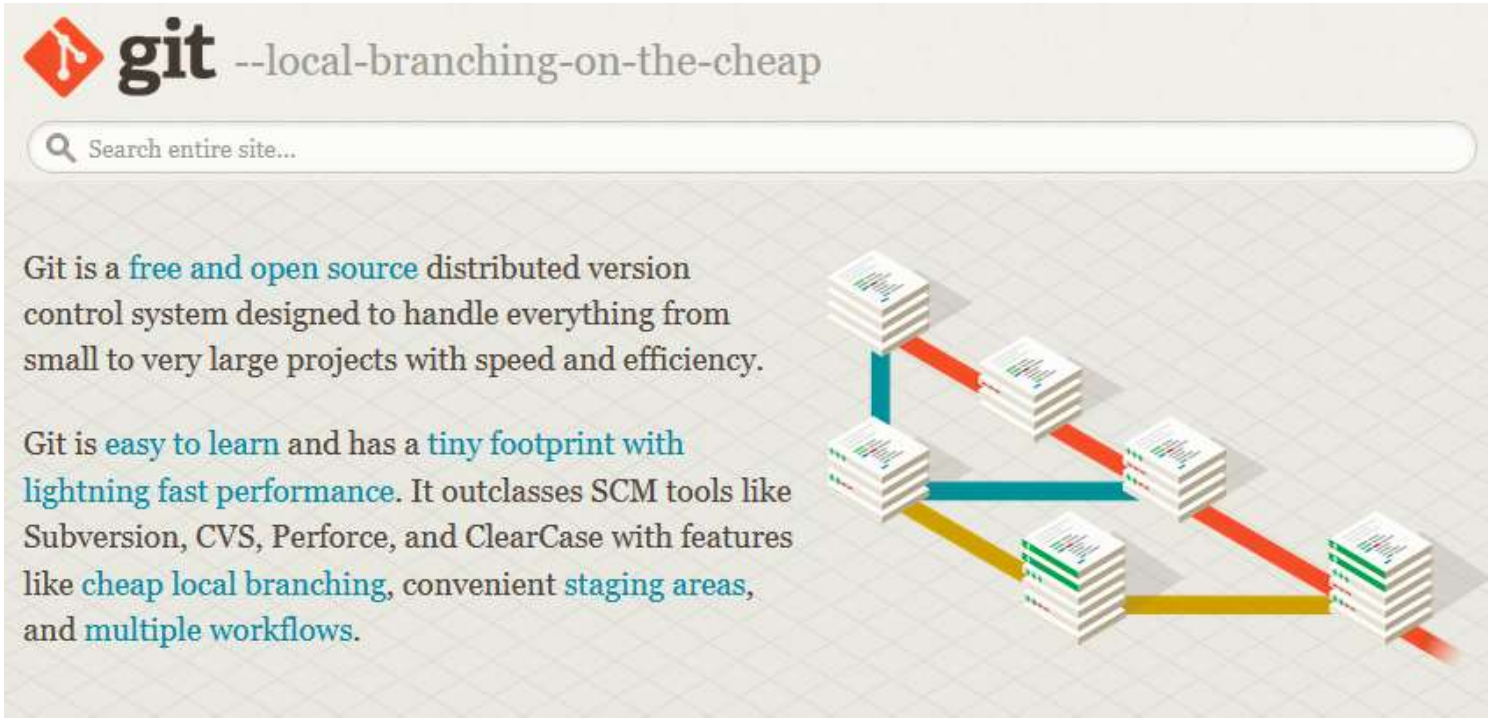
Using Git as Source Code Control System

- Know what Git is used for and who created it
- Install Git on your Windows PC
- Use Git to share code during the training
- Use git to download flutter sources from GitHub
- Understand Staging, Commit, Fetch, Pull and Push in Git
- Use the “Git Graph” extension in Visual Studio Code

What is Git



Copied from https://git-scm.com/

History of Git

10 Years of Git: An Interview with Git Creator Linus Torvalds

THE LINUX FOUNDATION | 06 APRIL 2015

Ten years ago this week, the Linux kernel community faced a daunting challenge: They could no longer use their revision control system BitKeeper and no other Source Control Management (SCMs) met their needs for a distributed system. Linus Torvalds, the creator of Linux, took the challenge into his own hands and disappeared over the weekend to emerge the following week with Git. Today Git is used for thousands of projects and has ushered in a new level of social coding among programmers.

Copied from https://www.linuxfoundation.org/blog/blog/10-years-of-git-an-interview-with-git-creator-linus-torvalds where you can find an interview with Linus Torvalds on Git.

Some questions on Git

What does Git do?

- Manage projects with **Repositories**
- **Clone** a project to work on a local copy
- Control and track changes with **Staging** and **Committing**
- **Branch** and **Merge** to allow for work on different parts and versions of a project
- **Pull** the latest version of the project to a local copy
- **Push** local updates to the main project

Why Git?

- Over 70% of developers use Git!
- Developers can work together from anywhere in the world.
- Developers can see the full history of the project.
- Developers can revert to earlier versions of a project.

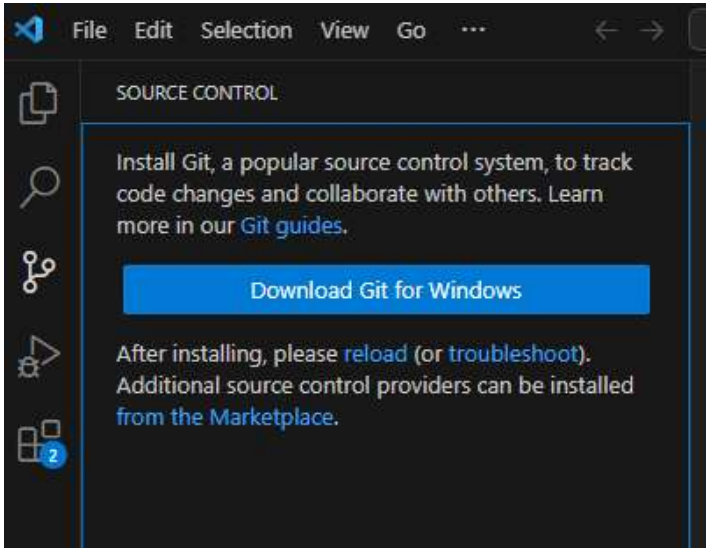
What is GitHub?

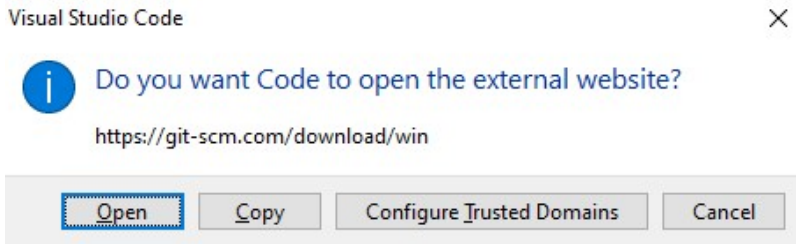
- Git is not the same as GitHub.
- GitHub makes tools that use Git.
- GitHub is the largest host of source code in the world, and has been owned by Microsoft since 2018.

Copied from https://www.w3schools.com/git/git_intro.asp?remote=github

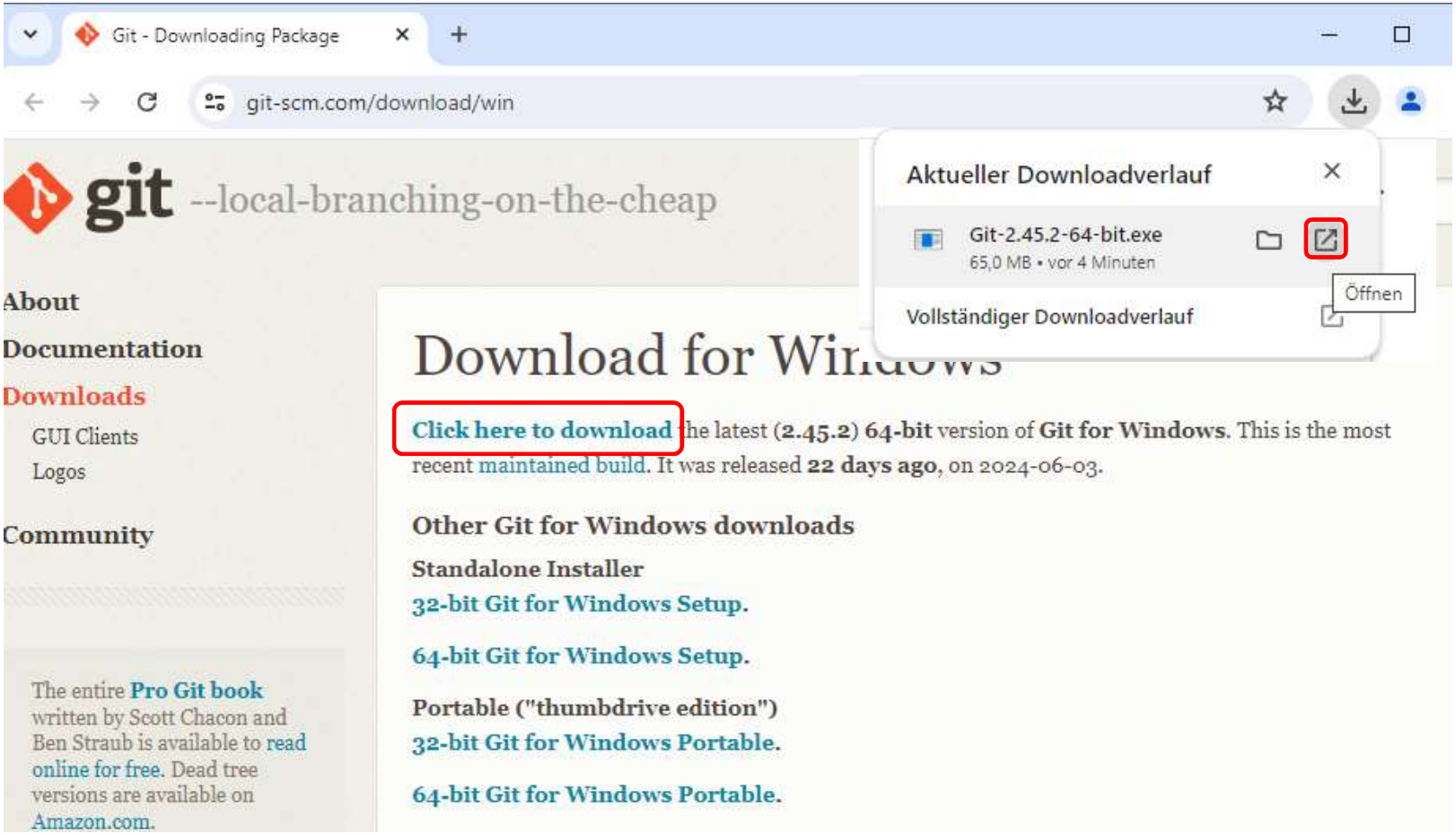
Install Git on your Windows PC

Open Visual Studio Code and select  on the left:

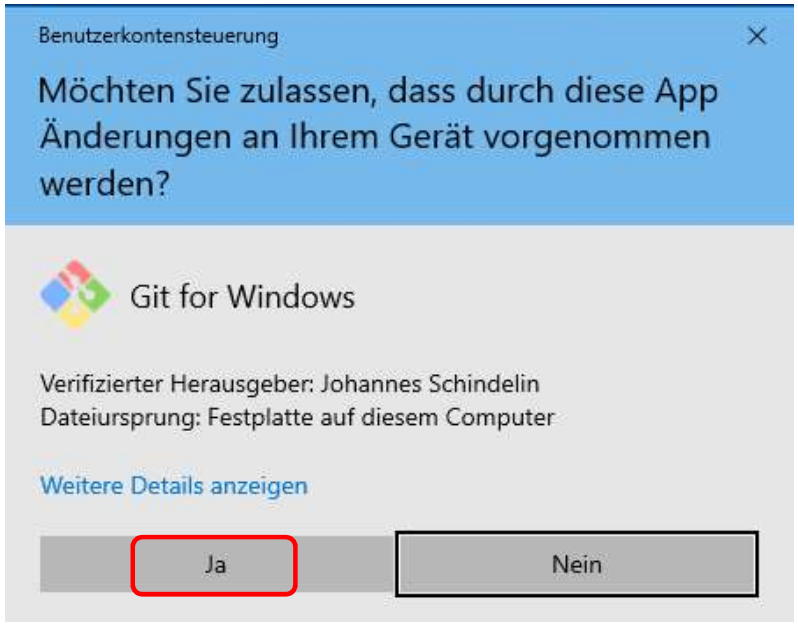




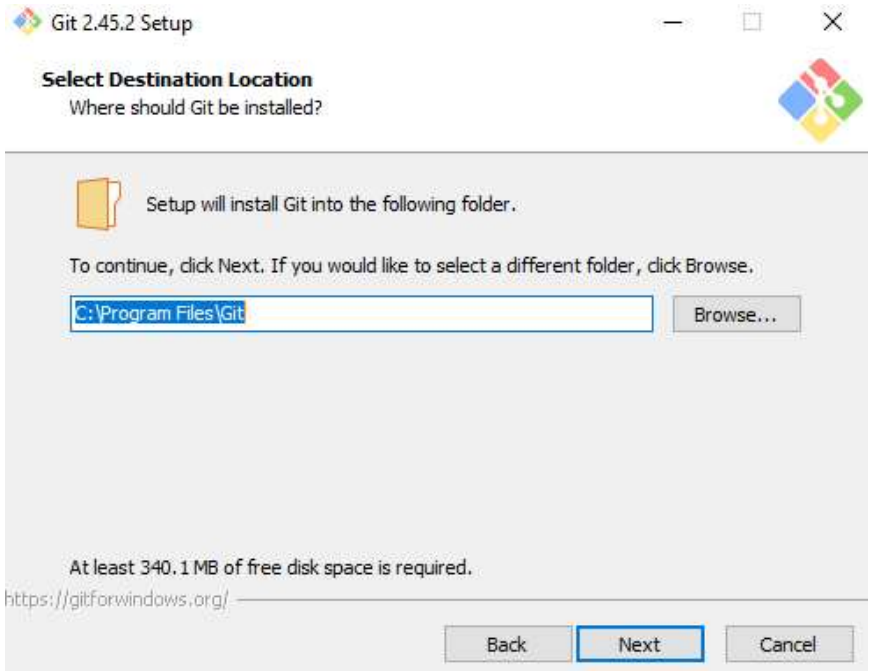
Install Git on your Windows PC

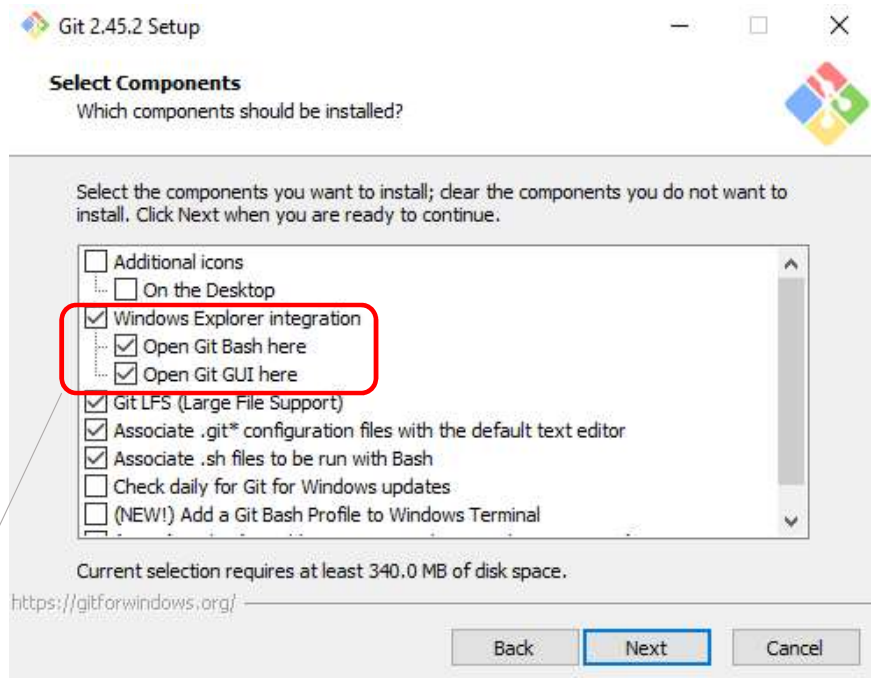


Install Git on your Windows PC



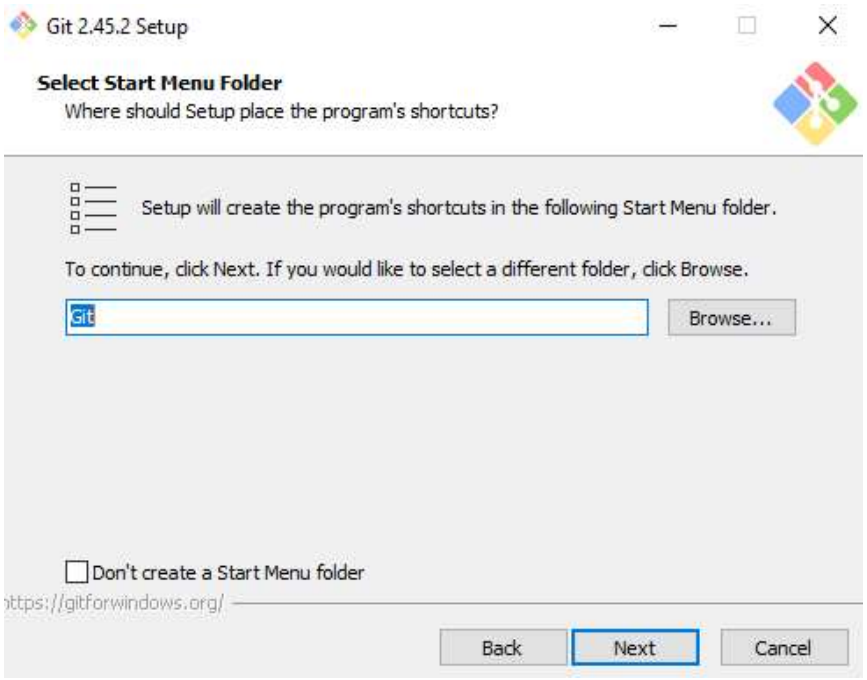
Install Git on your Windows PC

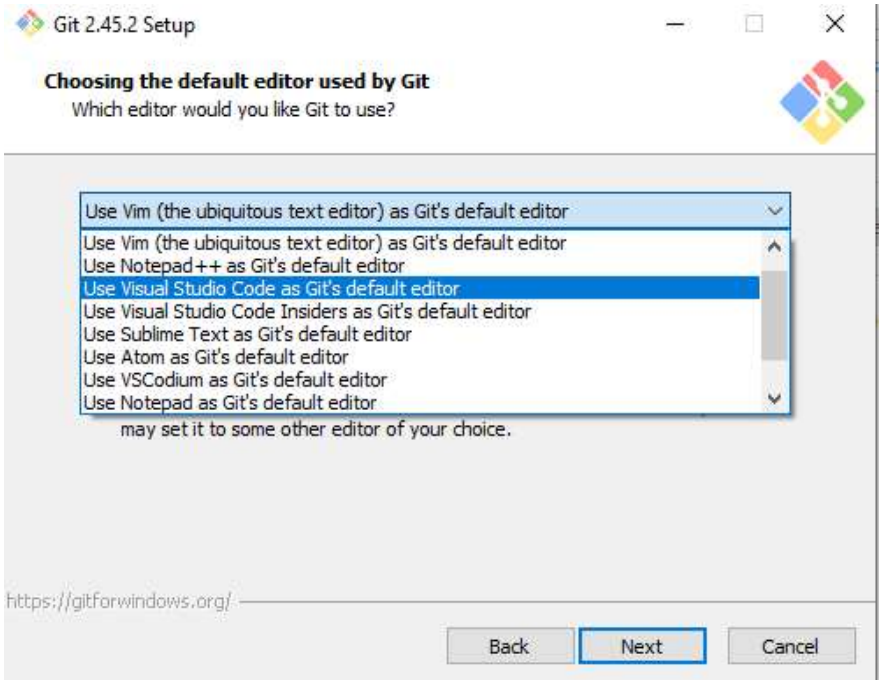




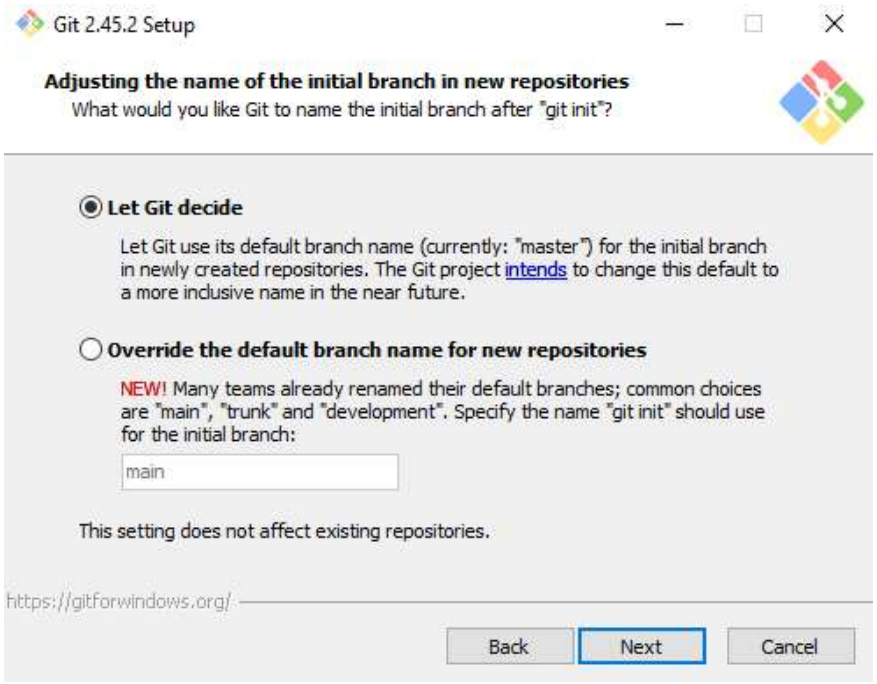
Abwählen, wenn nicht gewünscht.

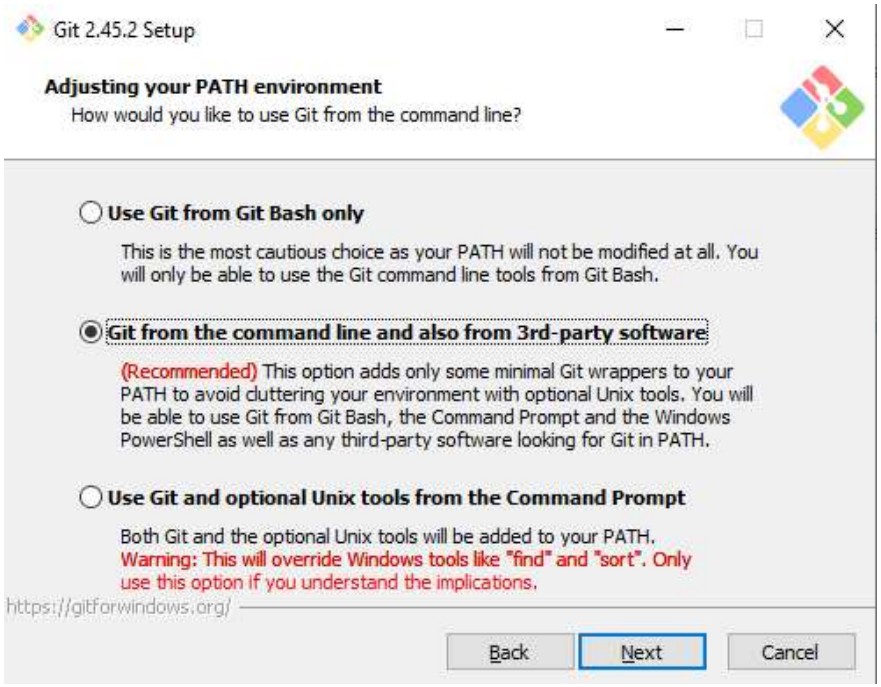
Install Git on your Windows PC



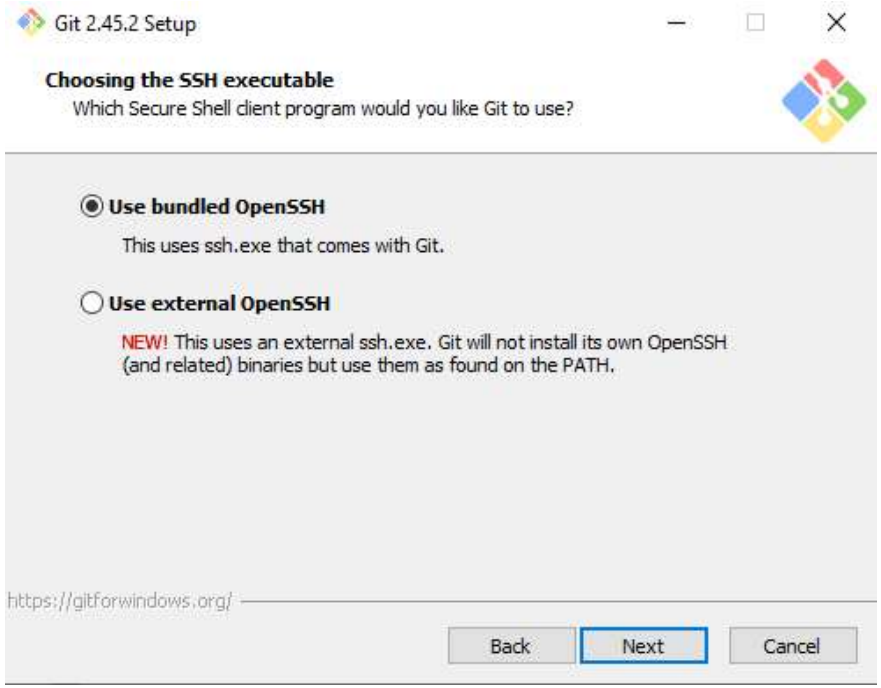


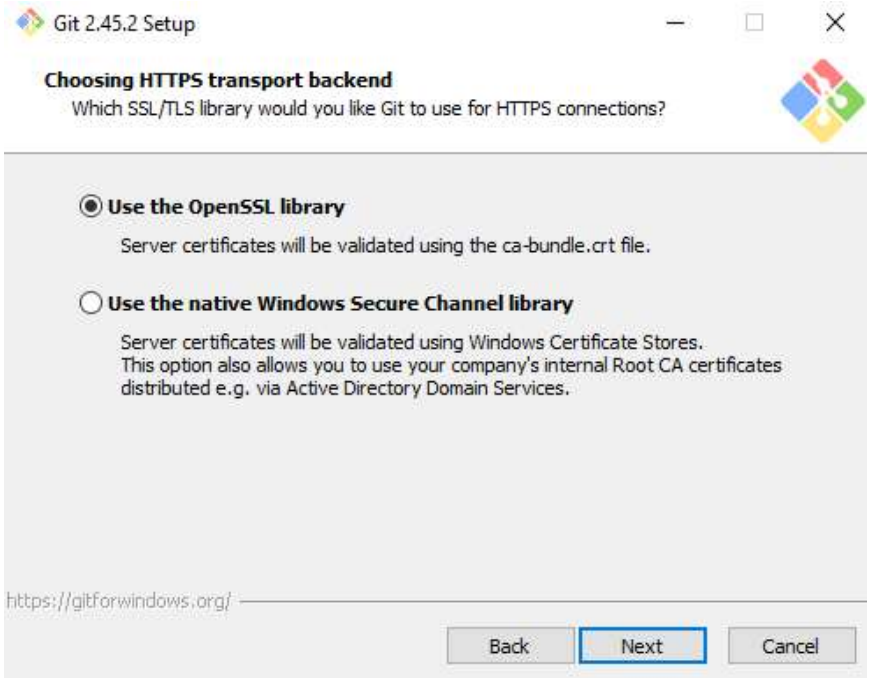
Install Git on your Windows PC



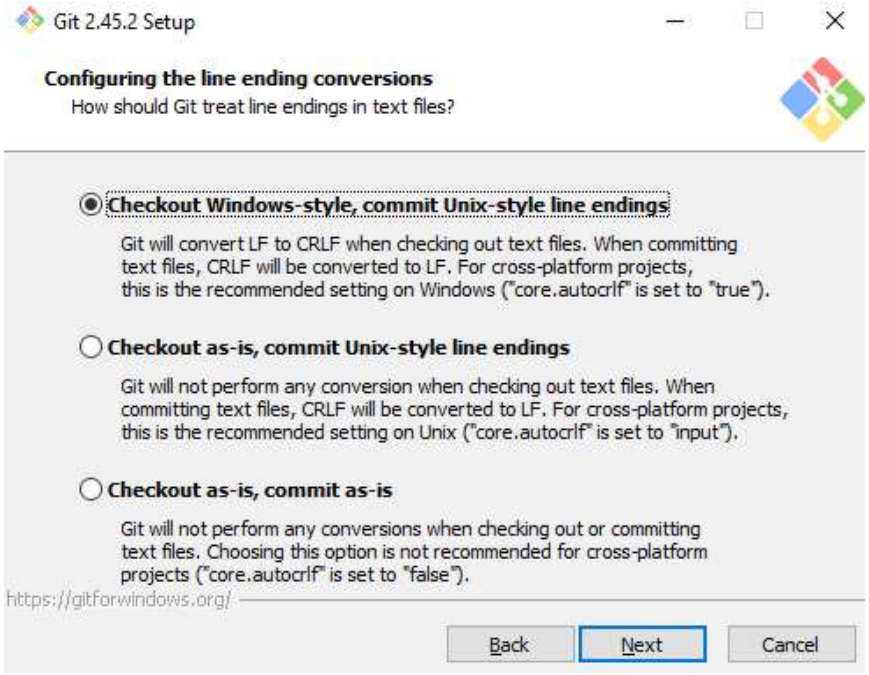


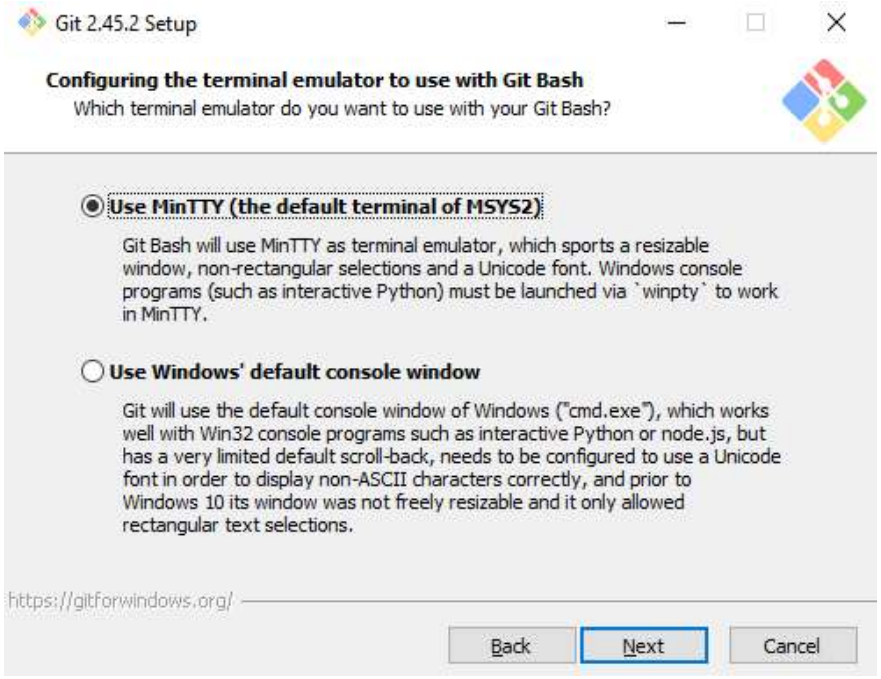
Install Git on your Windows PC



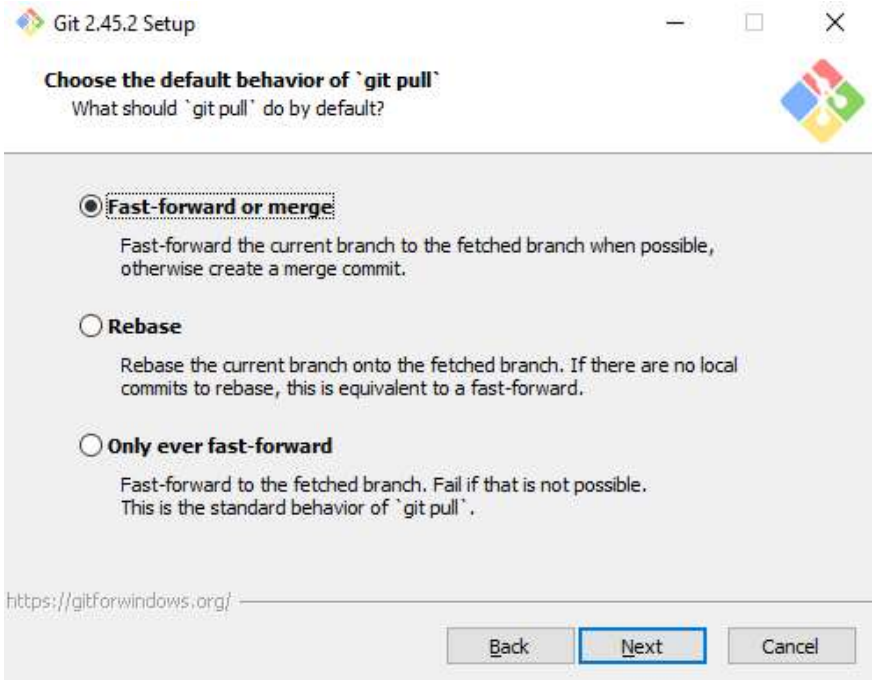


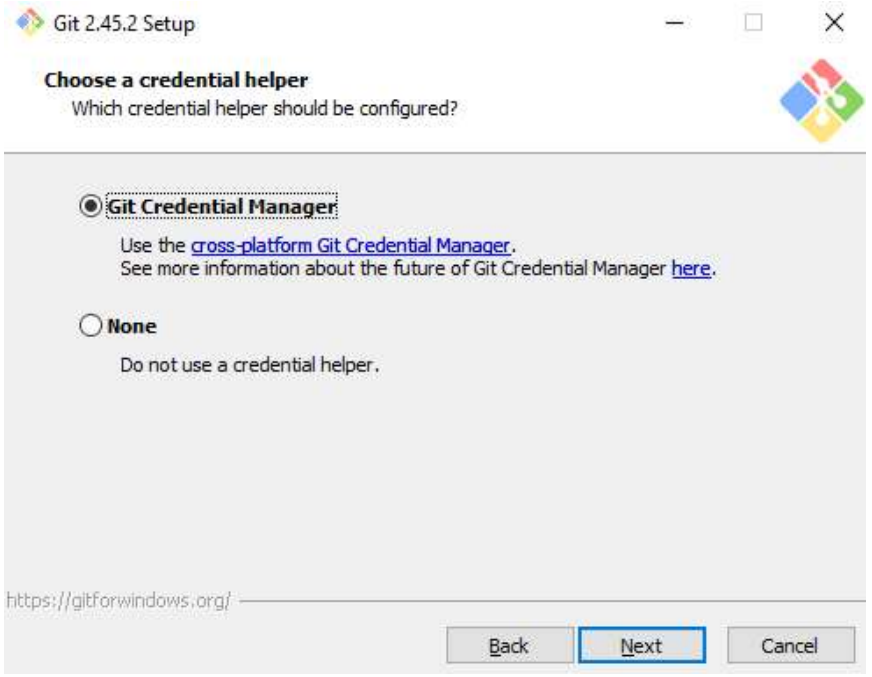
Install Git on your Windows PC



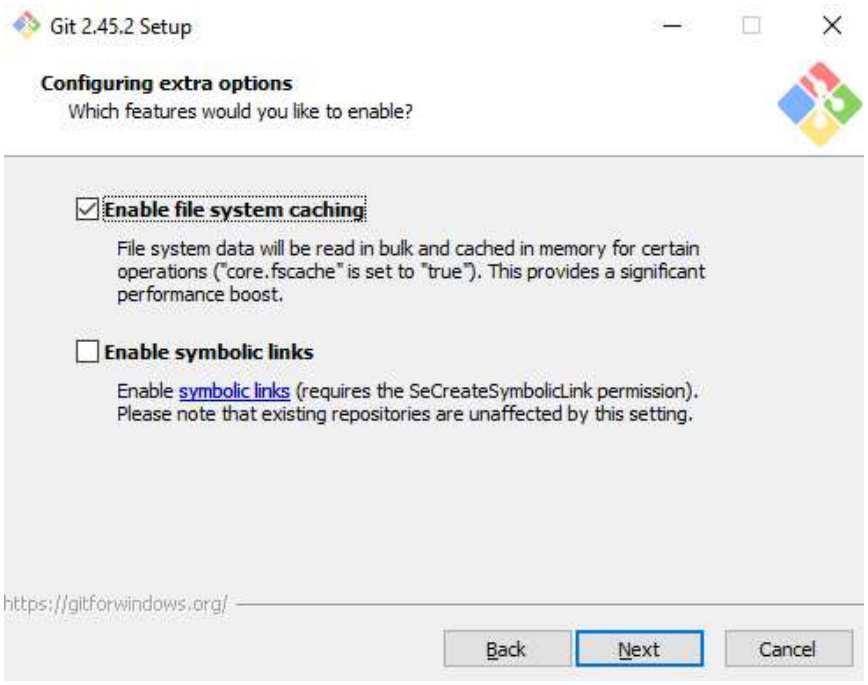


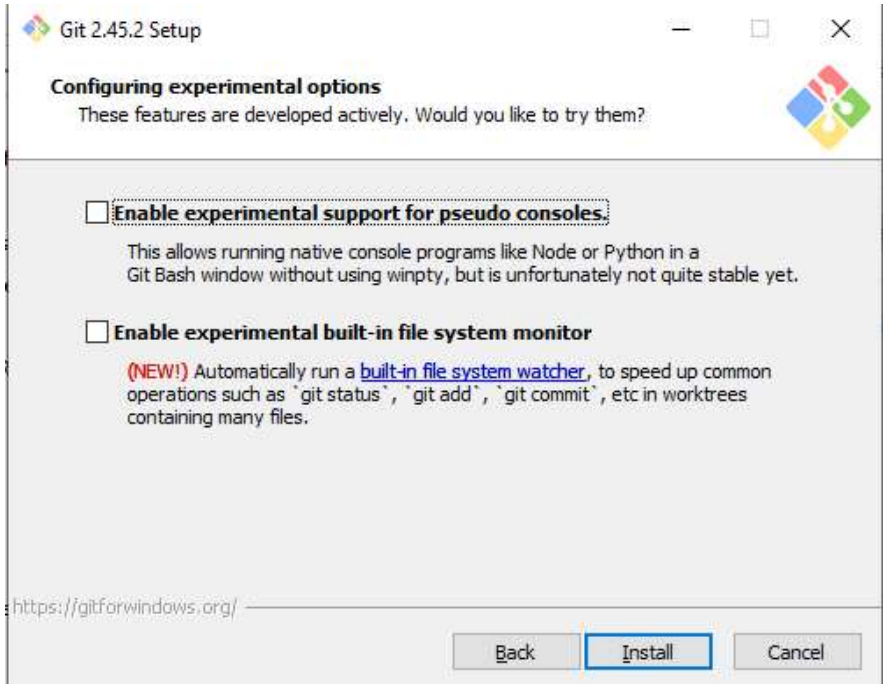
Install Git on your Windows PC



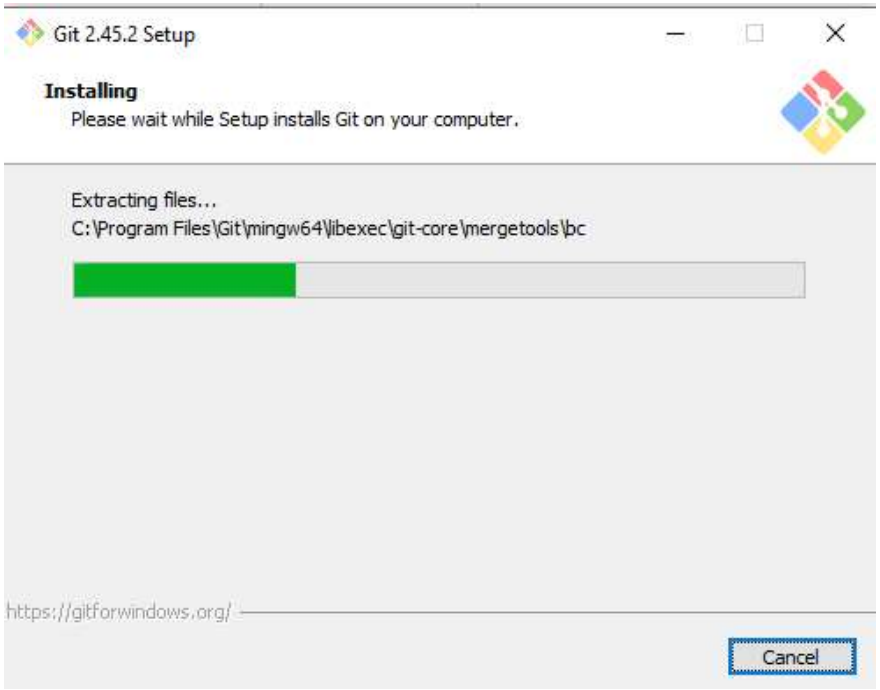


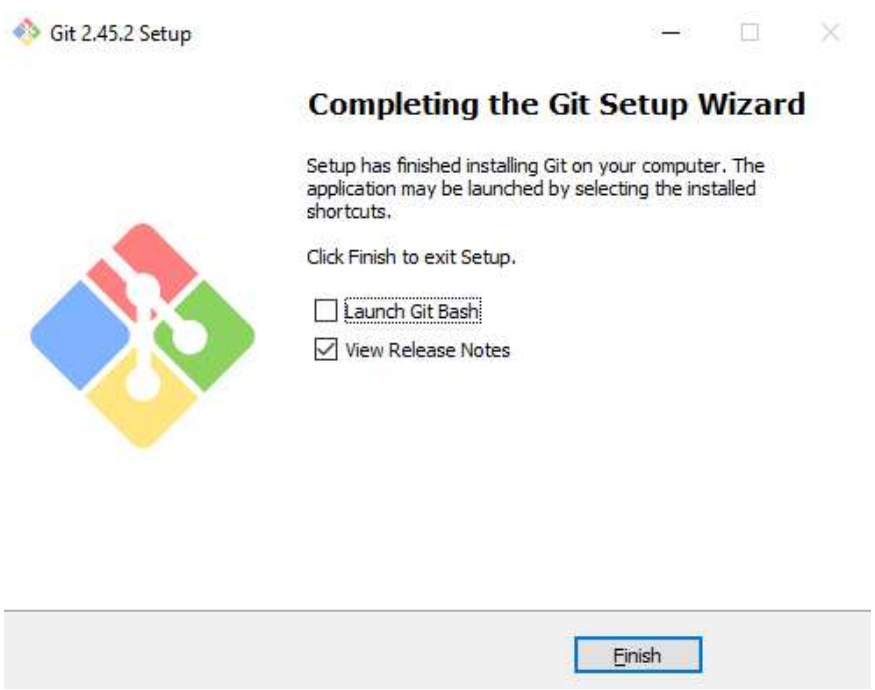
Install Git on your Windows PC





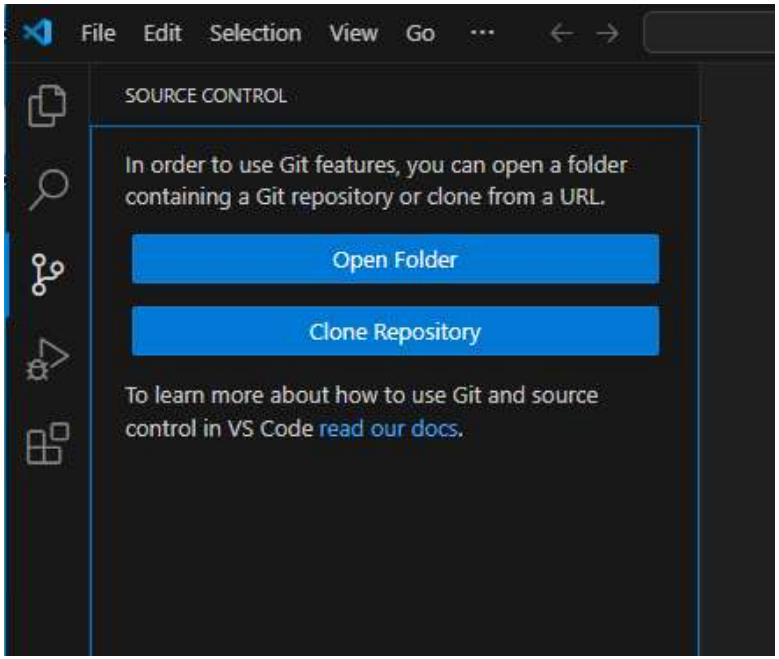
Install Git on your Windows PC



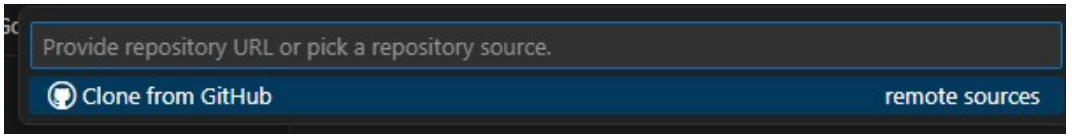


Install Git on your Windows PC

After restarting VS Code:



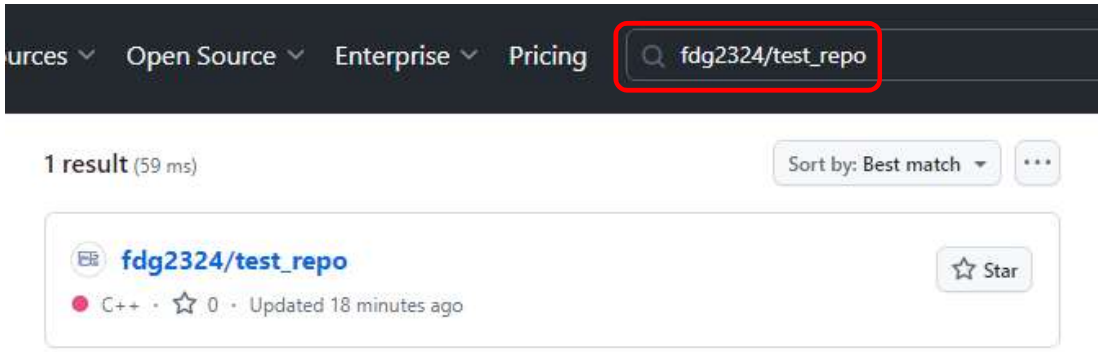
After “Clone Repository”:



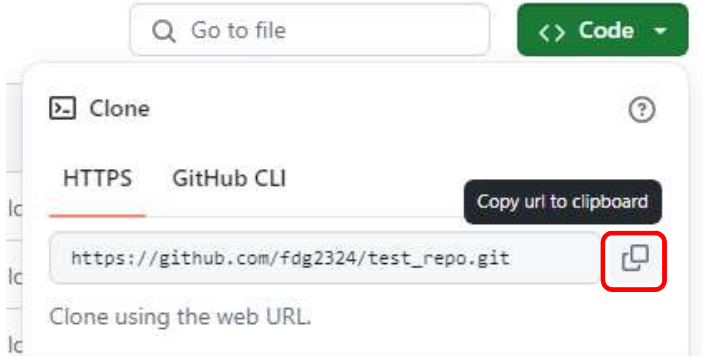
Search for repo on GitHub

Open GitHub in Browser:

Enter repository in Search field:

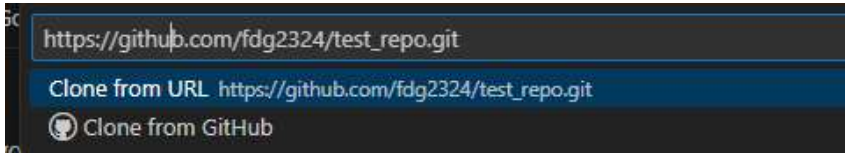


Open Repository, select <> Code and copy the url of the Repo:

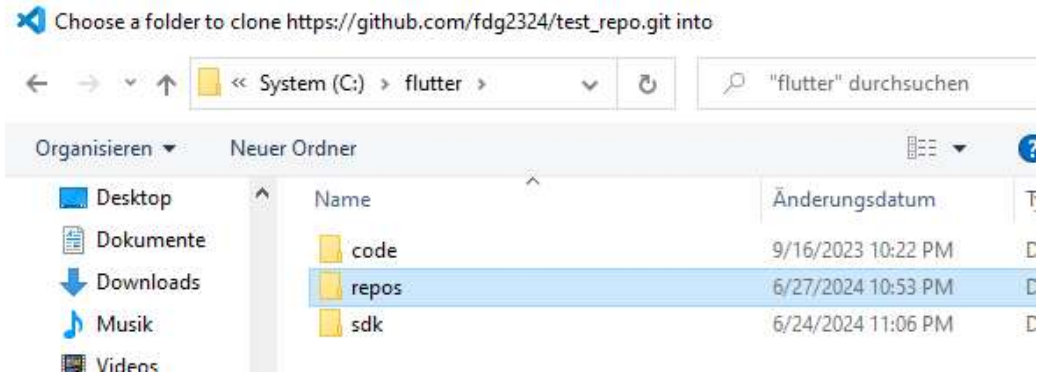


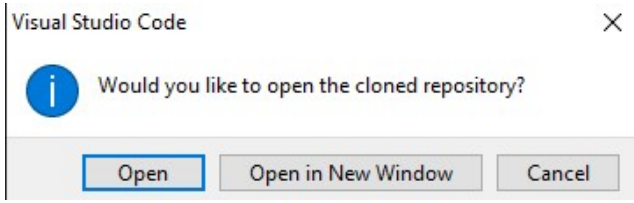
Clone Repo to your PC

Paste repo url to VS Code:

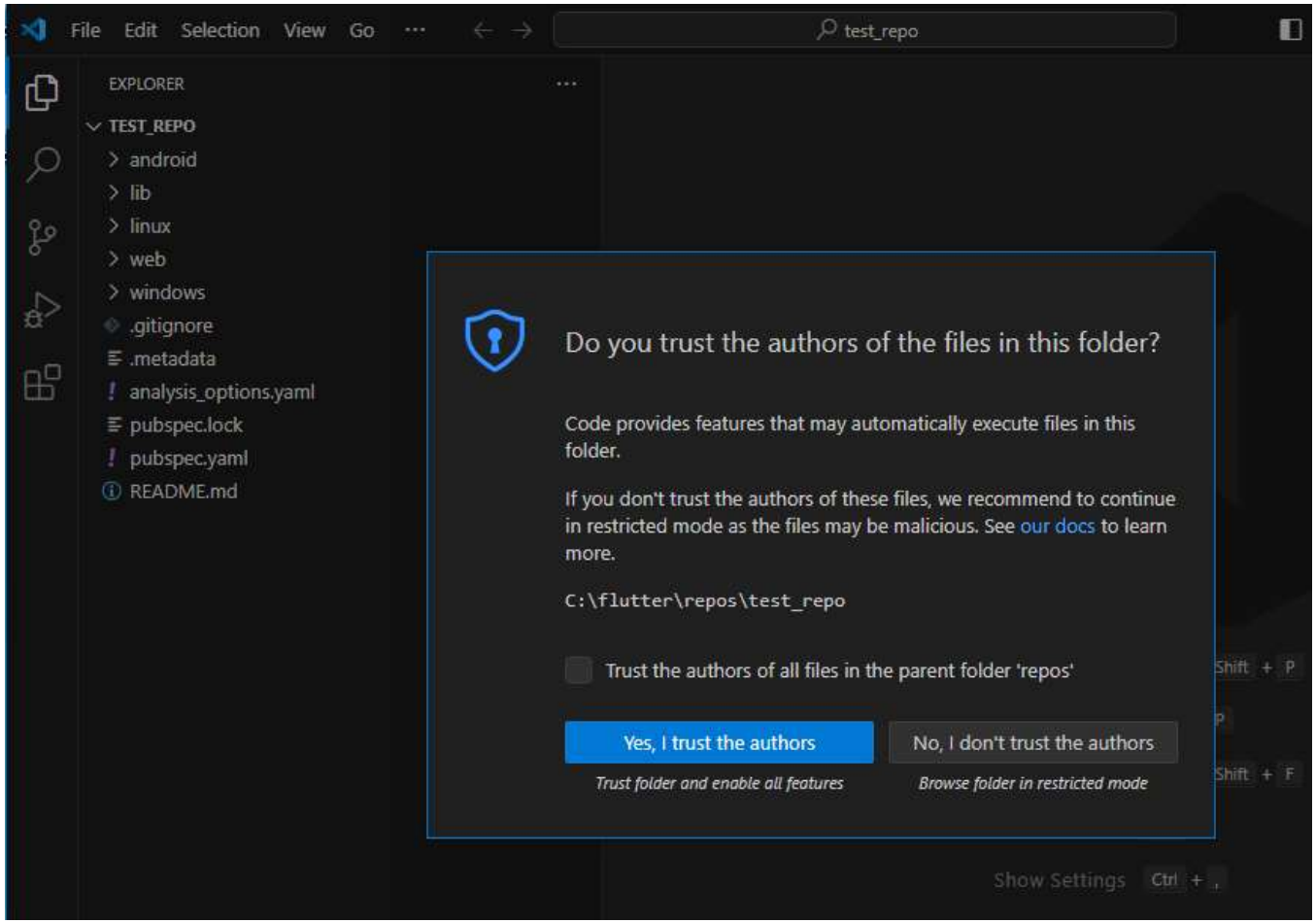


Select directory where to store the clone:



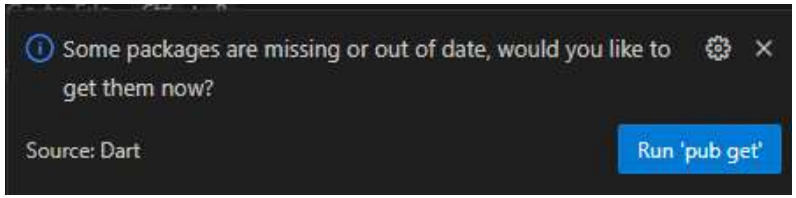


Finally the cloned repo is opened in VS Code



“pub get”

You are asked to update packages:

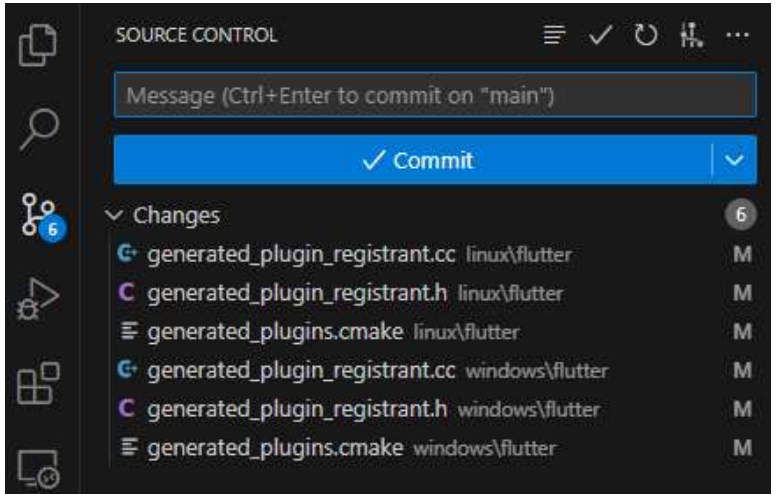


This can be repeated at any time by entering “**flutter pub get**” in Terminal:

```
PS C:\flutter\repos\test_repo> flutter pub get
Resolving dependencies... (3.2s)
Downloading packages... (5.7s)
  collection 1.18.0 (1.19.0 available)
  flutter_lints 3.0.2 (4.0.0 available)
  leak_tracker 10.0.4 (10.0.5 available)
  leak_tracker_flutter_testing 3.0.3 (3.0.5 available)
  lints 3.0.0 (4.0.0 available)
  material_color_utilities 0.8.0 (0.12.0 available)
  meta 1.12.0 (1.15.0 available)
  test_api 0.7.0 (0.7.2 available)
  vm_service 14.2.1 (14.2.4 available)
Got dependencies!
9 packages have newer versions incompatible with dependency constraints.
Try `flutter pub outdated` for more information.
PS C:\flutter\repos\test_repo>
```

Strange effect after “pub get” in cloned repo

6 files are marked as “changed”:



But when you stage them, they all disappear ?!

This is discussed (but not clarified) in https://stackoverflow.com/questions/74858985/flutter-generated-plugins-showed-with-git-changes-when-there-are-no-changes

Strange effect after “pub get” in cloned repo

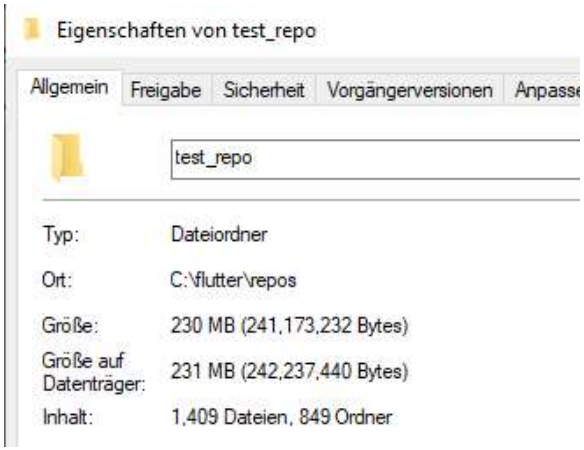
Checking this on command line:

```
C:\flutter\repos\lux_private>git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   linux/flutter/generated_plugin_registrant.cc
        modified:   linux/flutter/generated_plugin_registrant.h
        modified:   linux/flutter/generated_plugins.cmake
        modified:   windows/flutter/generated_plugin_registrant.cc
        modified:   windows/flutter/generated_plugin_registrant.h
        modified:   windows/flutter/generated_plugins.cmake

no changes added to commit (use "git add" and/or "git commit -a")
```

```
C:\flutter\repos\lux_private>git add .
warning: LF will be replaced by CRLF in linux/flutter/generated_plugin_registrant.cc.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in linux/flutter/generated_plugin_registrant.h.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in linux/flutter/generated_plugins.cmake.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in windows/flutter/generated_plugin_registrant.cc.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in windows/flutter/generated_plugin_registrant.h.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in windows/flutter/generated_plugins.cmake.
The file will have its original line endings in your working directory
```

“flutter clean”

After a build, repositories need quite a lot of disk space:



Run “flutter clean” in Terminal:

```
PS C:\flutter\repos\test_repo> flutter clean
Deleting build...                                1,037ms
Deleting .dart_tool...                             890ms
Deleting ephemeral...                              234ms
Deleting ephemeral...                              164ms
Deleting .flutter-plugins-dependencies...            8ms
Deleting .flutter-plugins...                       189ms
PS C:\flutter\repos\test_repo>
```

After a “flutter clean” you must run “flutter pub get” before the next build !

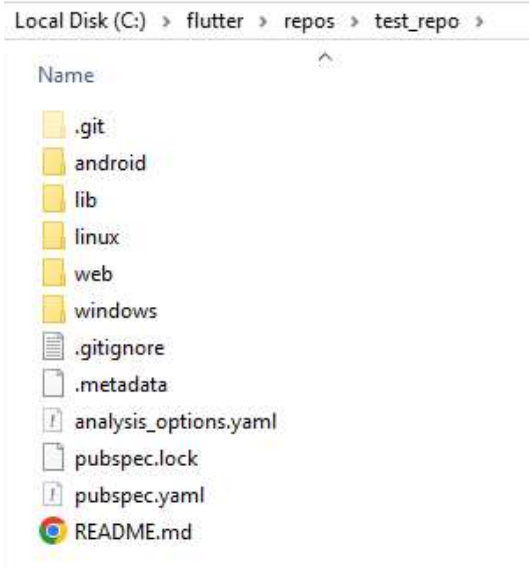
Hint: “flutter pub get” is performed automatically when you save the file “pubspec.yaml”, even if you have not done any changes therein.

Using Git on Command Prompt outside VS Code

Open a command prompt in the directory, where you want to store a local copy of the repo.

Enter “git clone URL-of-repo”:

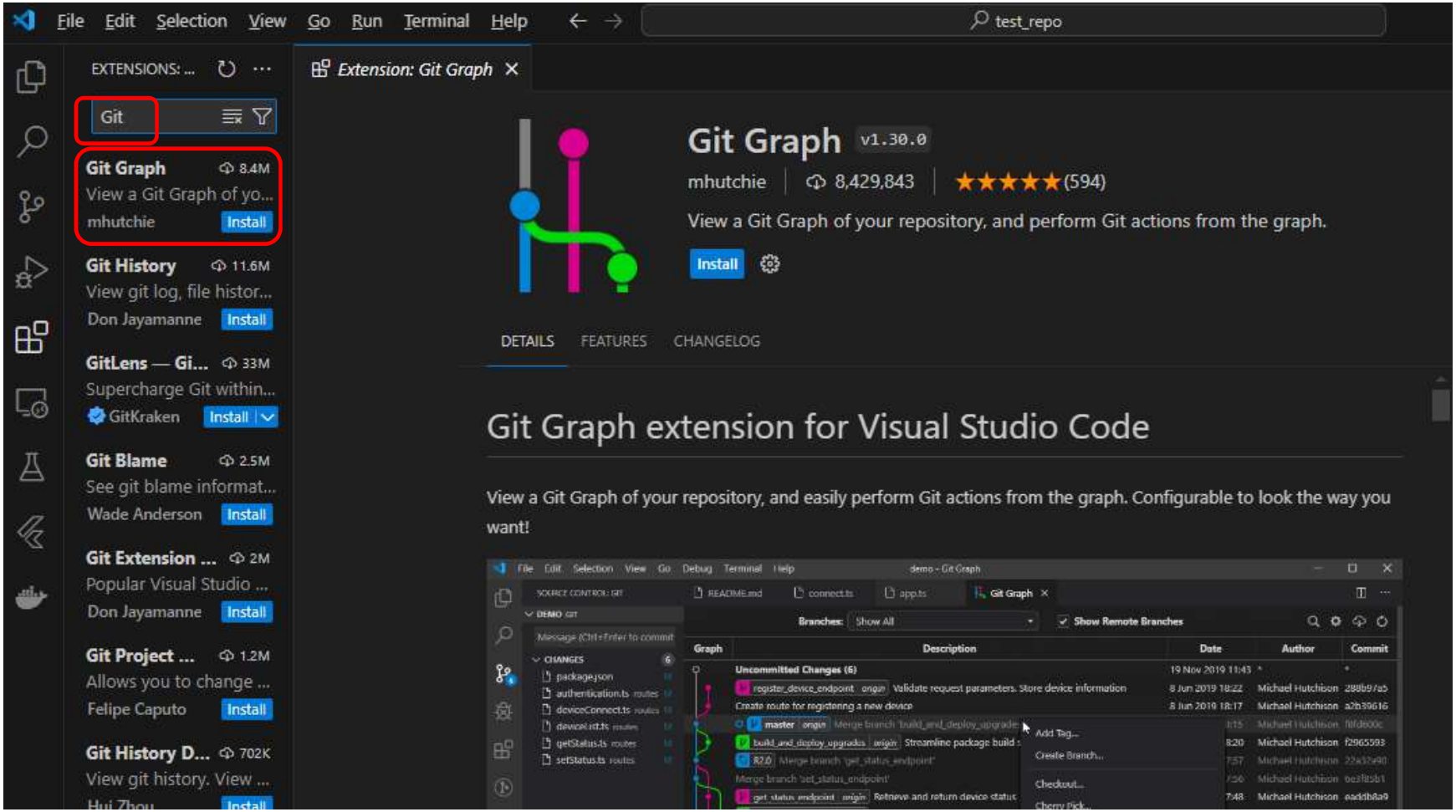
```
C:\WINDOWS\system32\cmd.exe

C:\flutter\repos>git clone https://github.com/fdg2324/test_repo.git
Cloning into 'test_repo'...
remote: Enumerating objects: 99, done.
remote: Counting objects: 100% (99/99), done.
remote: Compressing objects: 100% (66/66), done.
remote: Total 99 (delta 12), reused 98 (delta 11), pack-reused 0
Receiving objects: 100% (99/99), 80.84 KiB | 780.00 KiB/s, done.
Resolving deltas: 100% (12/12), done.
```



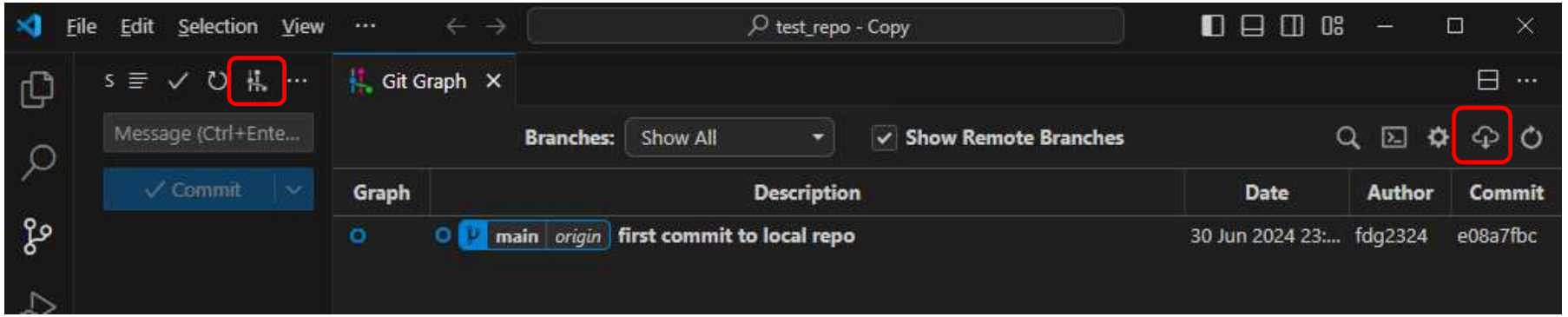
Other git command parameters

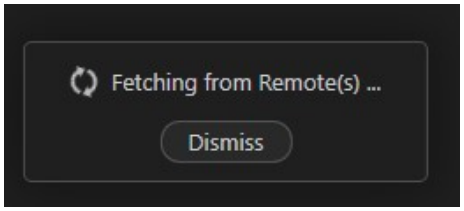
```
C:\flutter\repos>git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone             Clone a repository into a new directory
   init              Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add               Add file contents to the index
   mv                Move or rename a file, a directory, or a symlink
   restore           Restore working tree files
   rm                Remove files from the working tree and from the index
   sparse-checkout   Initialize and modify the sparse-checkout

grow, mark and tweak your common history
   branch            List, create, or delete branches
   commit            Record changes to the repository
   merge             Join two or more development histories together
   rebase            Reapply commits on top of another base tip
   reset             Reset current HEAD to the specified state
   switch            Switch branches
   tag               Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch             Download objects and refs from another repository
   pull              Fetch from and integrate with another repository or a local branch
   push              Update remote refs along with associated objects
```

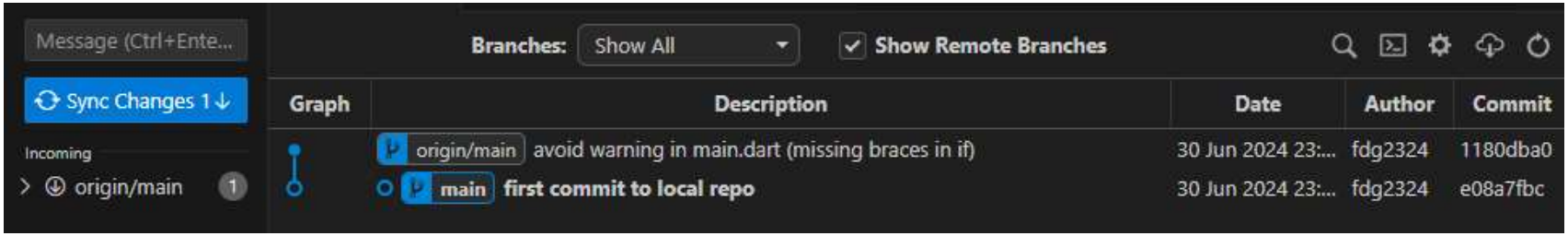
Install extension “Git Graph” in VS Code



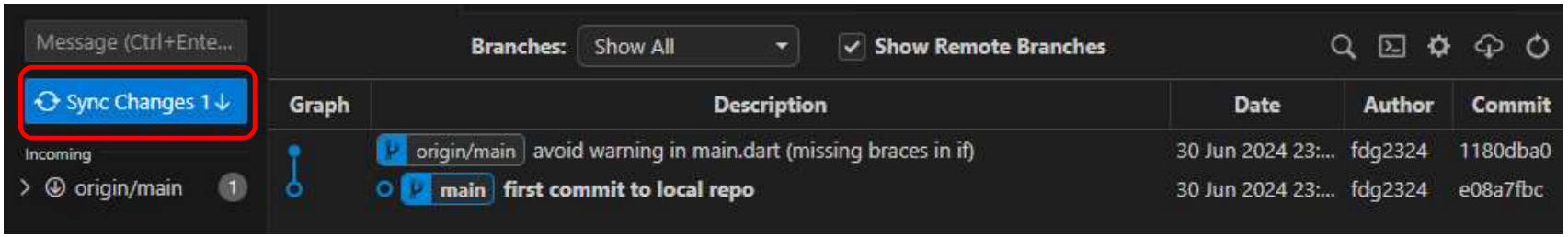
Check for new commits on GitHub

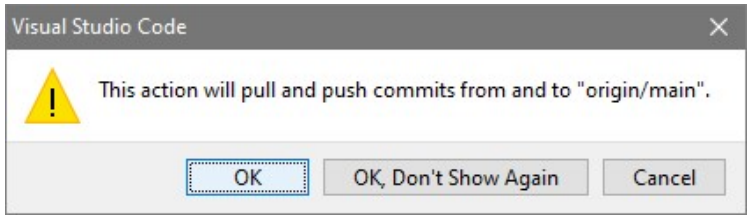


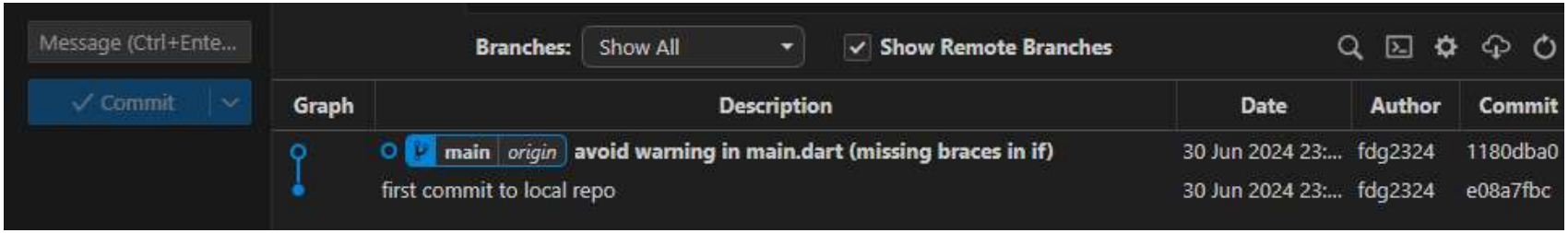




Pull latest commits from GitHub to your local repository

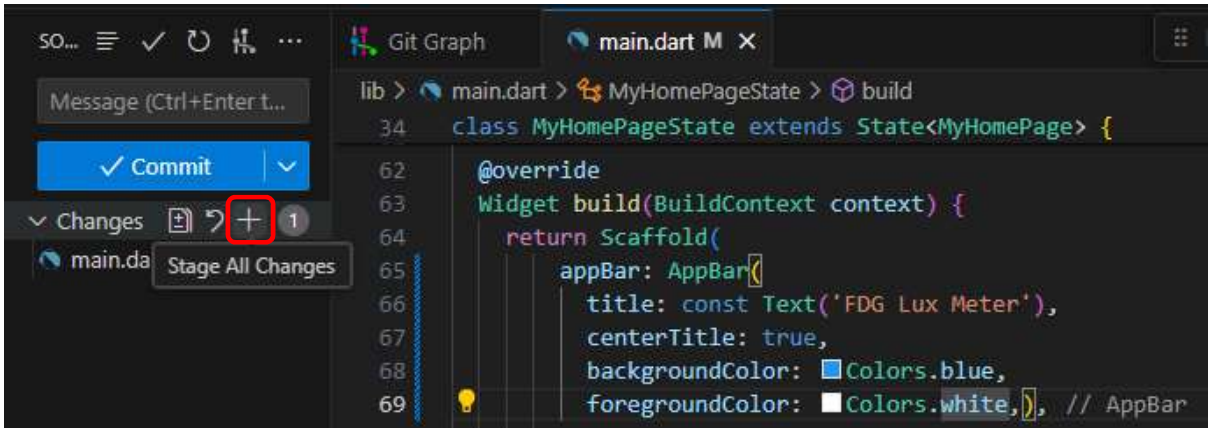




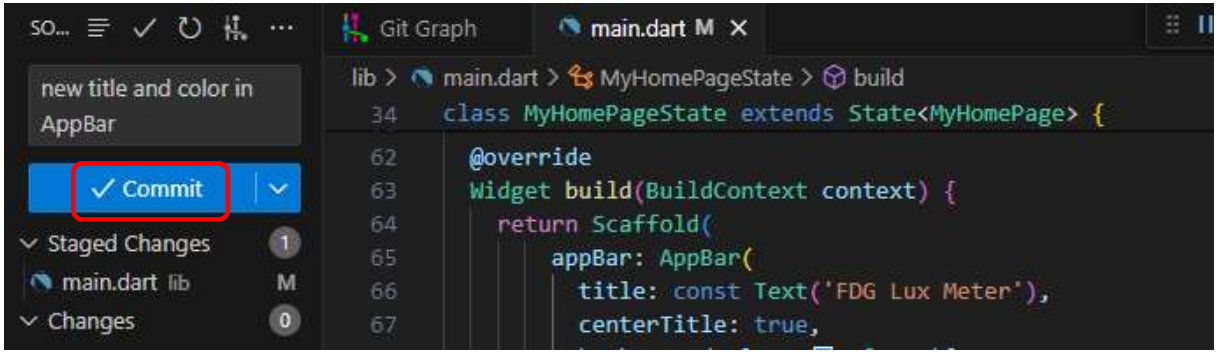


Push your local changes to GitHub (Part I)

Step 1:
Stage your files
(stage in Deutsch: arrangieren)

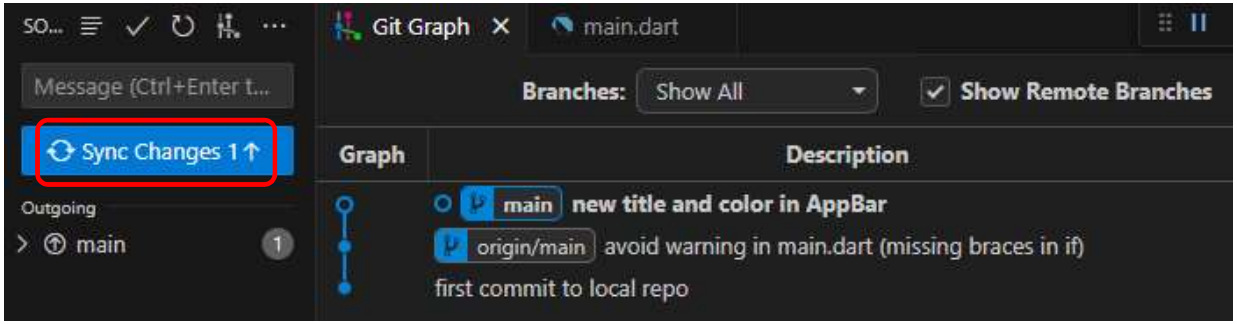


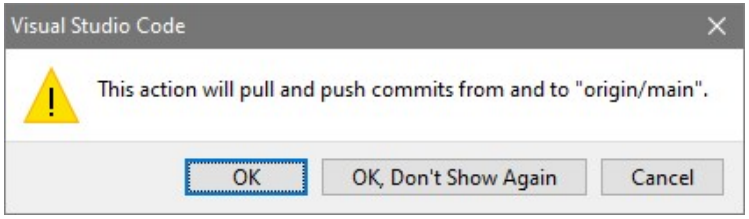
Step 2:
Commit your changes

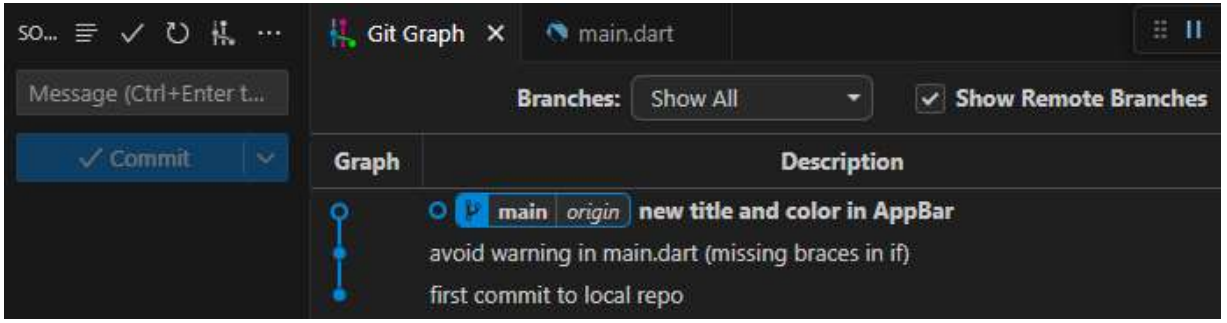


Push your local changes to GitHub (Part II)

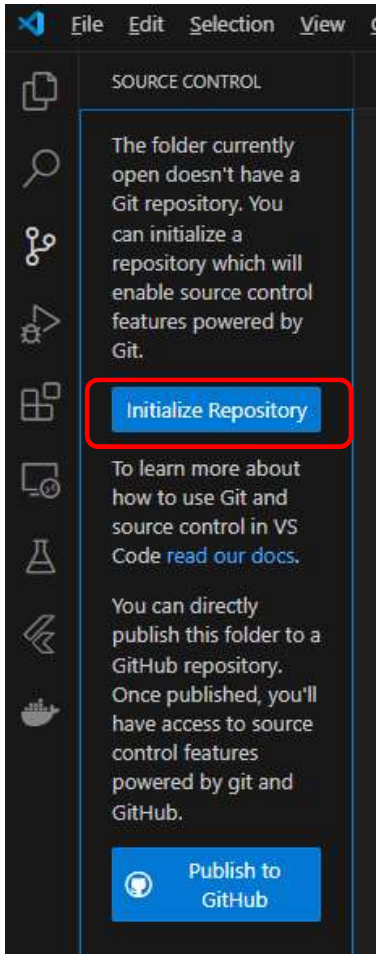
Step 3:
Sync your changes

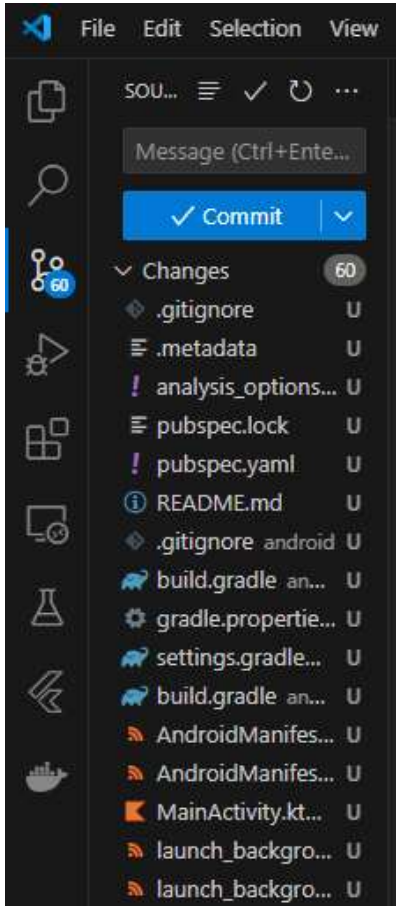


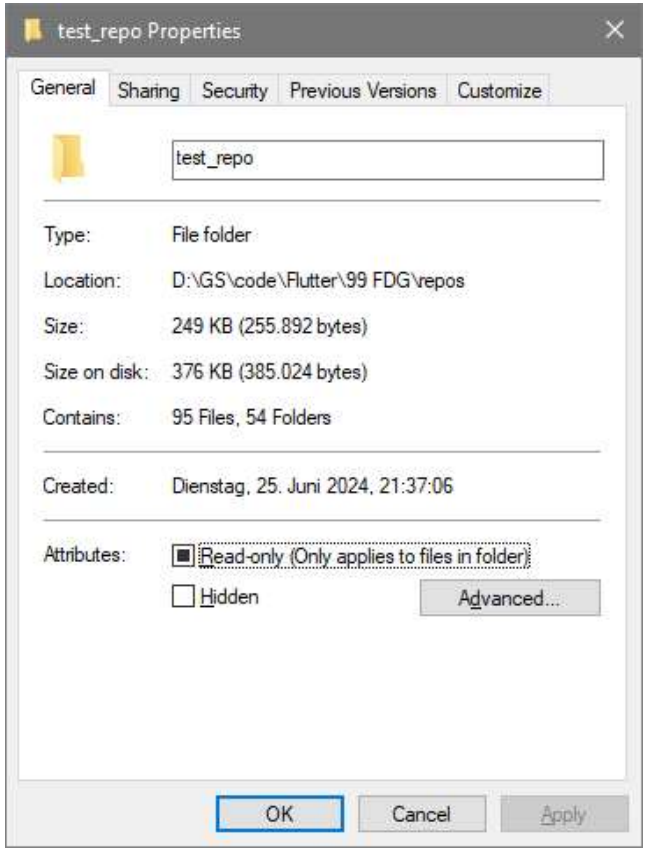




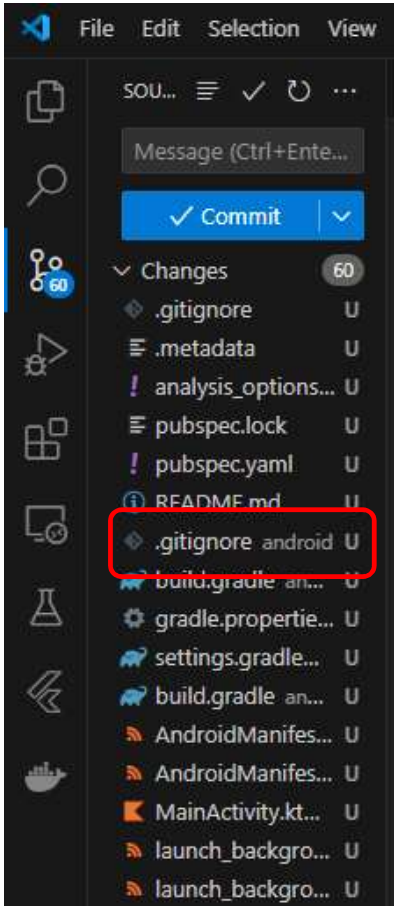
Appendix: Initialize a local git repository for your code







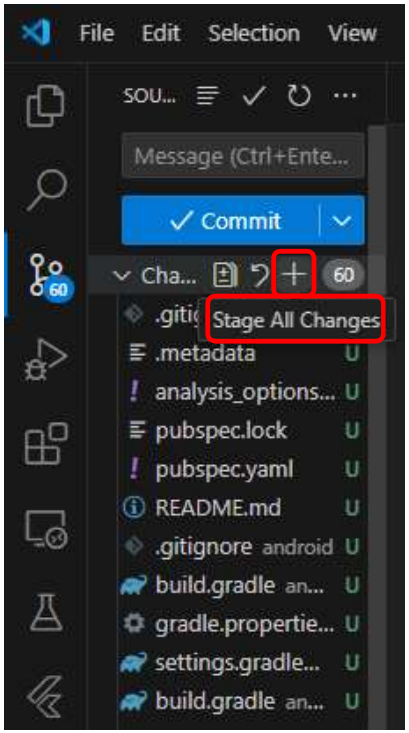
.gitignore file (controls what files are not offered for commit)

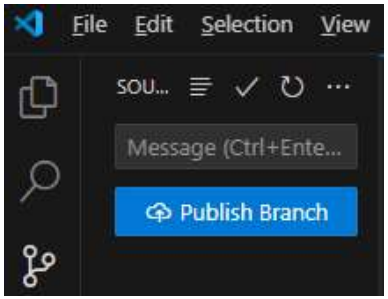


.gitignore

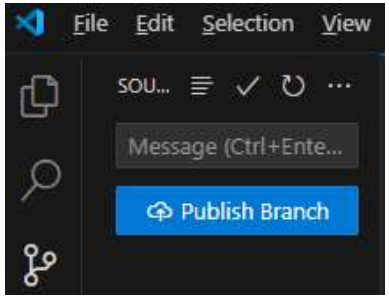
```
# Miscellaneous
*.class
*.log
*.pyc
*.swp
.DS_Store
.atom/
.buildlog/
.history
.svn/
migrate_working_dir/

# IntelliJ related
*.iml
*.ipr
*.iws
.idea/

# The .vscode folder contains launch configuration and tasks you configure in
# VS Code which you may wish to be included in version control, so this line
# is commented out by default.
#.vscode/

# Flutter/Dart/Pub related
**/doc/api/
**/ios/Flutter/.last_build_id
.dart_tool/
.flutter-plugins
.flutter-plugins-dependencies
.pub-cache/
.pub/
/build/

# Symbolication related
app.*.symbols

# Obfuscation related
app.*.map.json

# Android Studio will place build artifacts here
/android/app/debug
/android/app/profile
/android/app/release
```

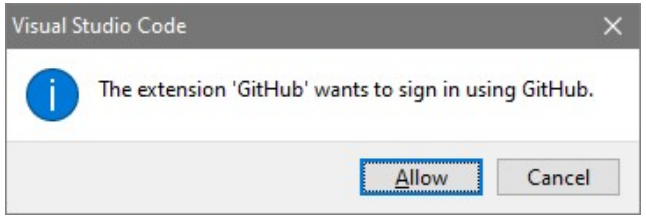
“Staging” files and Commit

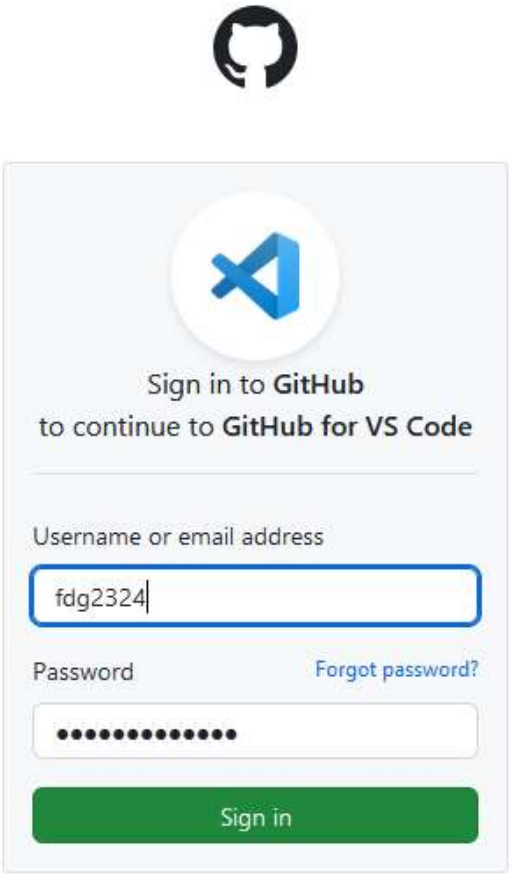


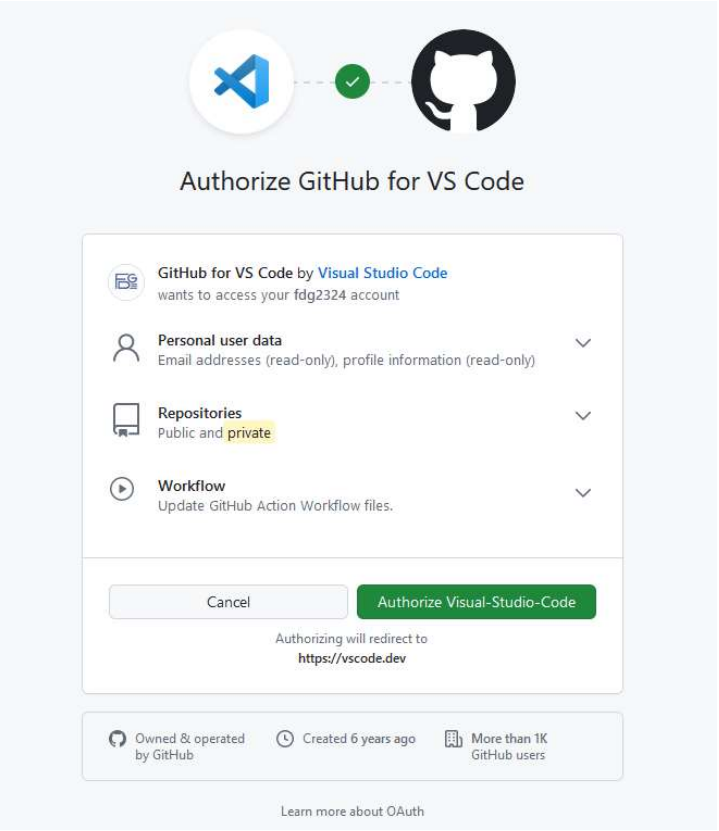


Publishing to GitHub – Authorization



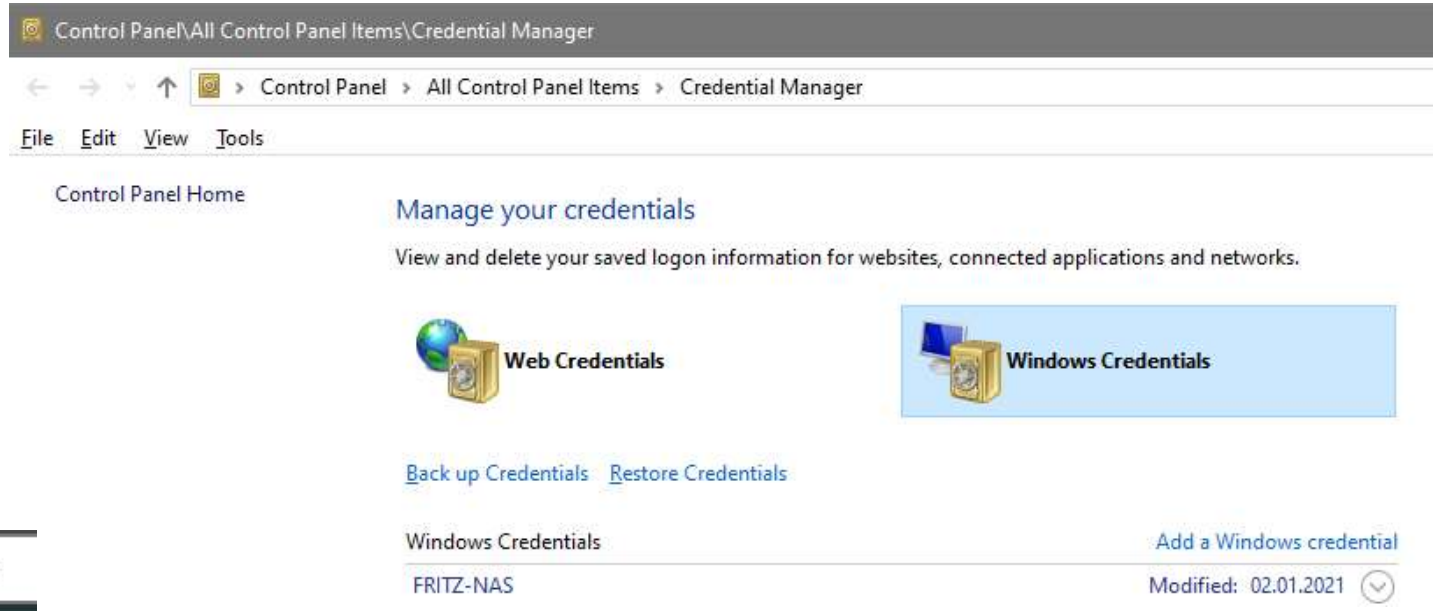






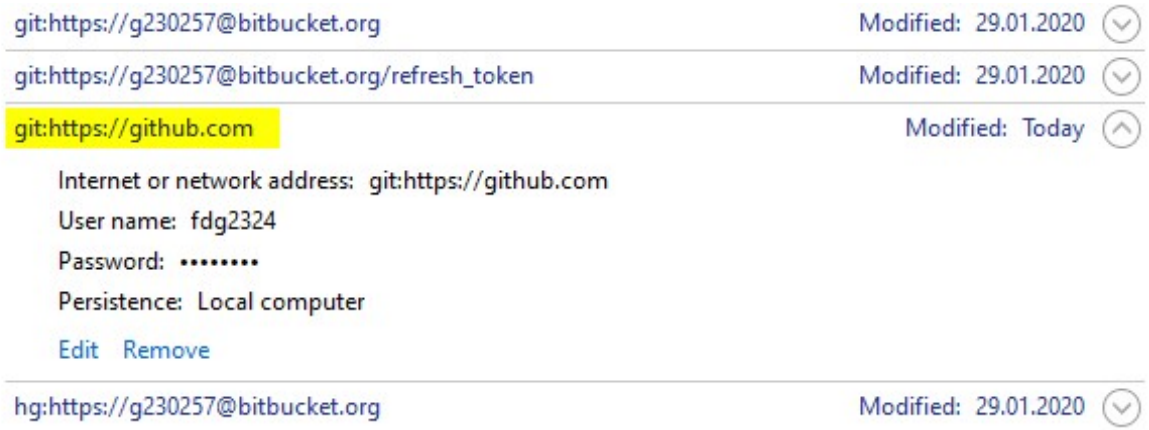
Switch to another GitHub account on your PC

Once you have authorized at GitHub for VS Code, your account is memorized in Windows CredentialManager. To open this CredentialManager, use Search button in Taskbar.



Switch to another GitHub account on your PC

Scroll down to the credential entry for GitHub:



In case you want to switch to another GitHub account for your next push, logout from GitHub in your Browser and remove this entry for GitHub in your Credential Manager.

Next time you try a push in VS Code, you are asked again for authorization.