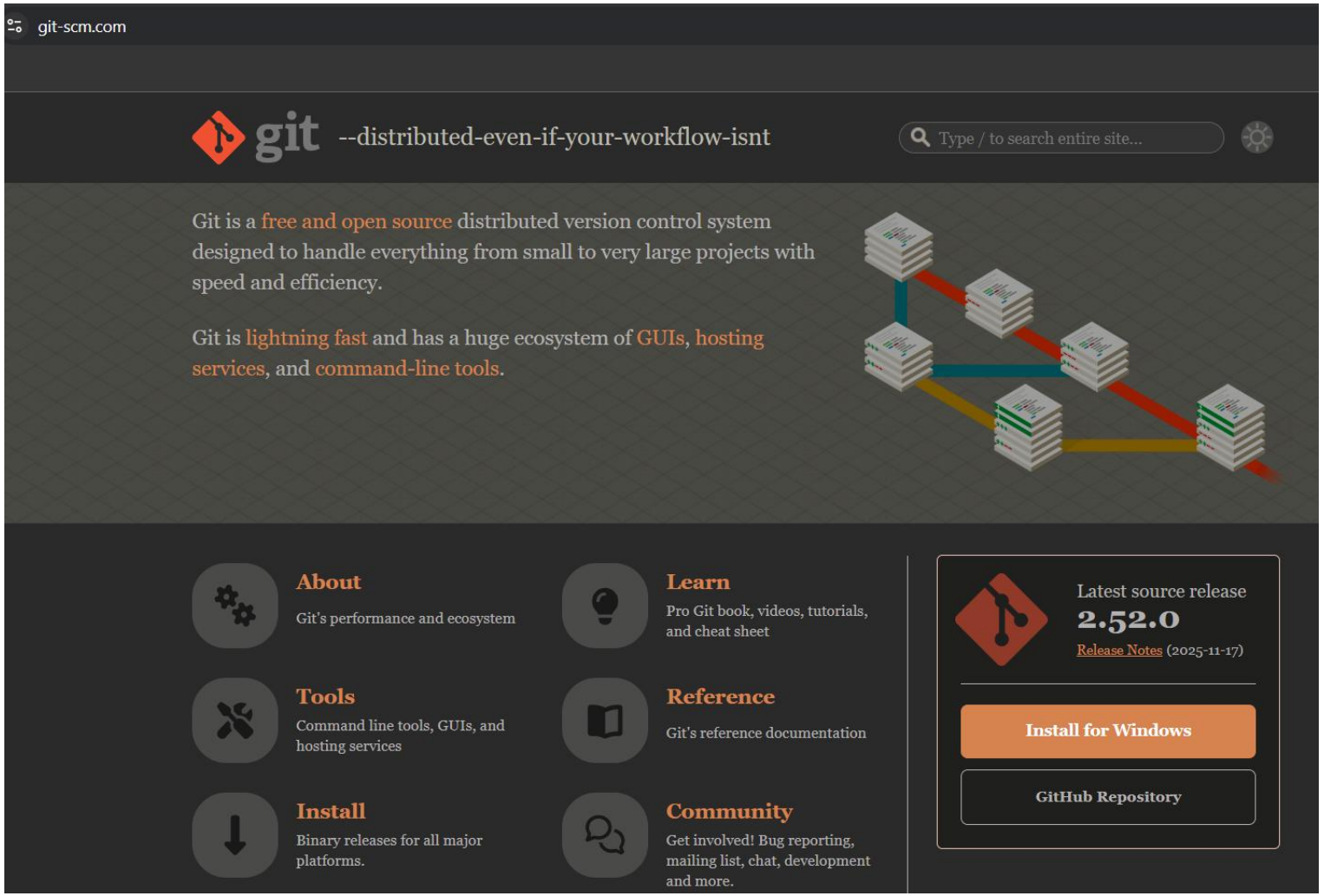
MTech (IT), BE (Computer Science), Diploma (Electrical)
AWS Certified Cloud Solution Architect Associate
AWS Certified Cloud Practitioner
Microsoft Certified Azure Fundamentals
SCJP (Java), Oracle (SQL) GLOBAL CERTIFIED
Microsoft Certified Innovative Educator
IBM Certification on Cloud, Docker, Kubernetes, Serverless, Python

GIT – GITHUB

GIT

Git is a free and open source distributed **Version Control System (VCS)** designed to handle everything from small to very large projects with speed and efficiency.

Git is a DevOps tool used for source code management



https://git-scm.com/

Git runs on your **computer** and works even without the internet.

Version Control

What is Version Control?

As a process, version control refers to tracking and managing changes to digital assets over time.

Version Control System (VCS)?

A Version Control System (VCS), also known as Source Code Management (SCM) or Revision Control System(RCM), is a software tool that helps manage changes to source code over time.

It provides a systematic way to track, organize, and coordinate changes made by multiple contributors to a codebase.

The primary goals of a VCS are to facilitate collaboration, enable versioning, and ensure the integrity and traceability of software development projects.

VCS is essential for any project that involves collaboration or multiple versions of files.

It's widely used in software development, writing, design, and many other fields.

Without Git vs With Git

Without Git	With Git
Manual backups	Automatic version history
Risky changes	Safe experimentation
Hard to collaborate	Built for teamwork

- project - v1.docx
- project - v2.docx
- project - v3.docx
- project - v3_23.12.23.docx
- project - v3_24.12.23.docx
- project - v3_27.12.23 -Review1.docx
- project - v3_27.12.23 -Review2.docx
- project - v3_27.12.23 -Review3.docx
- project - v3_27.12.23 -ReviewFinal.docx
- project - v3_27.12.23 -ReviewFinal2.docx
- project - v3_27.12.23 -ReviewFinal3.docx

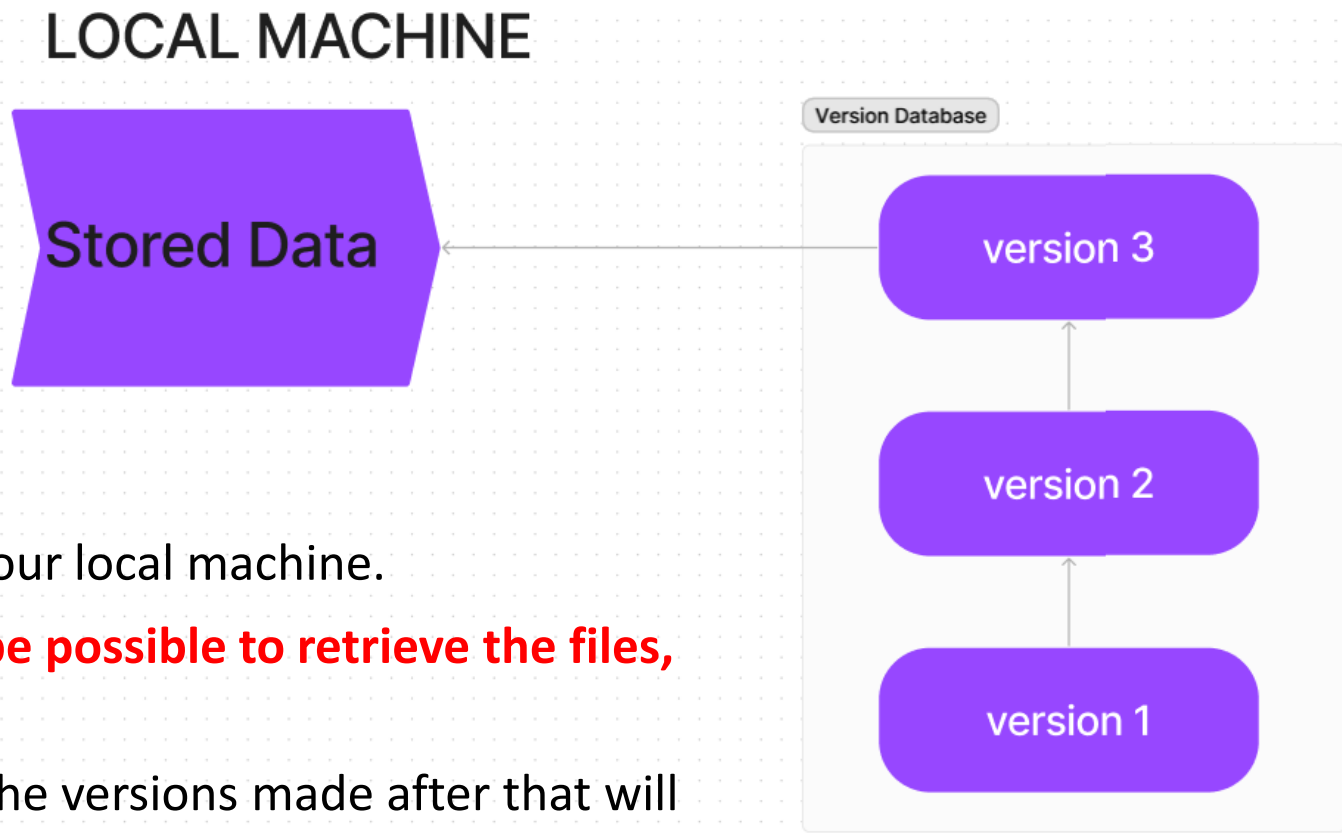
Types of VCS

1.Local Version Control System

2.Centralized Version Control System

3.Distributed Version Control System

Local Version Control System

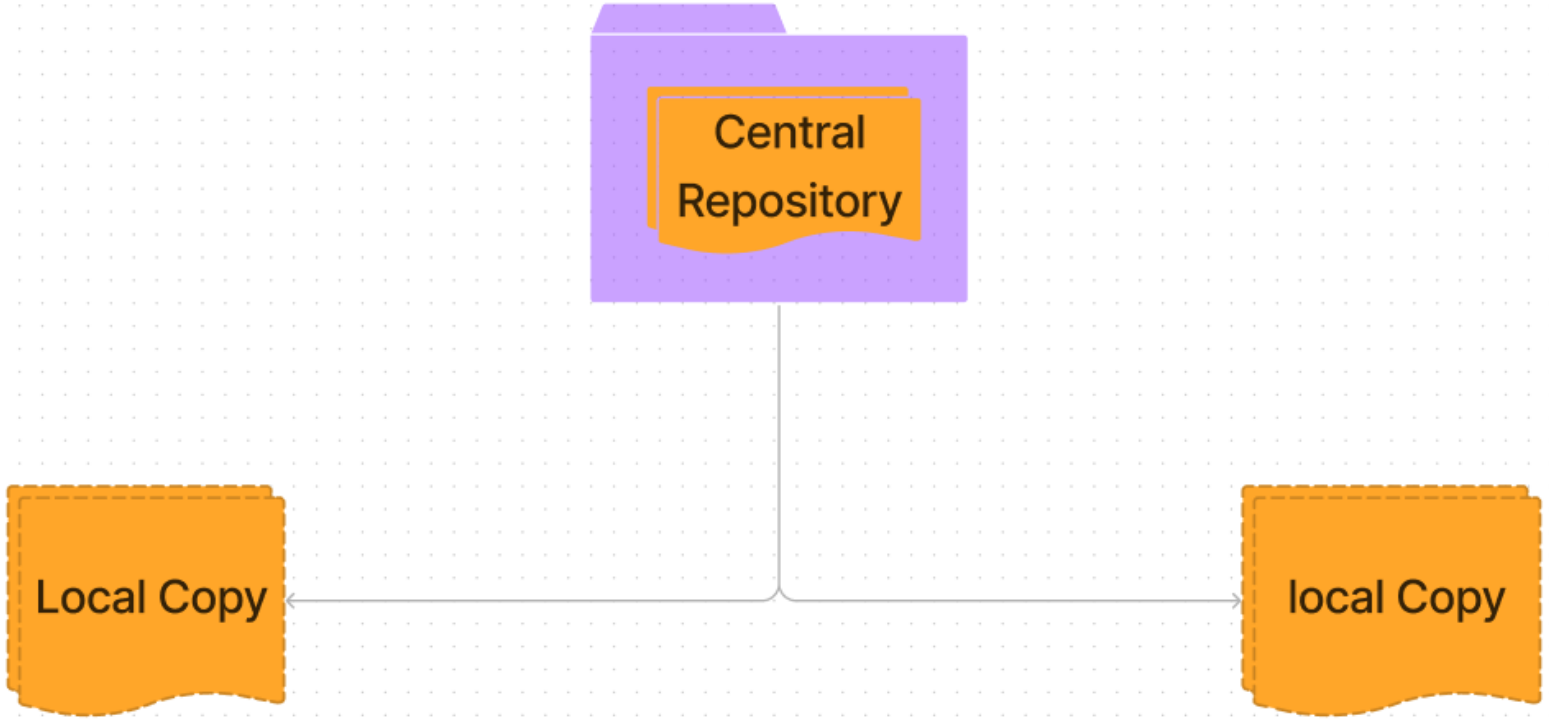


Local Version Control System is located in your local machine.

If the local machine crashes, it would not be possible to retrieve the files, and all the information will be lost.

If anything happens to a single version, all the versions made after that will be lost.

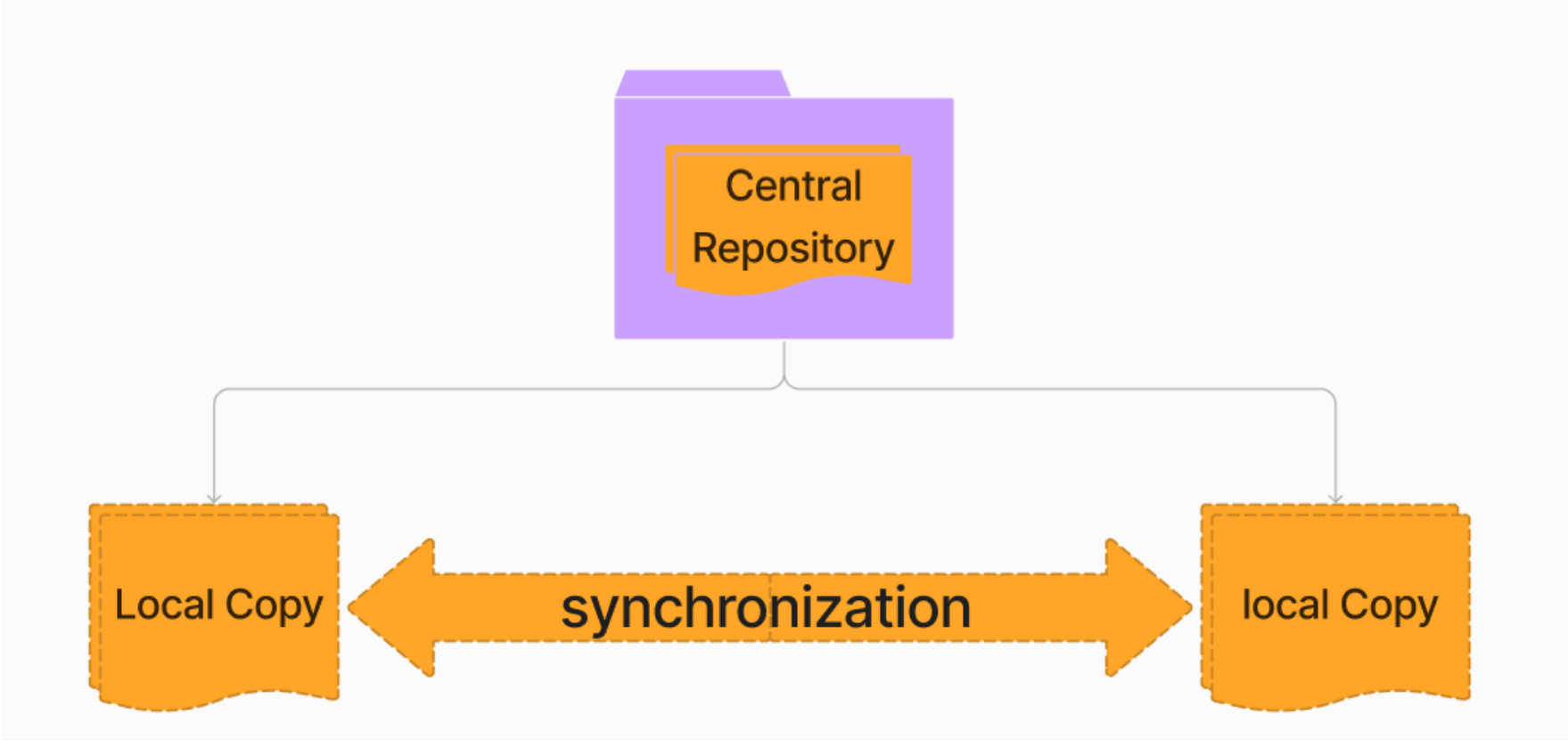
Centralized Version Control System



In the Centralized Version Control Systems, there will be a single central server that contains all the files related to the project, and many collaborators checkout files from this single server (you will only have a working copy).

The problem with the Centralized Version Control Systems is if the central server crashes, almost everything related to the project will be lost.

Distributed Version Control System



In a **distributed version control system, there will be one or more servers and many collaborators similar to the centralized system.**

But **the difference is, not only do they check out the latest version, but each collaborator will have an exact copy (mirroring) of the main repository(including its entire history) on their local machines.**

Each user has their own repository and a working copy.

This is very useful because even if the server crashes, we would not lose everything as several copies are residing in several other computers.

What Is Git?

Popular source control system

Distributed system

Free and open-source

📜 Track changes in your code over time

BACK Go back to earlier versions if something breaks

🌿 Work with branches to build features safely

👥 Collaborate locally without overwriting others' work

Git is originally created by Linus Torvalds, the creator of the Linux kernel.

It is not owned by any single company.

The trademark "Git" is registered by the Software Freedom Conservancy, a non-profit organization that supports open-source project

Why Use Git?

Fast

Disconnected

Powerful yet easy

Branching

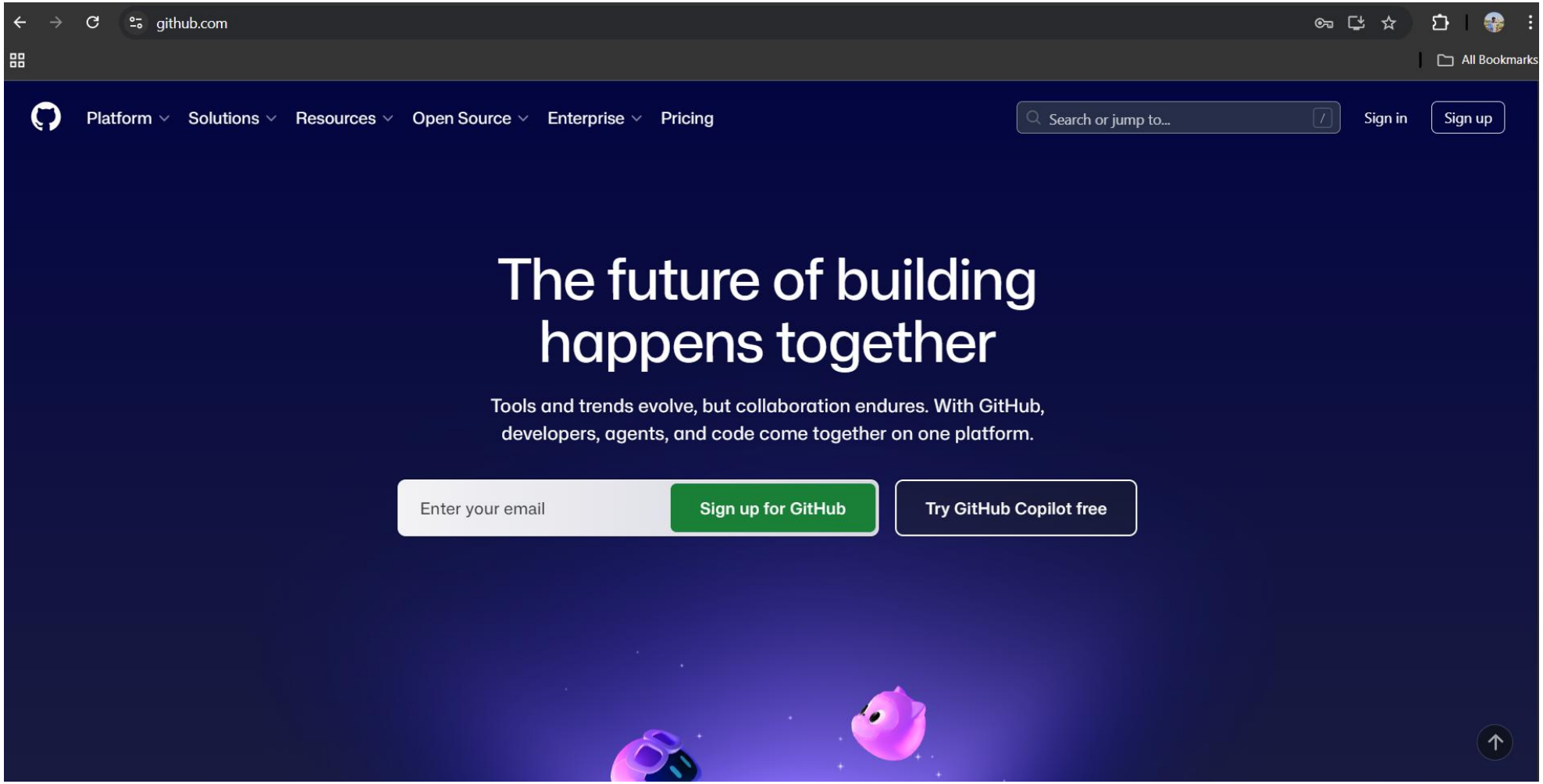
Pull requests

GITHUB

GitHub is a **cloud-based platform** that hosts Git repositories.

GitHub is owned by **Microsoft**, which acquired the company for $7.5 billion in stock in October 2018.

https://github.com/



What Is GitHub?

- ☁️ Store code online (backup + access anywhere)
- 🤝 Collaborate with teams on the same project
- 🔍 Review code using pull requests
- 🐞 Track bugs & tasks with issues
- 🚀 Showcase projects (portfolios, open source)

Hosting service based on Git

More than just source control for your code

Free and paid options

Git offers a broad variety of extra services, including issues management, working with teams, adding a wiki, and so much more.

Some source-based actions can also be done directly in the GitHub interface, such as creating a pull request or performing a merge.

GIT VS GITHUB

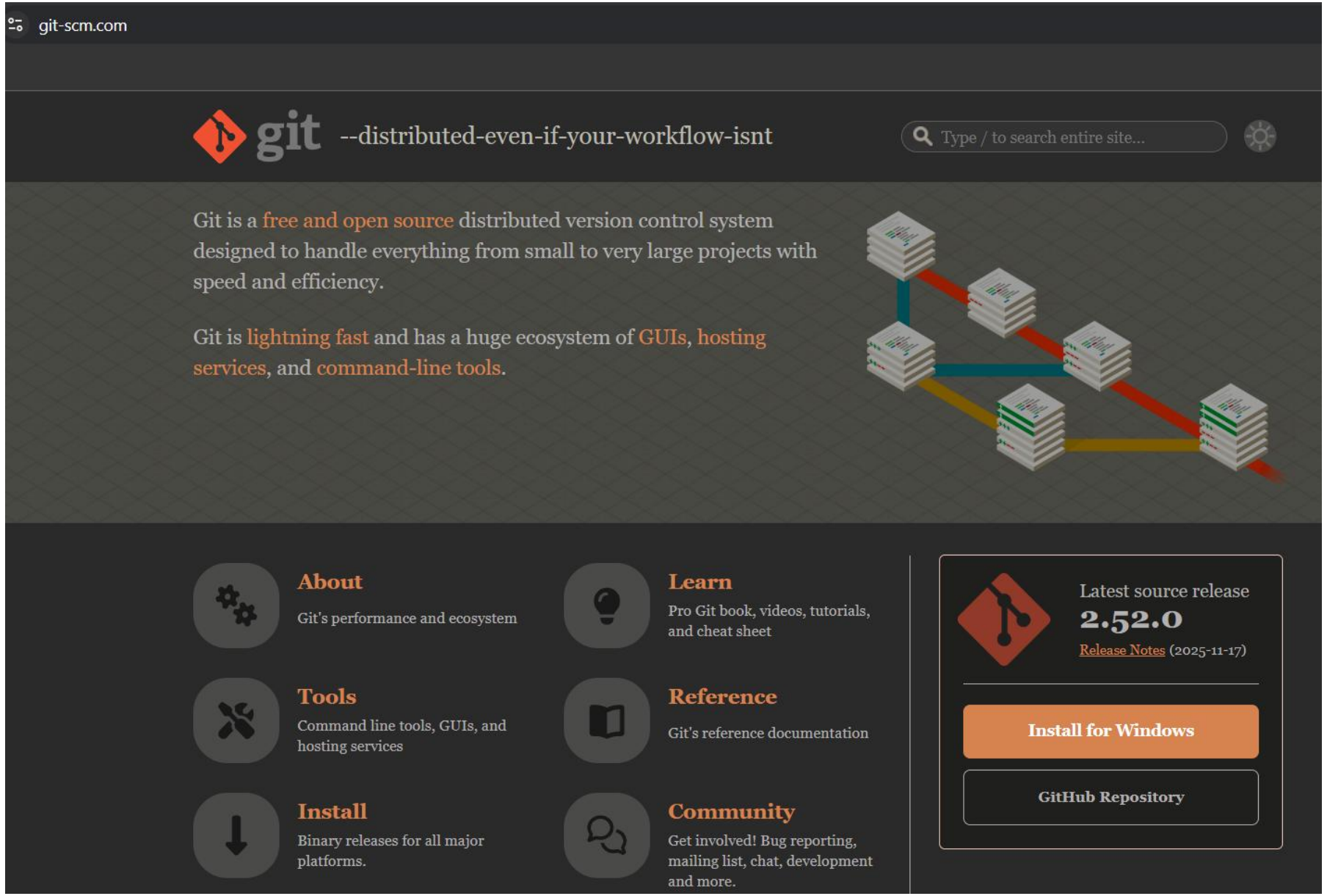
Git	GitHub
Software	Service
Version control	Git repository hosting
Maintained by Linux	Maintained by Microsoft
Open-Source	Free or paid membership
No user management	Built-in user management
Locally installed	Hosted on the web
Minimal external tool configuration	Active marketplace for tool integration
Little to no competition	High competition

GIT VS GITHUB

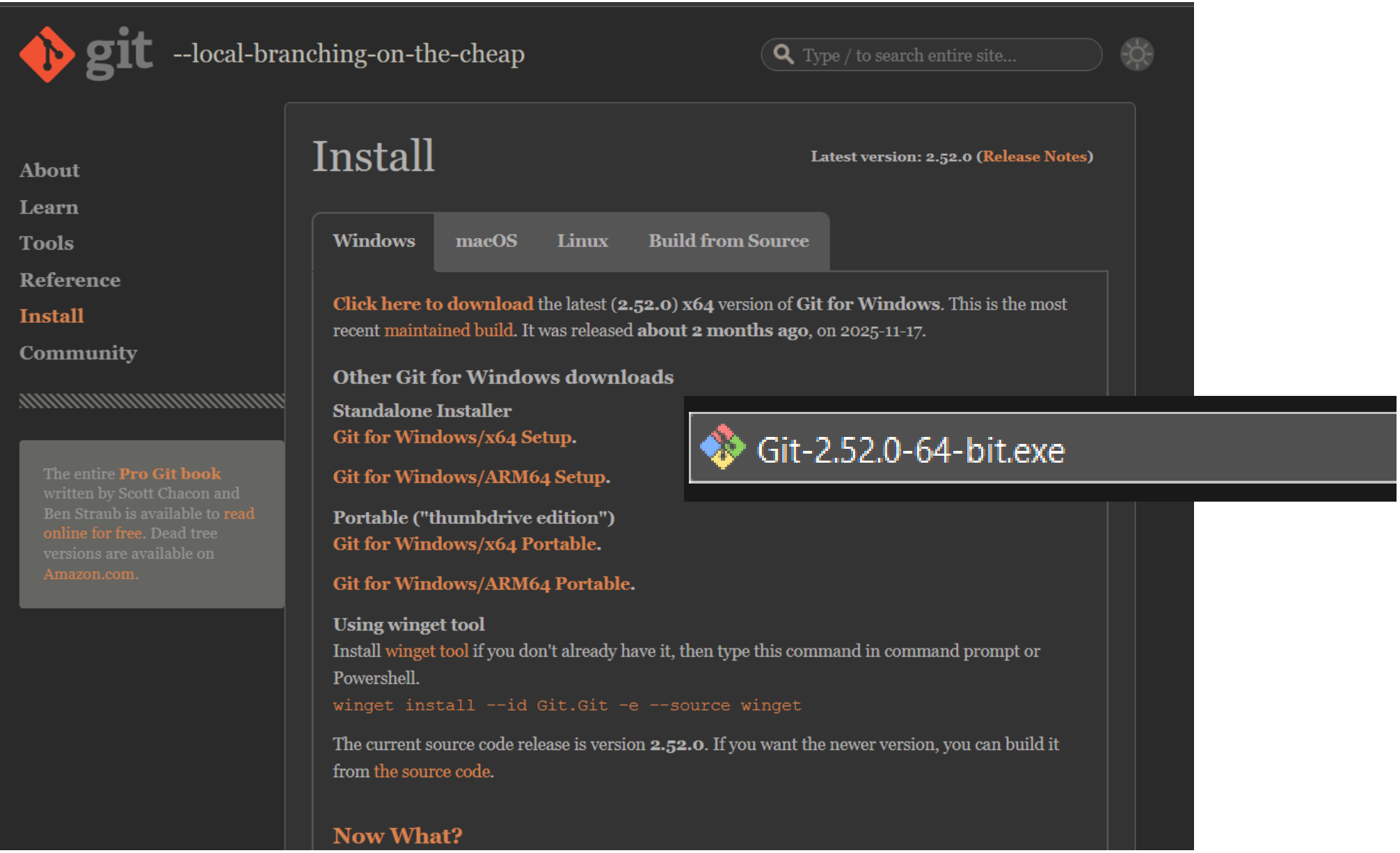
Feature	Git	GitHub
What it is	Software tool	Online platform
Works offline	✅ Yes	❌ No
Stores code	Locally	On the cloud
Tracks versions	✅ Yes	Uses Git
Collaboration	Limited	✅ Excellent
Account needed	❌ No	✅ Yes

GIT DOWNLOAD AND INSTALL

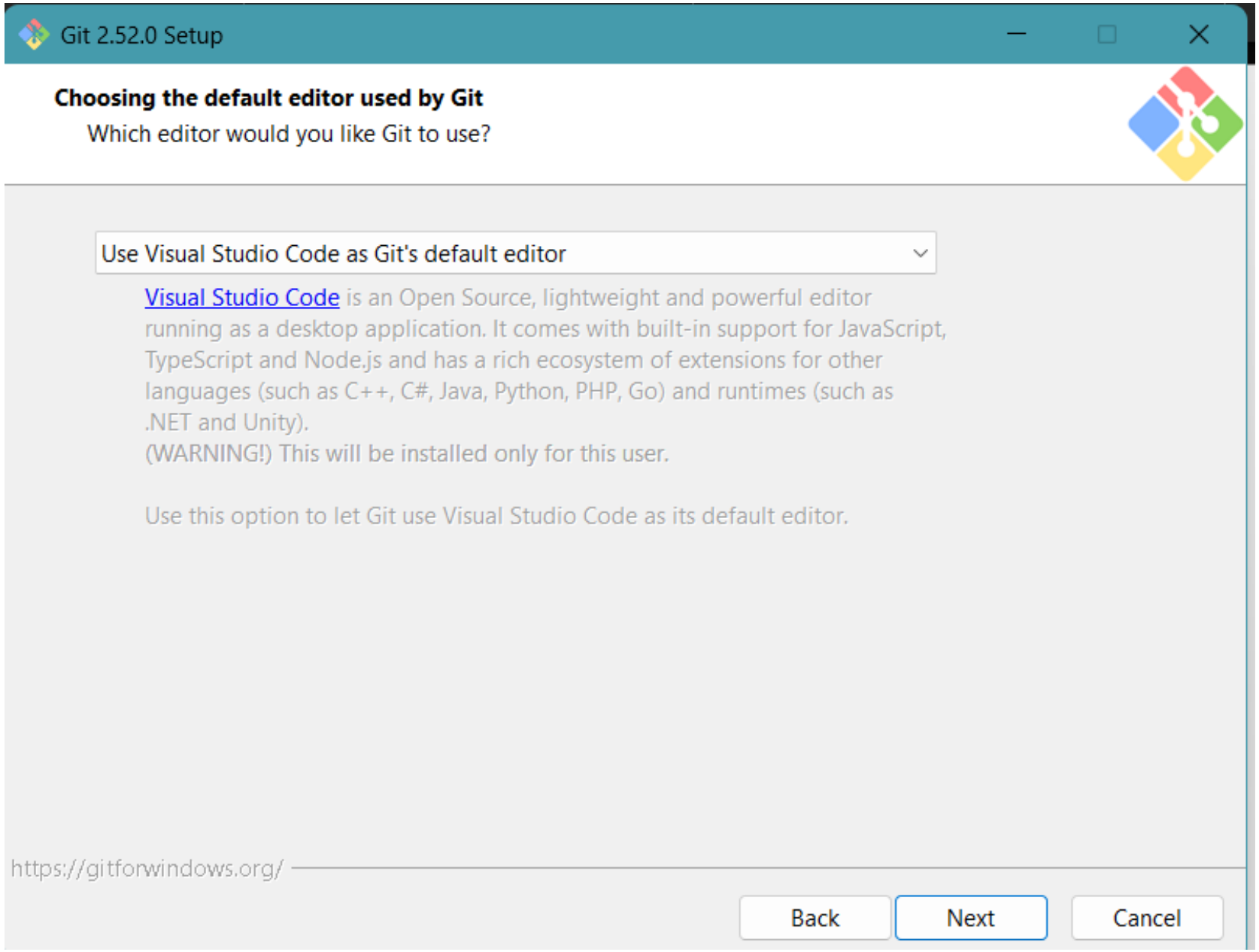
https://git-scm.com/



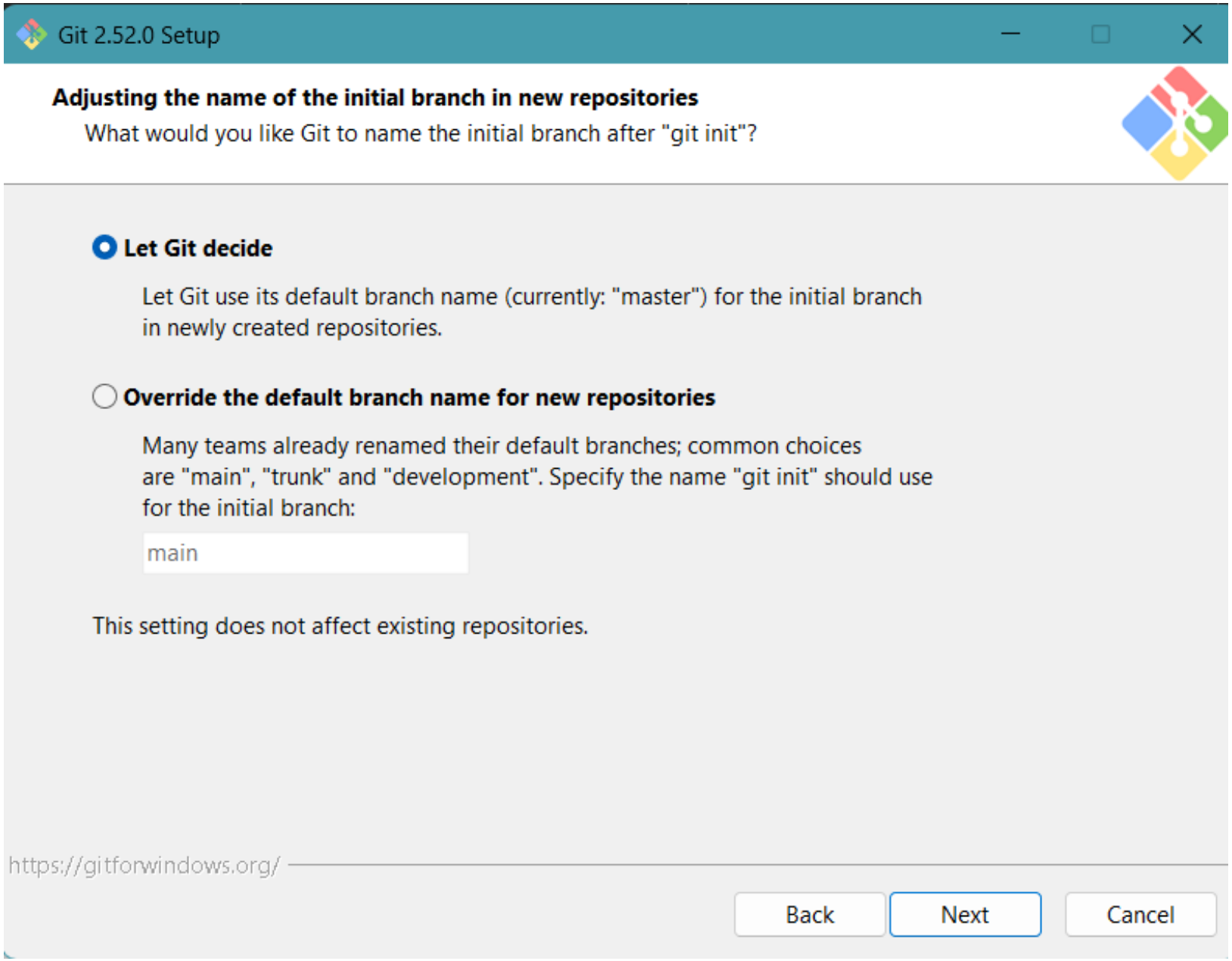
https://git-scm.com/install/windows

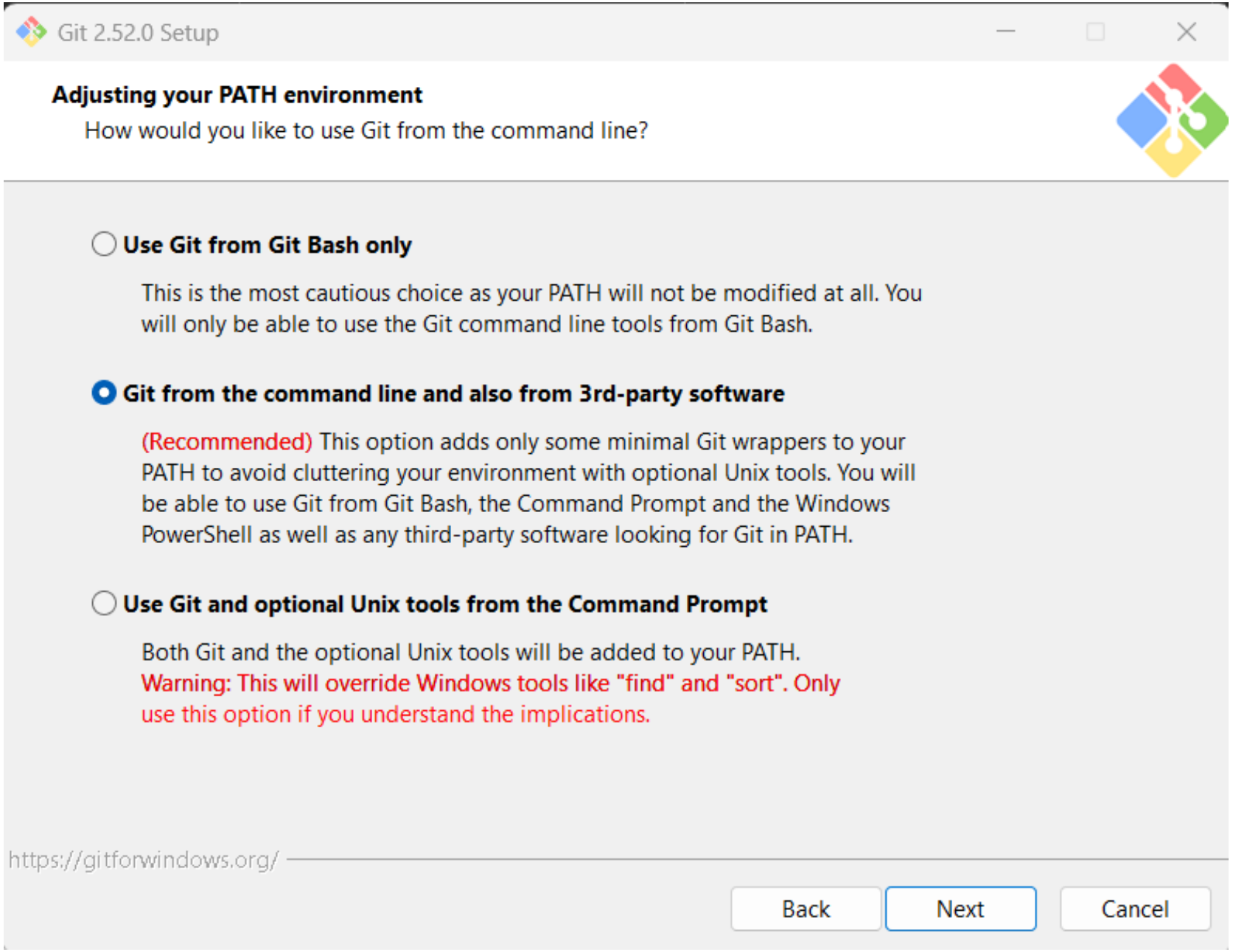


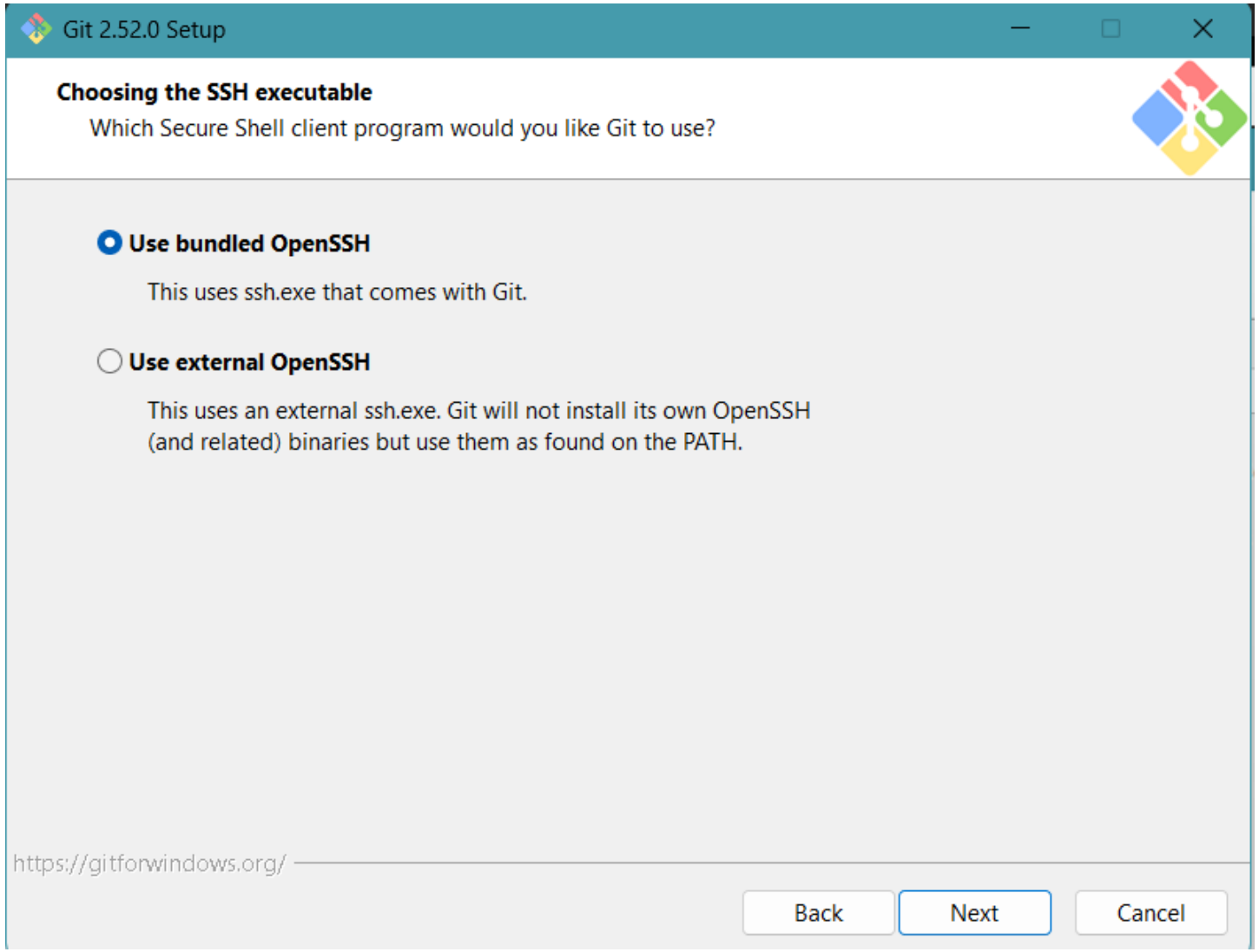
SET DEFAULT EDITOR

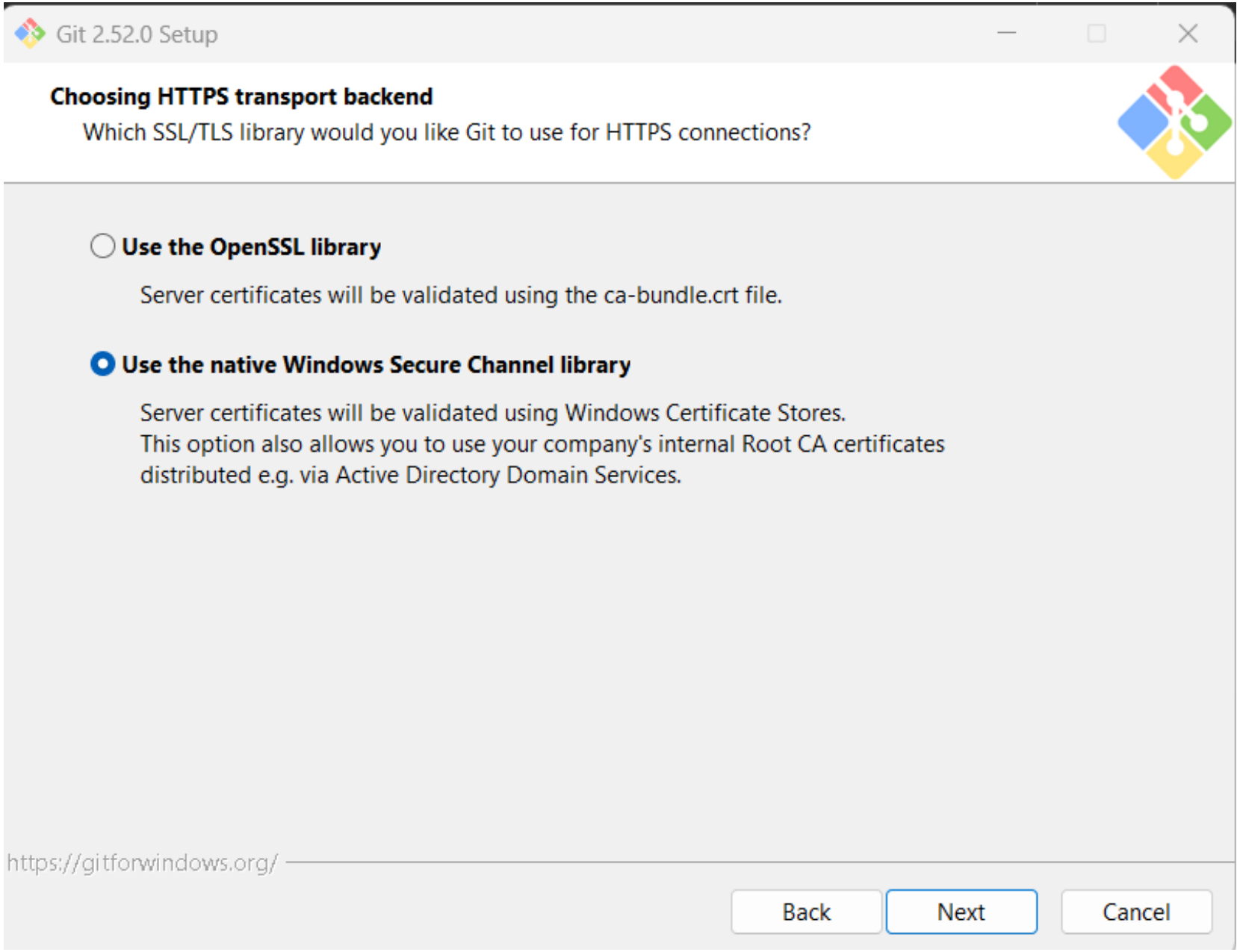


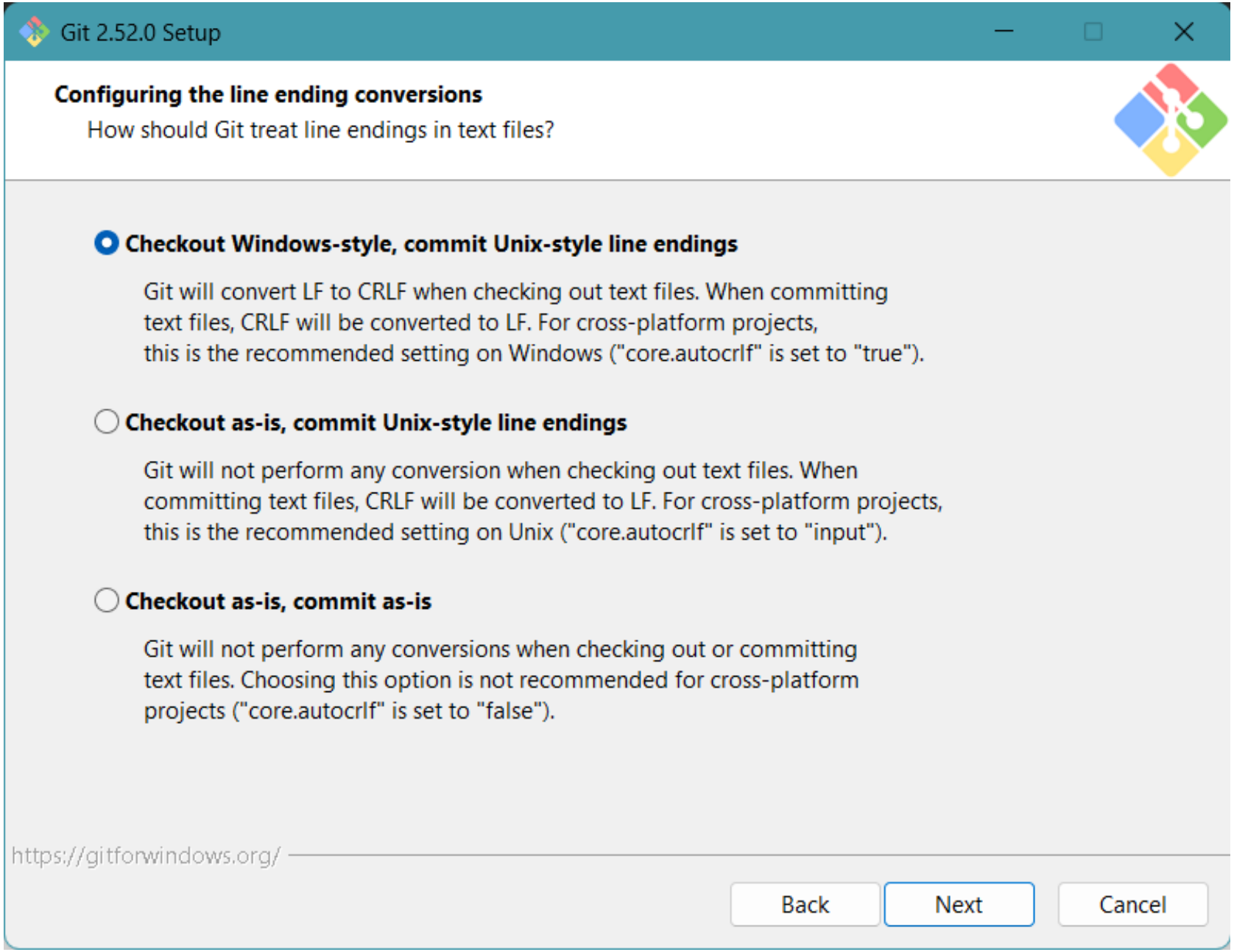
SET BRANCH NAME

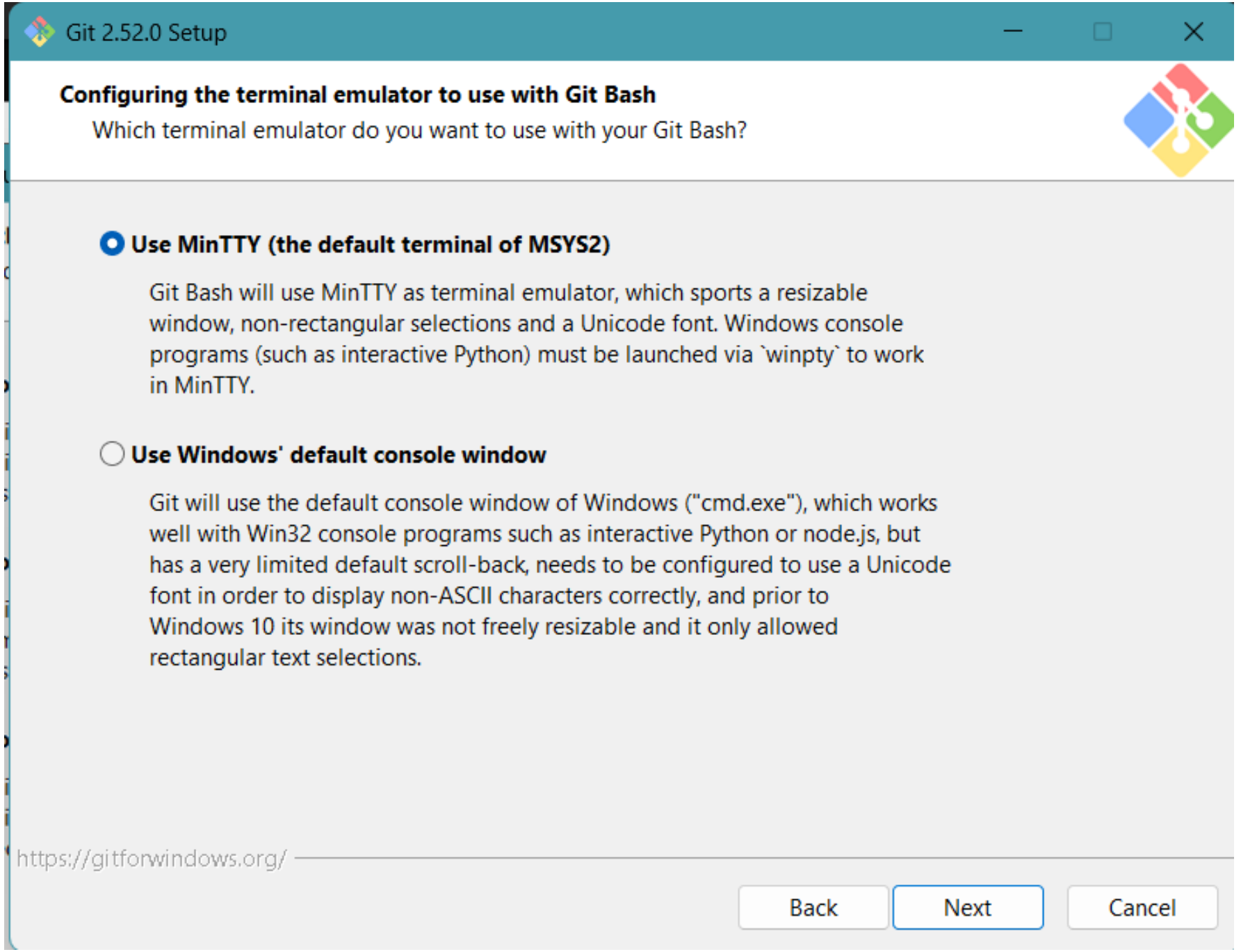


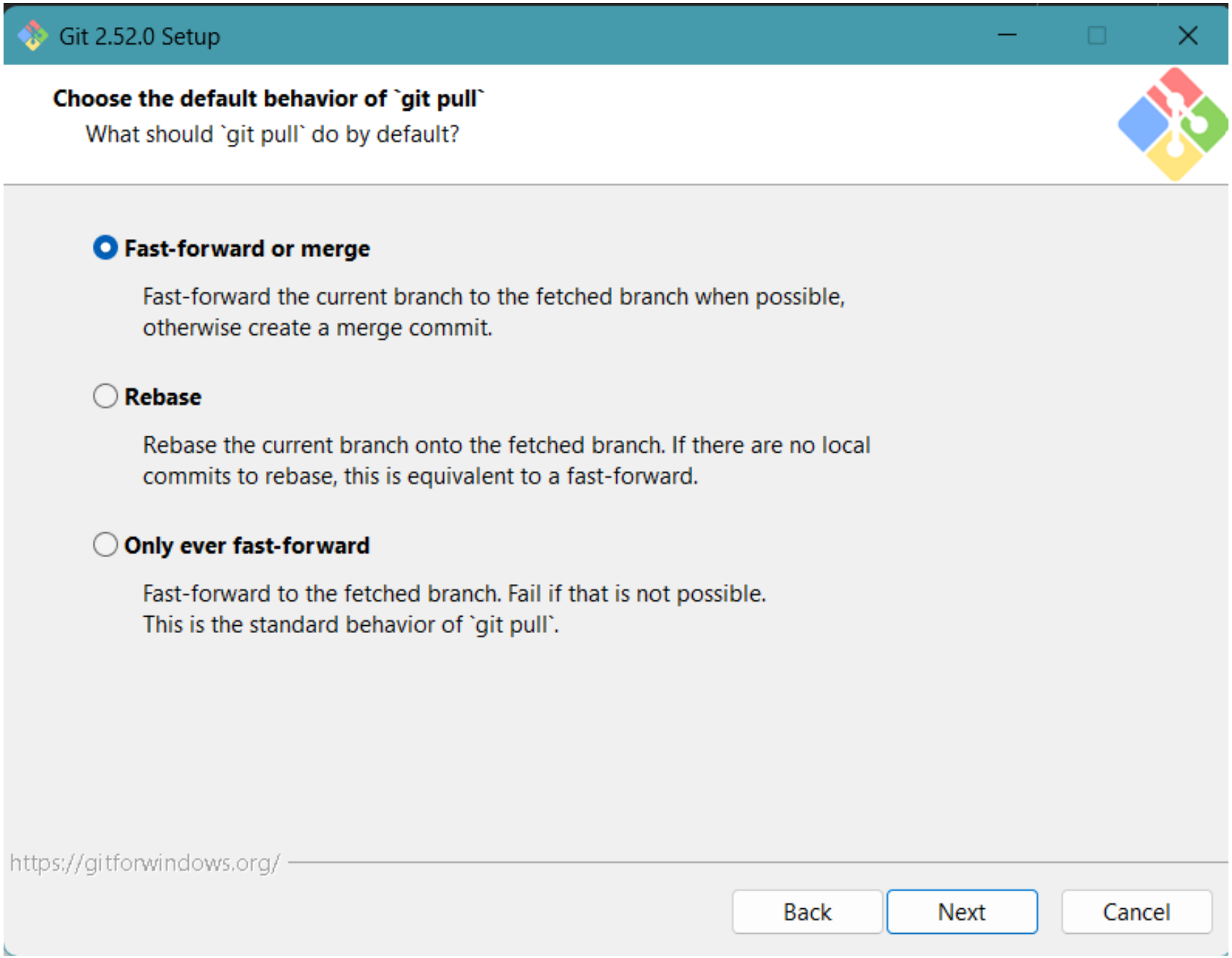

Git 2.52.0 Setup
Adjusting your PATH environment
How would you like to use Git from the command line?
Use Git from Git Bash only
This is the most cautious choice as your PATH will not be modified at all. You will only be able to use the Git command line tools from Git Bash.
Git from the command line and also from 3rd-party software
(Recommended) This option adds only some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools. You will be able to use Git from Git Bash, the Command Prompt and the Windows PowerShell as well as any third-party software looking for Git in PATH.
Use Git and optional Unix tools from the Command Prompt
Both Git and the optional Unix tools will be added to your PATH.
Warning: This will override Windows tools like "find" and "sort". Only use this option if you understand the implications.
https://gitforwindows.org/
Back
Next
Cancel

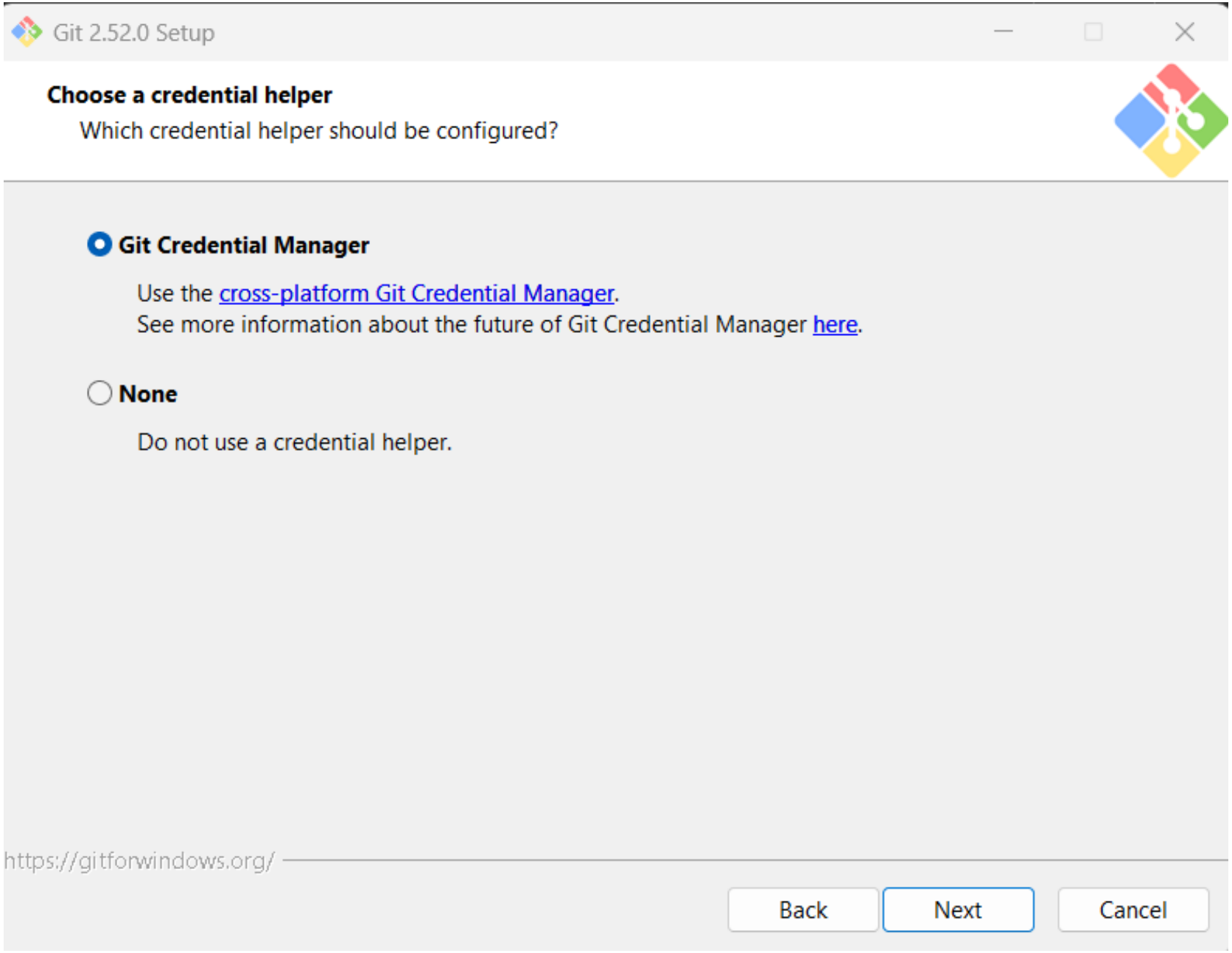

Git 2.52.0 Setup
Choosing the SSH executable
Which Secure Shell client program would you like Git to use?
Use bundled OpenSSH
This uses ssh.exe that comes with Git.
Use external OpenSSH
This uses an external ssh.exe. Git will not install its own OpenSSH
(and related) binaries but use them as found on the PATH.
https://gitforwindows.org/
Back
Next
Cancel

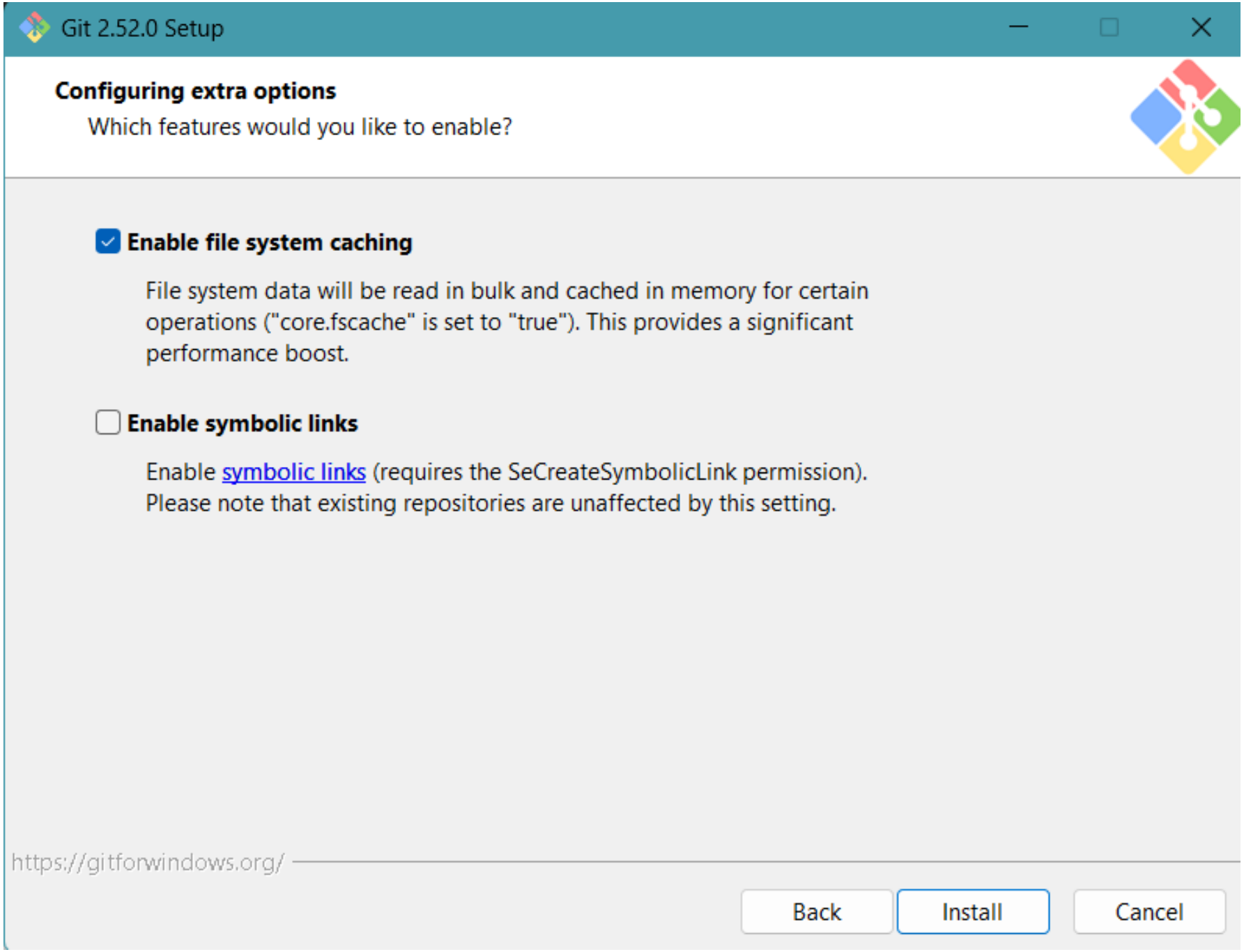

Git 2.52.0 Setup
Choosing HTTPS transport backend
Which SSL/TLS library would you like Git to use for HTTPS connections?
Use the OpenSSL library
Server certificates will be validated using the ca-bundle.crt file.
Use the native Windows Secure Channel library
Server certificates will be validated using Windows Certificate Stores.
This option also allows you to use your company's internal Root CA certificates
distributed e.g. via Active Directory Domain Services.
https://gitforwindows.org/
Back
Next
Cancel


Git 2.52.0 Setup
Configuring the line ending conversions
How should Git treat line endings in text files?
Checkout Windows-style, commit Unix-style line endings
Git will convert LF to CRLF when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Windows ("core.autocrlf" is set to "true").
Checkout as-is, commit Unix-style line endings
Git will not perform any conversion when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Unix ("core.autocrlf" is set to "input").
Checkout as-is, commit as-is
Git will not perform any conversions when checking out or committing text files. Choosing this option is not recommended for cross-platform projects ("core.autocrlf" is set to "false").
https://gitforwindows.org/
Back
Next
Cancel


Git 2.52.0 Setup
Configuring the terminal emulator to use with Git Bash
Which terminal emulator do you want to use with your Git Bash?
Use MinTTY (the default terminal of MSYS2)
Git Bash will use MinTTY as terminal emulator, which sports a resizable window, non-rectangular selections and a Unicode font. Windows console programs (such as interactive Python) must be launched via `winpty` to work in MinTTY.
Use Windows' default console window
Git will use the default console window of Windows ("cmd.exe"), which works well with Win32 console programs such as interactive Python or node.js, but has a very limited default scroll-back, needs to be configured to use a Unicode font in order to display non-ASCII characters correctly, and prior to Windows 10 its window was not freely resizable and it only allowed rectangular text selections.
https://gitforwindows.org/
Back
Next
Cancel


Git 2.52.0 Setup
Choose the default behavior of `git pull`
What should `git pull` do by default?
Fast-forward or merge
Fast-forward the current branch to the fetched branch when possible, otherwise create a merge commit.
Rebase
Rebase the current branch onto the fetched branch. If there are no local commits to rebase, this is equivalent to a fast-forward.
Only ever fast-forward
Fast-forward to the fetched branch. Fail if that is not possible.
This is the standard behavior of `git pull`.
https://gitforwindows.org/
Back
Next
Cancel


Git 2.52.0 Setup
Choose a credential helper
Which credential helper should be configured?
Git Credential Manager
Use the cross-platform Git Credential Manager.
See more information about the future of Git Credential Manager here.
None
Do not use a credential helper.
https://gitforwindows.org/
Back
Next
Cancel


Git 2.52.0 Setup
Configuring extra options
Which features would you like to enable?
Enable file system caching
File system data will be read in bulk and cached in memory for certain operations ("core.fscache" is set to "true"). This provides a significant performance boost.
Enable symbolic links
Enable symbolic links (requires the SeCreateSymbolicLink permission). Please note that existing repositories are unaffected by this setting.
https://gitforwindows.org/
Back
Install
Cancel

Git 2.52.0 Setup

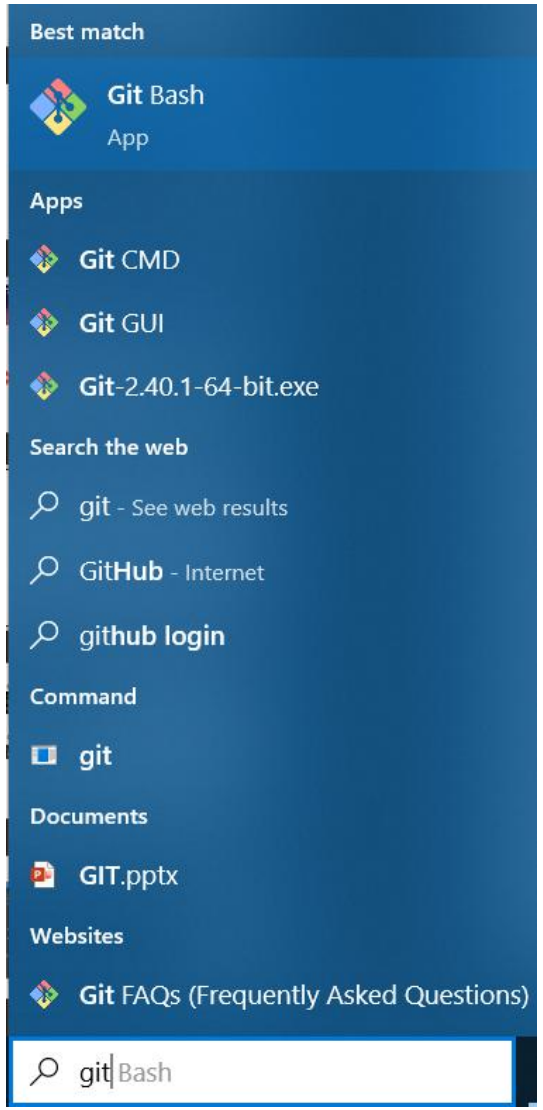
Completing the Git Setup Wizard

Setup has finished installing Git on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.

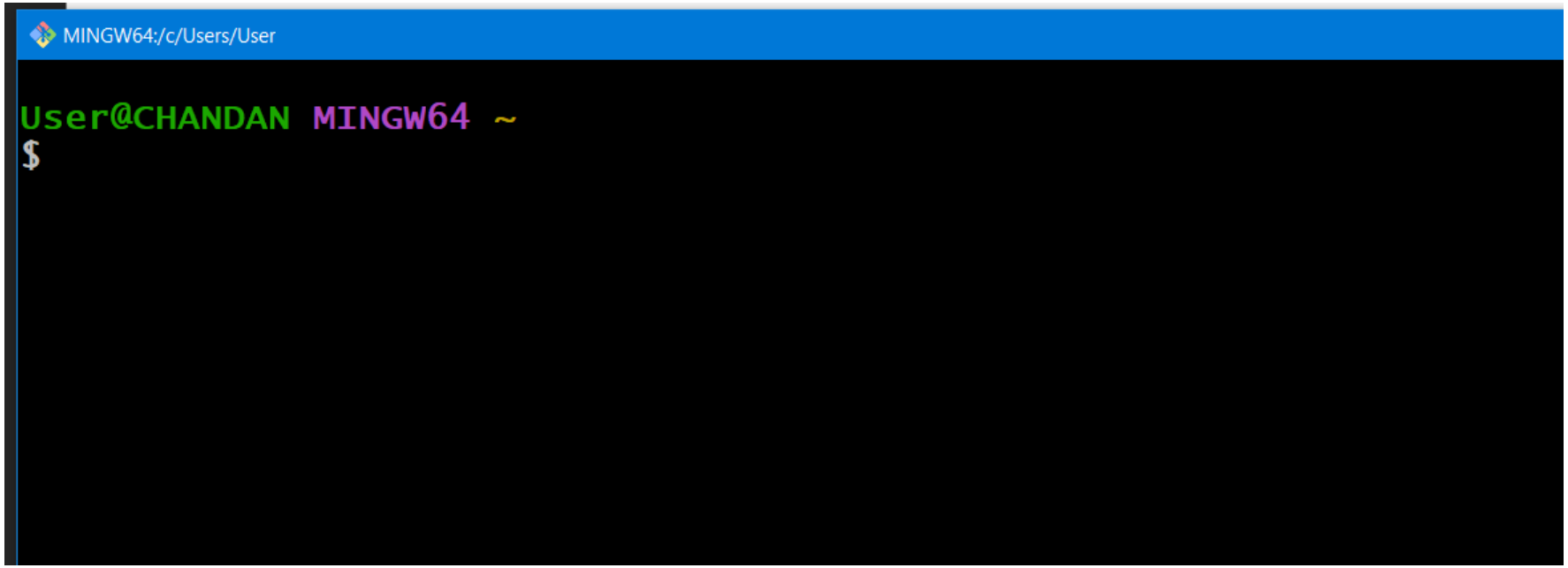
- [ ] Launch Git Bash
- [ ] View Release Notes

Finish



GIT COMMAND INTRODUCTION

USE GIT BASH



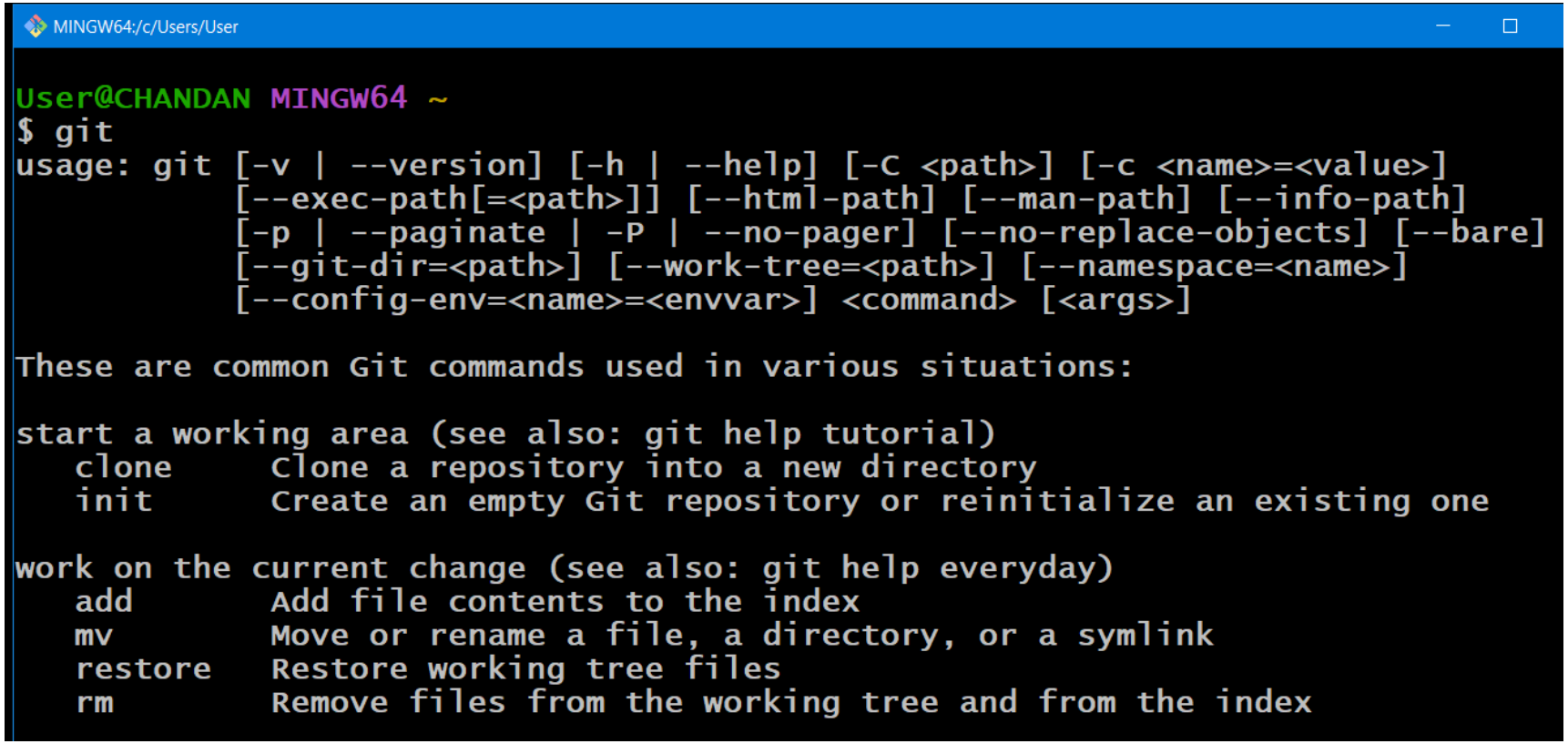
GIT VERSION CHECK

```
User@CHANDAN MINGW64 ~
$ git --version
git version 2.40.1.windows.1
```

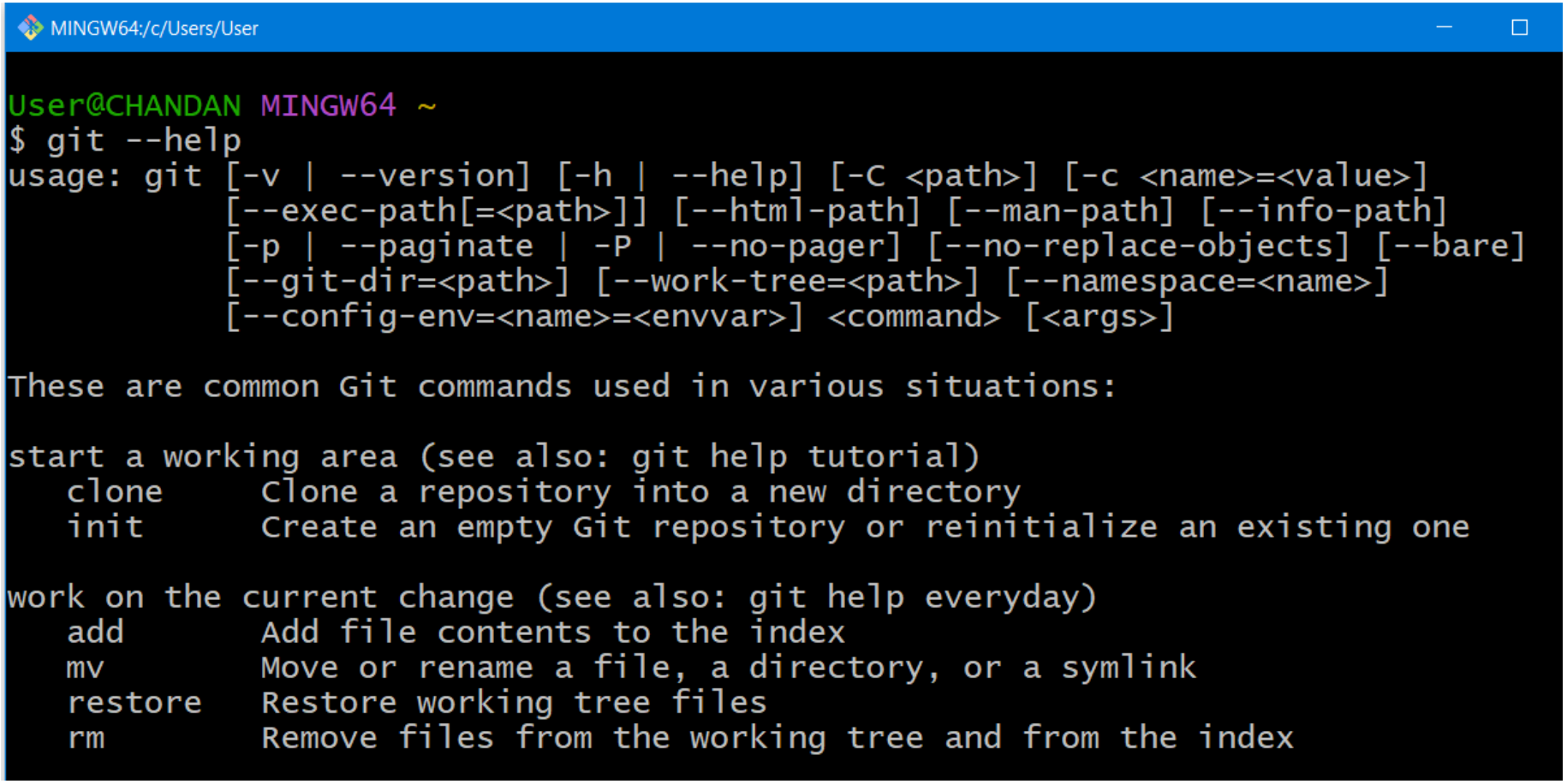
CLEAR BASH

```
User@CHANDAN MINGW64 ~
$ clear
```

GET ALL GIT COMMANDS



GIT HELP



https://www.adservio.fr/post/top-20-git-commands-with-examples#el9

The 3 States of Git

All these changes are still in local

Modified

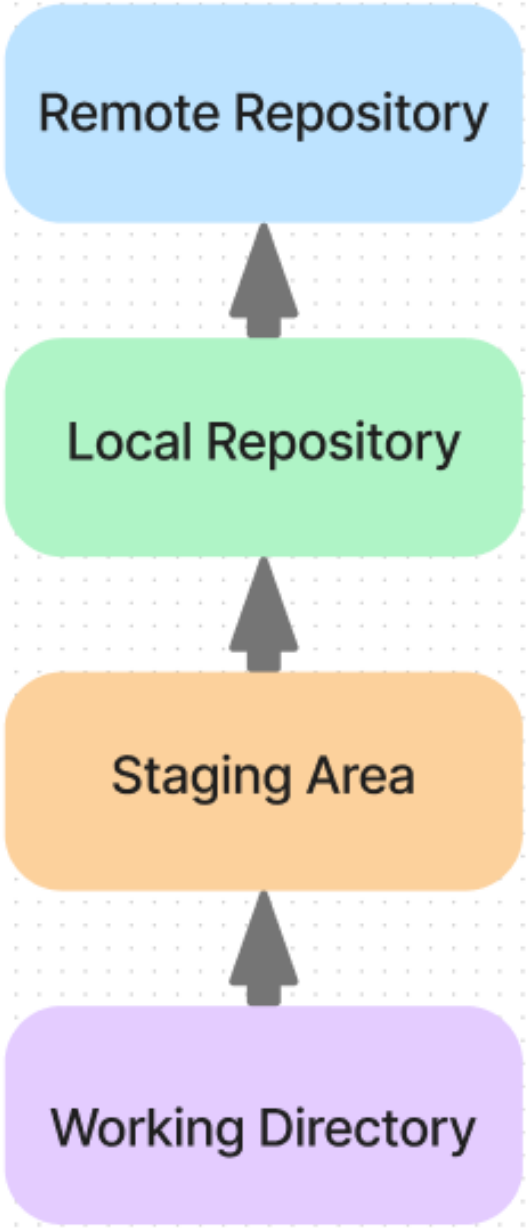
The modified state means that the file has been changed, but it hasn't been committed to the database yet.

Staged

staged means that the modified file is marked to be part of the next commit snapshot.

Committed

In the committed state, the data is basically stored in the local database.



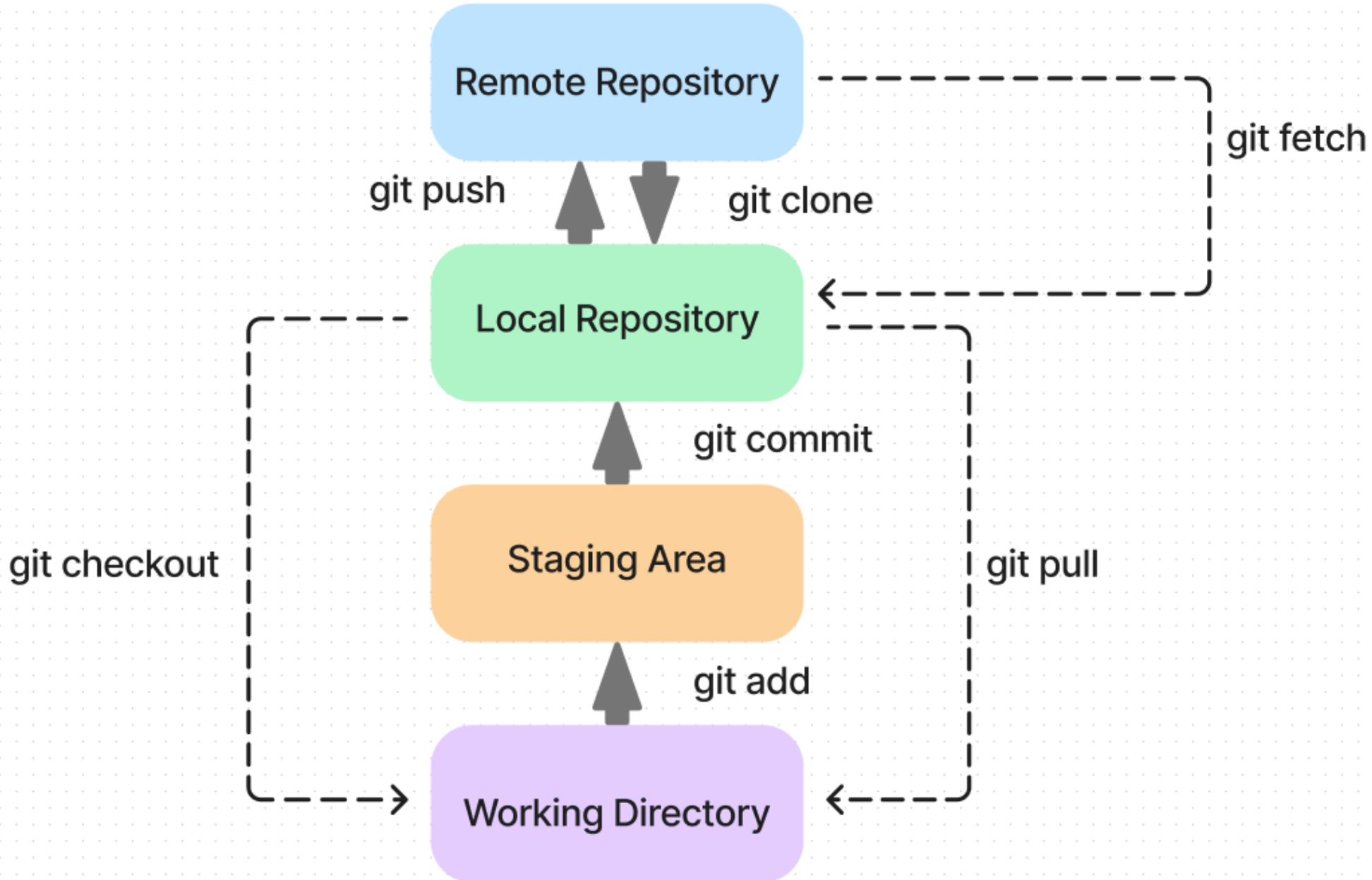
Starting at the bottom is the **working directory** where **content is created, edited, deleted, and so on**.
Any new content must exist here before it can be put into (tracked by) Git.
This serves the same purpose as the Dev area in the dev-test-prod-public model.

Next is the **staging area**.
This **serves as a holding area to accumulate and stage changes from the working directory before they are committed into the next level**—the local repository.
You can think of this process as being similar to how you might move content to the testing stage in your dev-test-prod-public model.

After the staging area comes the **local repository**.
This is the actual source repository where content that Git manages is stored.
Once content is committed to the local repository, it becomes a version in the repository and can be retrieved later.

The combination of the working directory, staging area, and local repository make up your **local environment**.
These are the parts of the Git system that exist on your local machine—actually, within a special subdirectory of the root (top-level) directory of your working directory.

The remote repository is a separate Git repository intended to collect and host content pushed to it from one or more local repositories.


Remote Repository
git fetch
git push
git clone
Local Repository
git commit
git checkout
Staging Area
git pull
git add
Working Directory