Version Control: (git/github)

Feb 26

Agenda

- In this lecture you will learn:
  - Motivation for using Github
  - How to keep a history of your coding projects
    - Git Installation and creating account on Github
    - Some (out of many) useful commands
  - How to collaborate with other people efficiently when doing coding

Motivation

- When you work on a CS project or an assignment, the progress you make is incremental, therefore you want to keep track of your progress and.
- A lot of new coders start making copies of their code with multiple versions like:
  - Filename_v1.cpp
  - Filename_v2.cpp
  - ...
  - Filename_v10.cpp

Motivation

Some employers take a look at your Github profile

Upload CV *

Please use the Europass format (https://europass.cedefop.europa.eu/documents/curriculum-vitae). Only PDF, max. 10MB.

Add file

Upload Motivation Letter *

Only PDF, max. 1MB, max. 1 page.

Add file

Link to your code repository (optional)

Your answer

Link to video demonstration (optional)

Your answer

Motivation

Some employers take a look at your Github profile

Software Developer / Product Development

Ziiva Inc.
Roanoke, VA 24018

Employer actively reviewed candidates 8 days ago

Apply Now | Save this job

Are you independent, innovative, and looking for a fast-paced, yet casual environment in software product development? If so we would love to talk to you.

Our customers are unique. They tend to have very particular business requirements, and look to us to provide targeted solutions using our highly customizable software. You'll be enhancing our software product to support a highly configurable system, and working with clients to configure and customize the system for their needs.

We're looking for someone who finds this particularly exciting.

The ideal candidate will possess the following qualities:

- You love systems. You spend your time putting systems and solutions together in your head for fun. If this doesn't describe you, this isn't the job for you.
- You love writing code. You've written in multiple languages, and are open minded in terms of technical solutions. Code is fun. You have a github, codepen, or other such account.
- You're a fast, independent learner and a hard core problem solver. If a solution isn't obvious, you find it one way or another.
- You enjoy solution driven discussions with customers. This job doesn't require chit chat. It does require a good understanding of a customer's needs and the ability to drive these conversations. It requires being able to read between the lines and understand what's needed and what might even be better.
- You love data flow / work flow diagrams, and spreadsheets.

Responsibilities

- Consult with clients and coworkers to develop website requirements
- Design, code, test, document, debug, and maintain robust websites to meet stated goals

Version control

- Version control helps you keep track of your progress in CS projects.
- Using Github, you can have a backup of your code.
- In case you lose your code due to hard drive or laptop malfunction, you can restore your project code.
- You might be using multiple PCs and laptops and you wanna access the most recent version of your code
- **Not mandatory** for this course, just a recommendation

Version control

Accessing project files using multiple laptops or PCs

Dropbox/Drive are good, but github is better

Git Installation

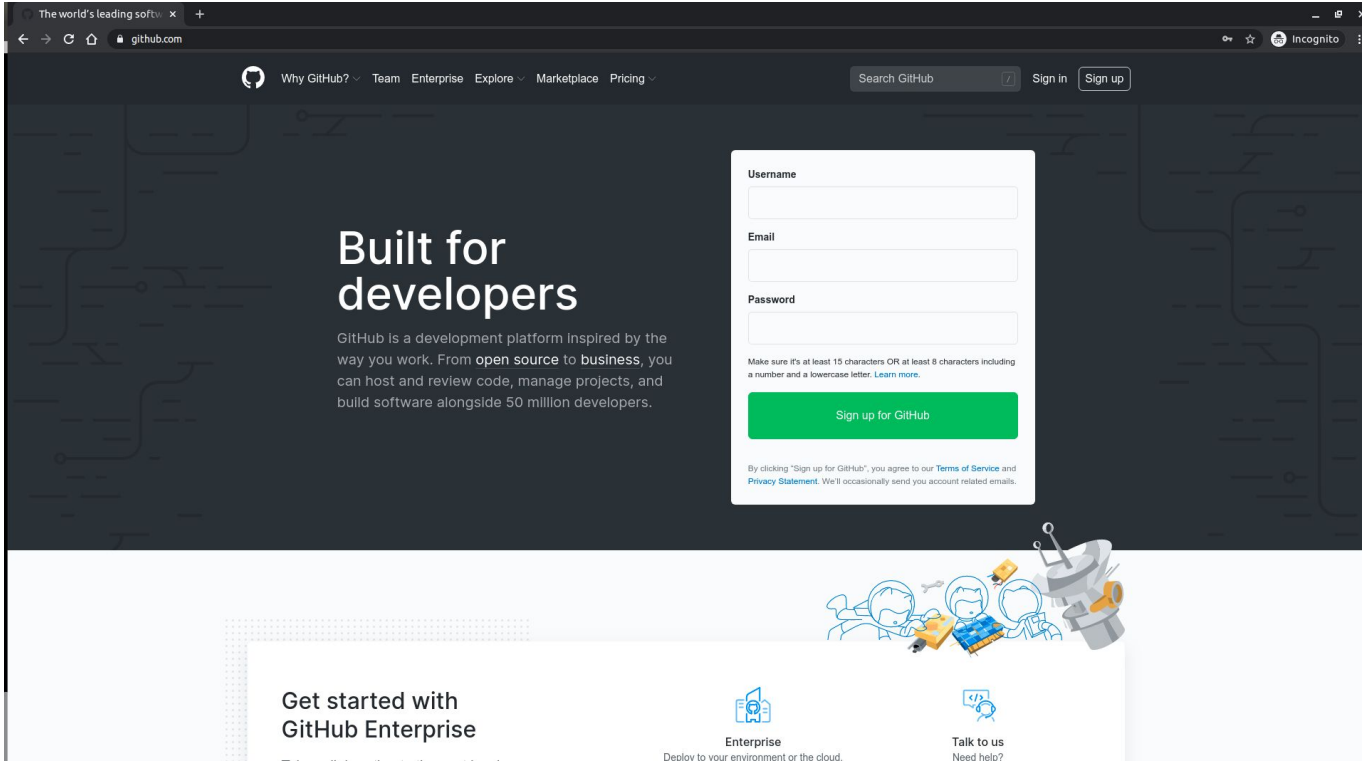
(Debian-based Linux such as Ubuntu):

```
$sudo apt-get install -y git
```

(Windows):

Download here

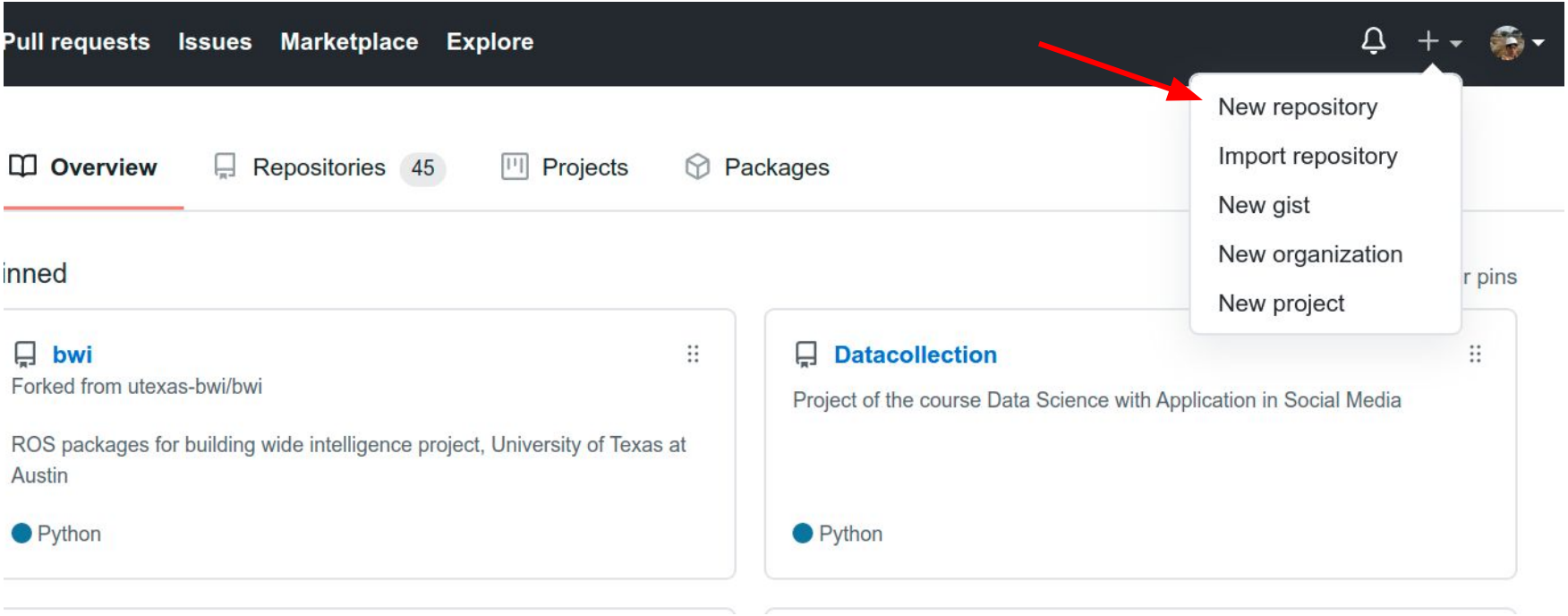
Create Github account



How to start a project with version control?

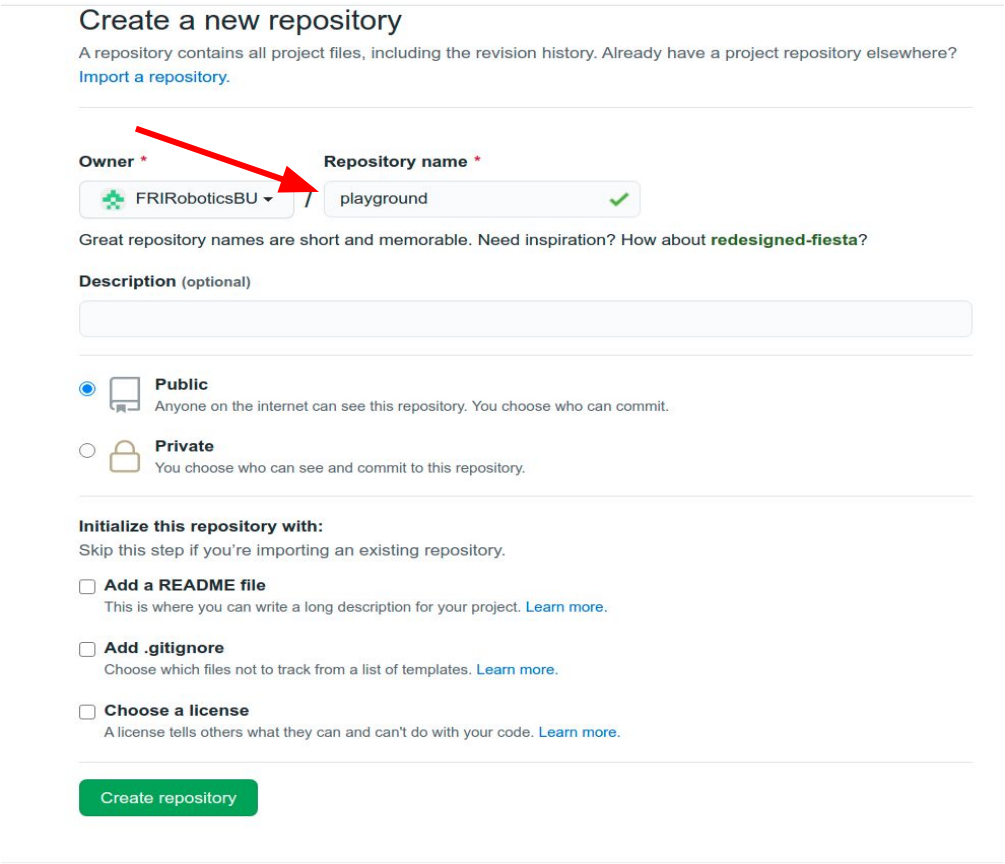
- Multiple ways:
  - 1- Create a new repository on your Github account first, clone it on your laptop, and then start working on your project
  - 2- You can create a repository for an already **existing** folder

Create new repository



Create new repository

- Choose a name
- Decide if it is public or private



Clone the repository

Copy the repo link to the clipboard

FRIRoboticsBU / **playground**

Watch

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

Give access to the people you work with

You should give access to the collaborators and teams you need to work with.

Add teams and collaborators

Quick setup — if you've done this kind of thing before

or HTTPS SSH https://github.com/FRIRoboticsBU/playground.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# playground" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M master
git remote add origin https://github.com/FRIRoboticsBU/playground.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/FRIRoboticsBU/playground.git
git branch -M master
git push -u origin master
```

Clone the repository

```
saeid@saeid-Lenov: ~
saeid@saeid-Lenov: ~ 96x25
(base) saeid@saeid-Lenov:~$ git clone https://github.com/FRIRoboticsBU/playground.git
```

git clone *[repository link]*

Go to the repository folder

cd *[repo_name]*

```
saeid@saeid-Lenov: ~
saeid@saeid-Lenov: ~ 87x23
(base) saeid@saeid-Lenov:~$ cd playground/
```

.git

Though current repo folder is empty, but there are some hidden files there. Let's take a look at them by typing

$ls -a

```
(base) saeid@saeid-Lenov:~/playground$ ls -a
.  ..  .git
(base) saeid@saeid-Lenov:~/playground$
```

.git folder stores all the necessary information about your coding progress

Create a file

In this case, I want to write a python code that sorts a list. First, I create the file using

$ touch sort.py

Or you can find other ways to create a file

Do some coding

```
def bubble_sort(nums):

    swapped = True
    while swapped:
        swapped = False
        for i in range(len(nums) - 1):
            if nums[i] > nums[i + 1]:

                nums[i], nums[i + 1] = nums[i + 1], nums[i]

                swapped = True


random_list_of_nums = [5, 2, 1, 8, 4]
bubble_sort(random_list_of_nums)
print(random_list_of_nums)
```

Let's backup my code

Git status helps you find out what branch you are in, what files are being tracked and what files are commited.

$ git status

```
(base) saeid@saeid-Lenov:~/playground$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        sort.py

nothing added to commit but untracked files present (use "git add" to track)
```

Sort.py is not tracked yet

Add your code to the tracked files

Now we want sort.py to be tracked:

Alternatively, for multiple files:

Git add file1.py file2.py file3.py

```
(base) saeid@saeid-Lenov:~/playground$ git add sort.py
```

Let's double-check:

$ git status

```
(base) saeid@saeid-Lenov:~/playground$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   sort.py
```

Now, time to commit the code

Commit the changes

$ git commit -m "brief message reminding you/collaborators of what changes you made"

```
(base) saeid@saeid-Lenov:~/playground$ git commit -m "initial commit"
[master (root-commit) ec17b80] initial commit
 1 file changed, 17 insertions(+)
 create mode 100644 sort.py
```

Your first time, you may be asked to enter your name and email before being able to commit your changes

Commit the changes

Let's double-check:

$ git status

```
(base) saeid@saeid-Lenov:~/playground$ git status
On branch master
Your branch is based on 'origin/master', but the upstream is gone.
  (use "git branch --unset-upstream" to fixup)

nothing to commit, working tree clean
```

Let’s make more updates to the code

Now, we want to make more updates to the code.

Just for illustration/educational purposes, I would like to add comments to my code and commit again.

In a real project/assignment, you want to make commits after incremental progress.

Let’s update sort.py

Let's make more updates to the code

```
1  def bubble_sort(nums):
2      # We set swapped to True so the loop looks runs at least once
3      swapped = True
4      while swapped:
5          swapped = False
6          for i in range(len(nums) - 1):
7              if nums[i] > nums[i + 1]:
8                  # Swap the elements
9                  nums[i], nums[i + 1] = nums[i + 1], nums[i]
10                 # Set the flag to True so we'll loop again
11                 swapped = True
12
13
14
15 random_list_of_nums = [5, 2, 1, 8, 4]
16 bubble_sort(random_list_of_nums)
17 print(random_list_of_nums)
```

Sort.py updated with comments on lines 2, 8, and 10

Time to commit again

To commit the new changes (the added comments in this case), do one of the following:

1- git add sort.py

2- git commit -m “added comments”

OR

git commit -a -m “added comments”

```
(base) saeid@saeid-Lenov:~/playground$ git commit -a -m "added comments"
[master 3b8634f] added comments
 1 file changed, 3 insertions(+), 3 deletions(-)
```

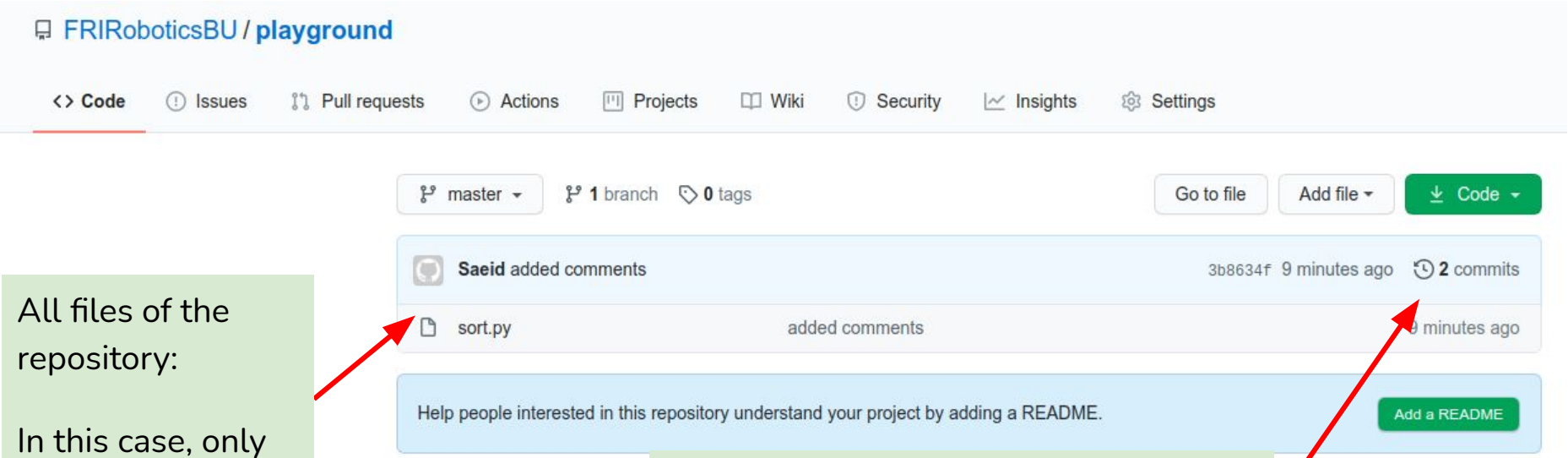
Time to backup your code

We can use *git push* command to back up our code to the remote repository we created in the beginning

```
$ git push origin master
```

```
(base) saeid@saeid-Lenov:~/playground$ git push origin master
Username for 'https://github.com': astrosaeed
Password for 'https://astrosaeed@github.com':
Counting objects: 6, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 672 bytes | 336.00 KiB/s, done.
Total 6 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/FRIRoboticsBU/playground.git
 * [new branch]      master -> master
```

Looking back at the repository

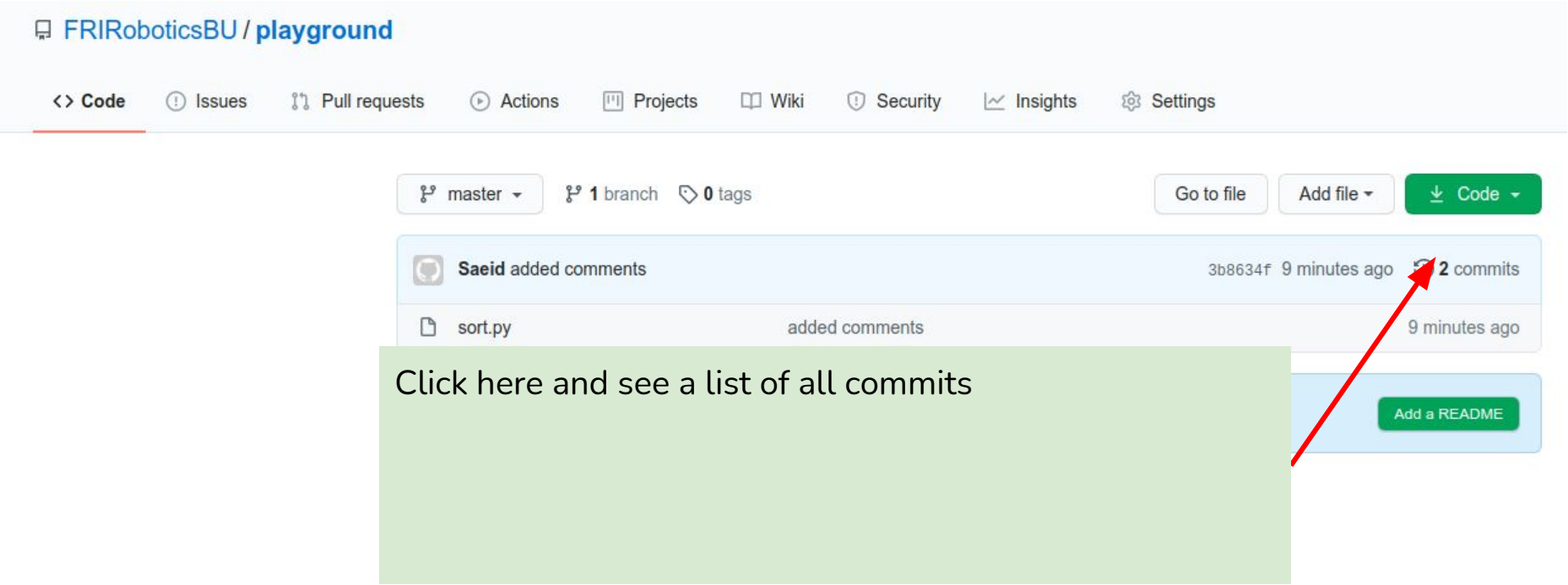


All files of the repository:

In this case, only sort.py

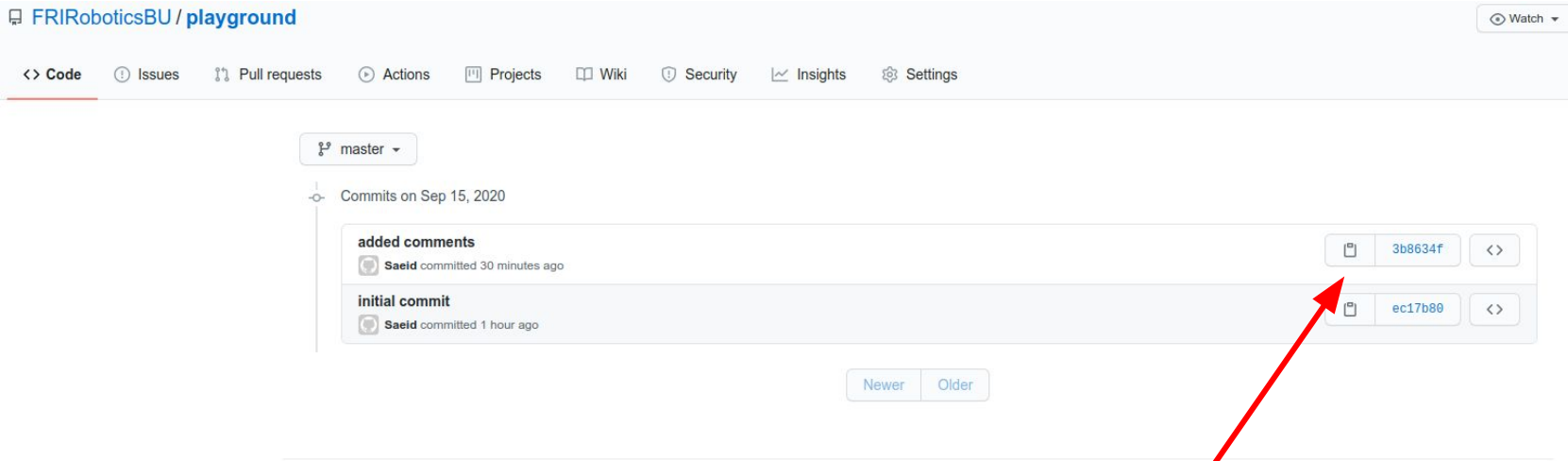
Click here and see a list of all commits

Looking back at the repository



Click here and see a list of all commits

List of all commits



Click on the latest commit and see what has changed.

Changes of the commit

added comments

master

Saeid committed 31 minutes ago

1 parent ec17b80 commit 3b

Commit message we wrote earlier

Showing 1 changed file with 3 additions and 3 deletions.

6 sort.py

```
@@ -1,13 +1,13 @@
 def bubble_sort(nums):
-
     swapped = True
     while swapped:
         swapped = False
         for i in range(len(nums) - 1):
             if nums[i] > nums[i + 1]:
-
                 nums[i], nums[i + 1] = nums[i + 1], nums[i]
-
                 swapped = True
```

```
 def bubble_sort(nums):
+    # We set swapped to True so the loop looks runs at least once
     swapped = True
     while swapped:
         swapped = False
         for i in range(len(nums) - 1):
             if nums[i] > nums[i + 1]:
+                # Swap the elements
                 nums[i], nums[i + 1] = nums[i + 1], nums[i]
+                # Set the flag to True so we'll loop again
                 swapped = True
```

Red lines show what has been deleted

Green lines show what has been added

Going back to older commits

For any reasons, you may wanna access to the older commit. For example, I wanna access to the version of sort.py that did not have any comments. To do that, on your local machine, type: $ git log

```
(base) saeid@saeid-Lenov:~/playground$ git log
commit 3b8634fab282c7d57fb833de964b51453df7a88a (HEAD -> master, origin/master)
Author: Saeid <astrosaeed@gmail.com>
Date:   Tue Sep 15 11:41:38 2020 -0400

    added comments

commit ec17b80f24f61dd331e04091b01453d83c9fbd73
Author: Saeid <astrosaeed@gmail.com>
Date:   Tue Sep 15 11:21:59 2020 -0400

    initial commit
```

We want to go back to this commit

Going back to older commits

```
(base) saeid@saeid-Lenov:~/playground$ git log
commit 3b8634fab282c7d57fb833de964b51453df7a88a (HEAD -> master, origin/master)
Author: Saeid <astrosaeed@gmail.com>
Date:   Tue Sep 15 11:41:38 2020 -0400

    added comments

commit ec17b8bf24f61dd331e04091b01453d83c9fbd73
Author: Saeid <astrosaeed@gmail.com>
Date:   Tue Sep 15 11:21:59 2020 -0400

    initial commit
```

Remember the first 5 characters of this ID

Going back to older commits

Use git checkout command followed by the commit ID:

$ git checkout ec17b

```
(base) saeid@saeid-Lenov:~/playground$ git checkout ec17b
Note: checking out 'ec17b'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b <new-branch-name>

HEAD is now at ec17b80 initial commit
```

Git pull

Always do **$git pull** first, to get the most recent updates from your collaborators

If you worry about your implementation not

Summary

As long as you know:

How to create a repository,

How to access older version of your code,

How to add, commit, pull, and push code

Then that would be great

Some additional info

Working in teams

Some notes about working in teams:

1- Your collaborators might have already pushed some updates to the repository, to make sure that you get the most recent version of the project, type

```
$ git pull origin master
```

2- To avoid conflicts when more than one member is working on the same file, each person can create their own branch, and at some point (if needed) merge that branch to master branch (more info on branches, will be added soon).

Adding branches

```
$ git branch branch_name

$ git checkout branch_name
```

Going to the most recent commit

```
$ git checkout master
```

Optional practice

1- Follow the steps below:

2.1. start cloning the repository,

2.2. add a new file that prints “hello world” in any language that you like

2.3. Add and commit your code

2.4. Push it to the github repository

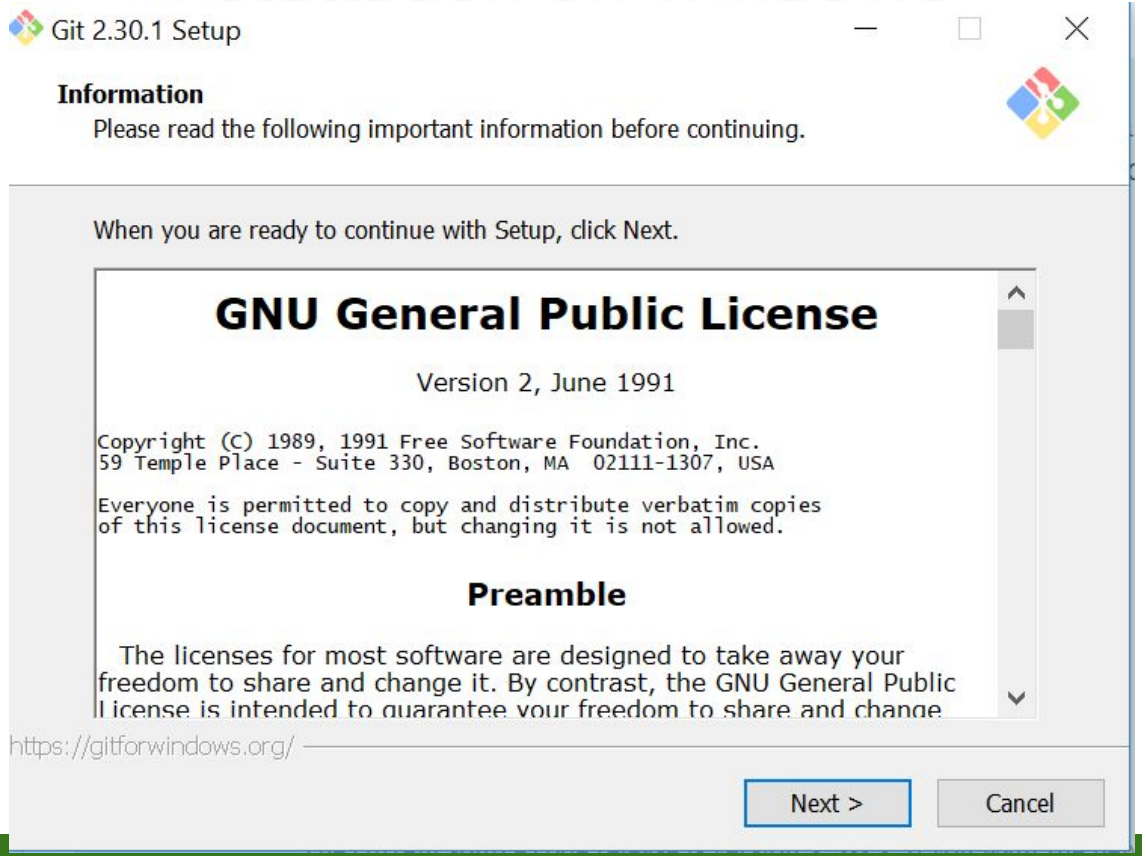
2.5. Back to your code, print another line saying “I learned basic git commands”

2.6. Repeat steps 2.3. and 2.4.

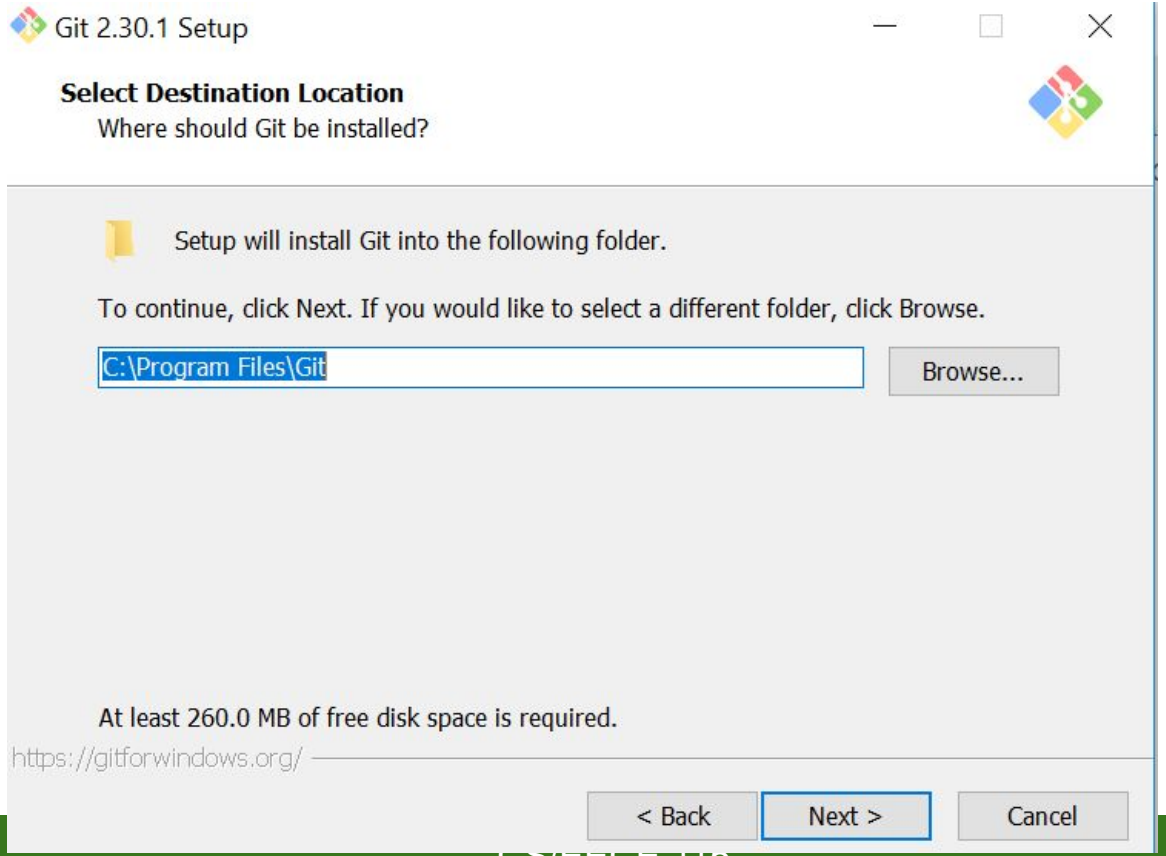
Installation on windows

https://git-scm.com/download/win

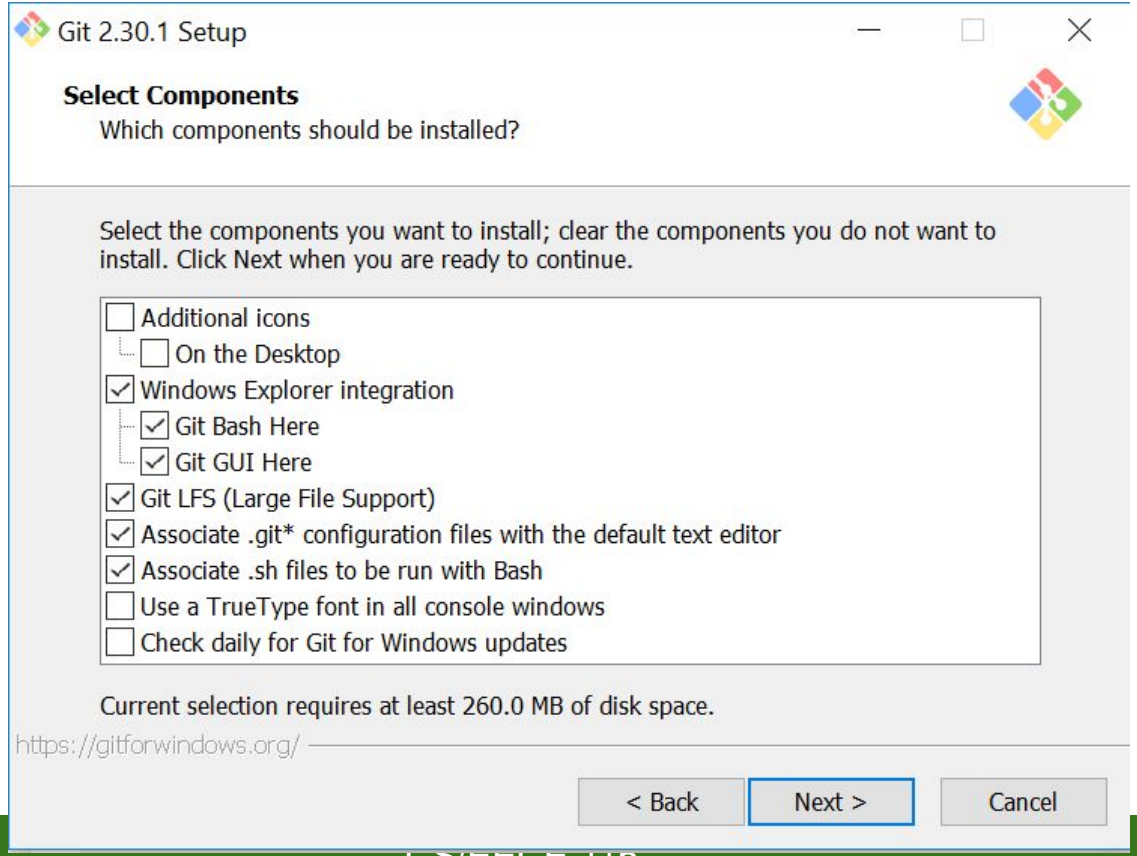
Installation on windows



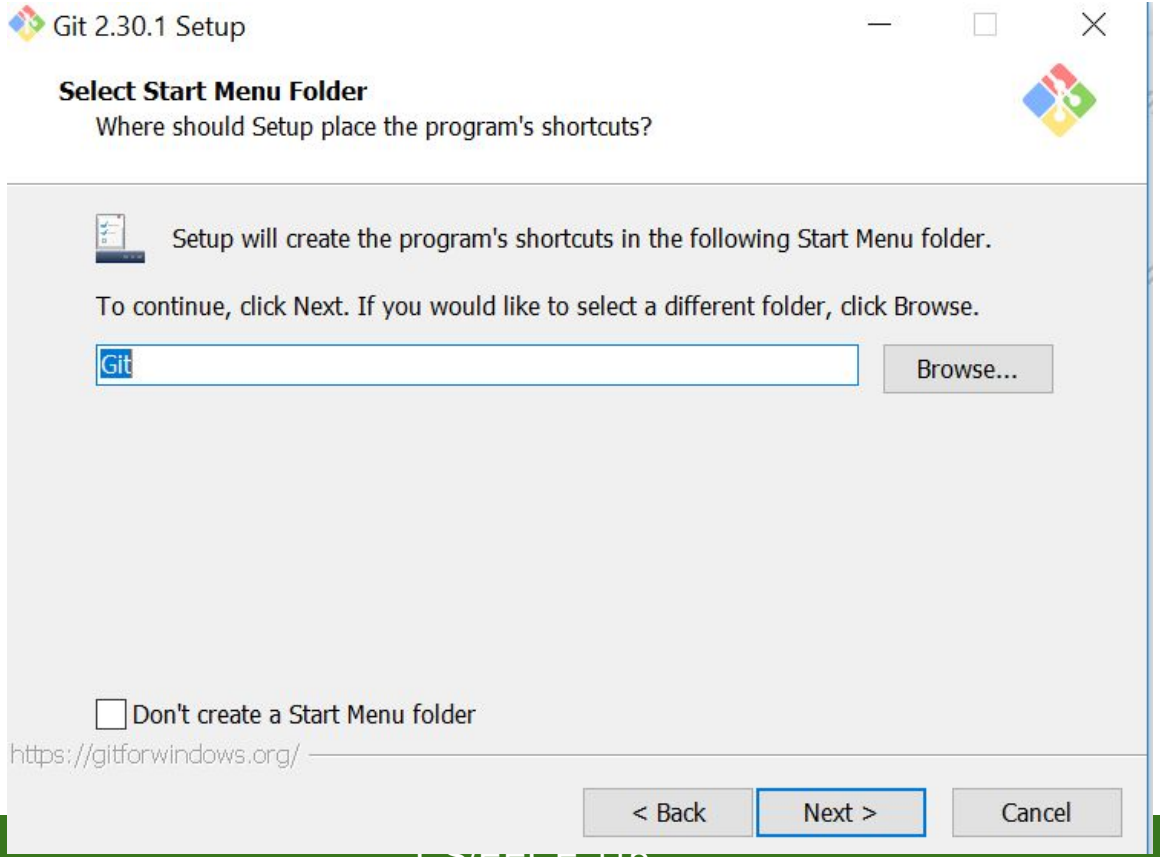
Installation on windows



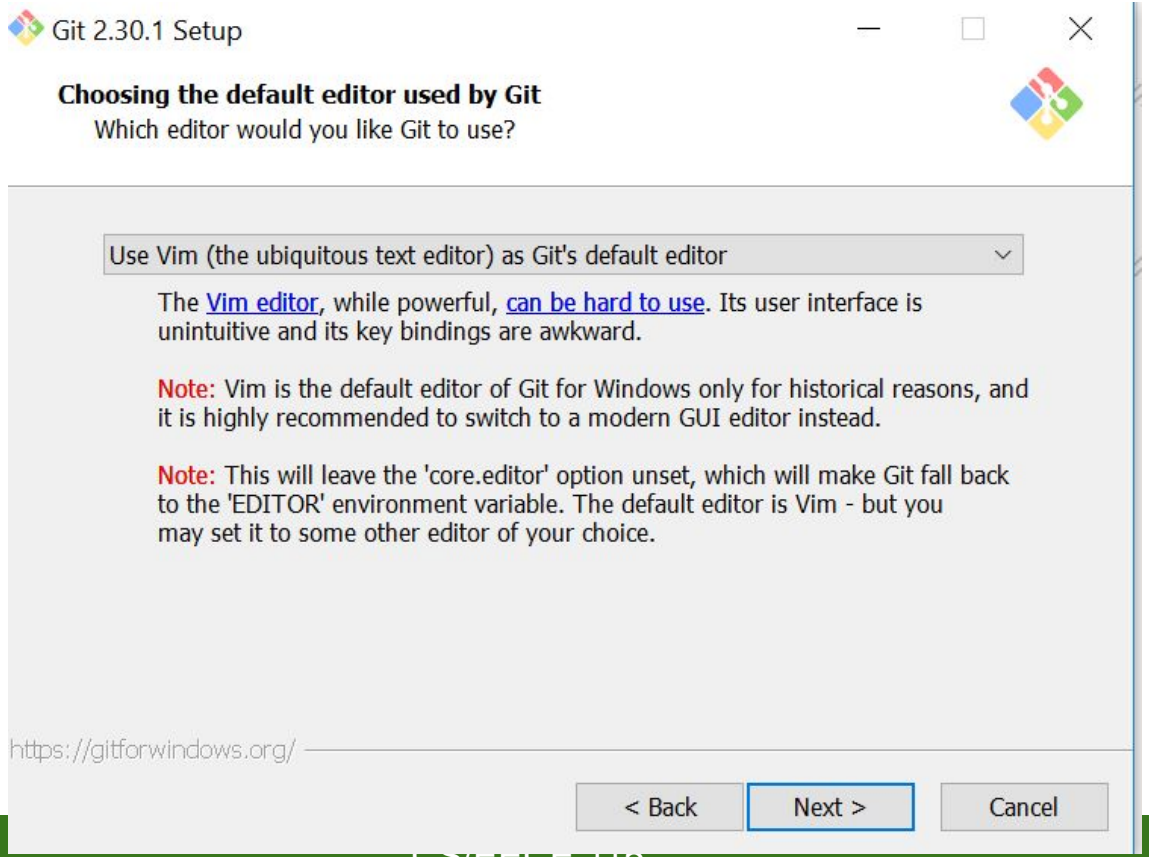
Installation on windows



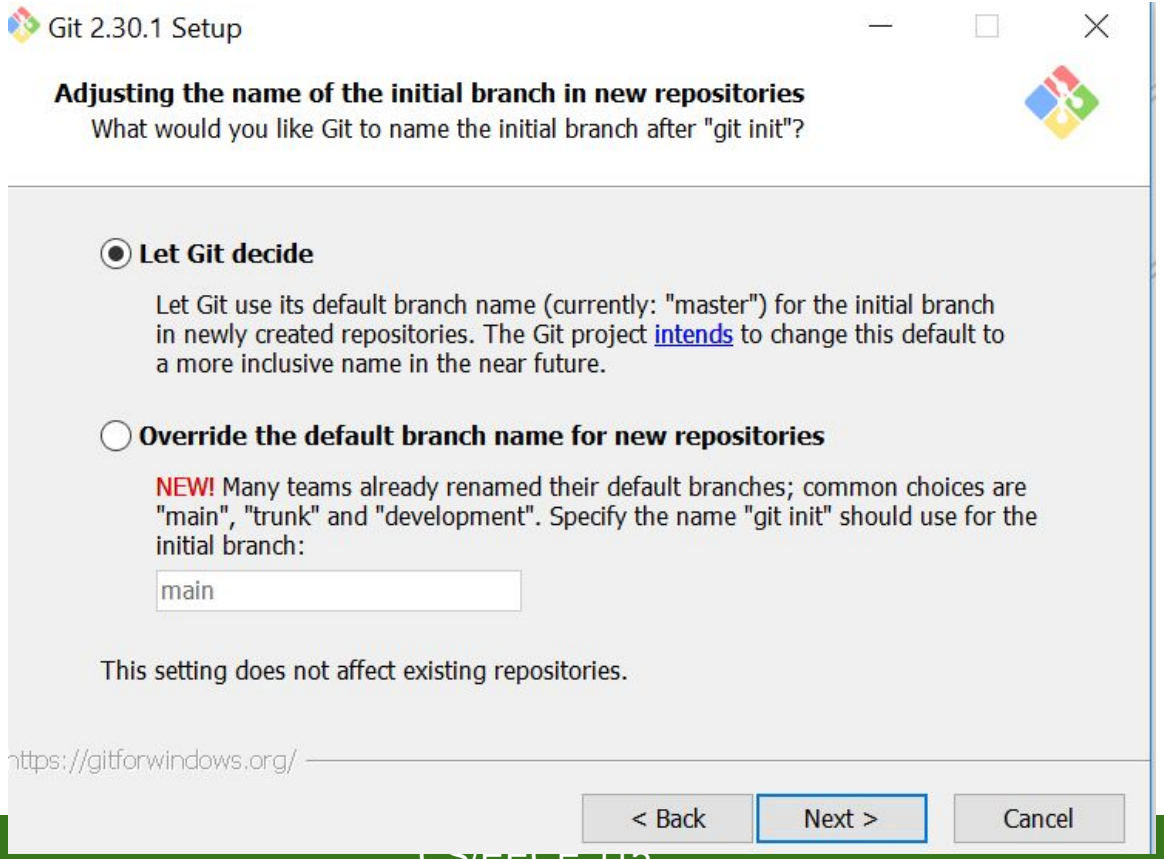
Installation on windows



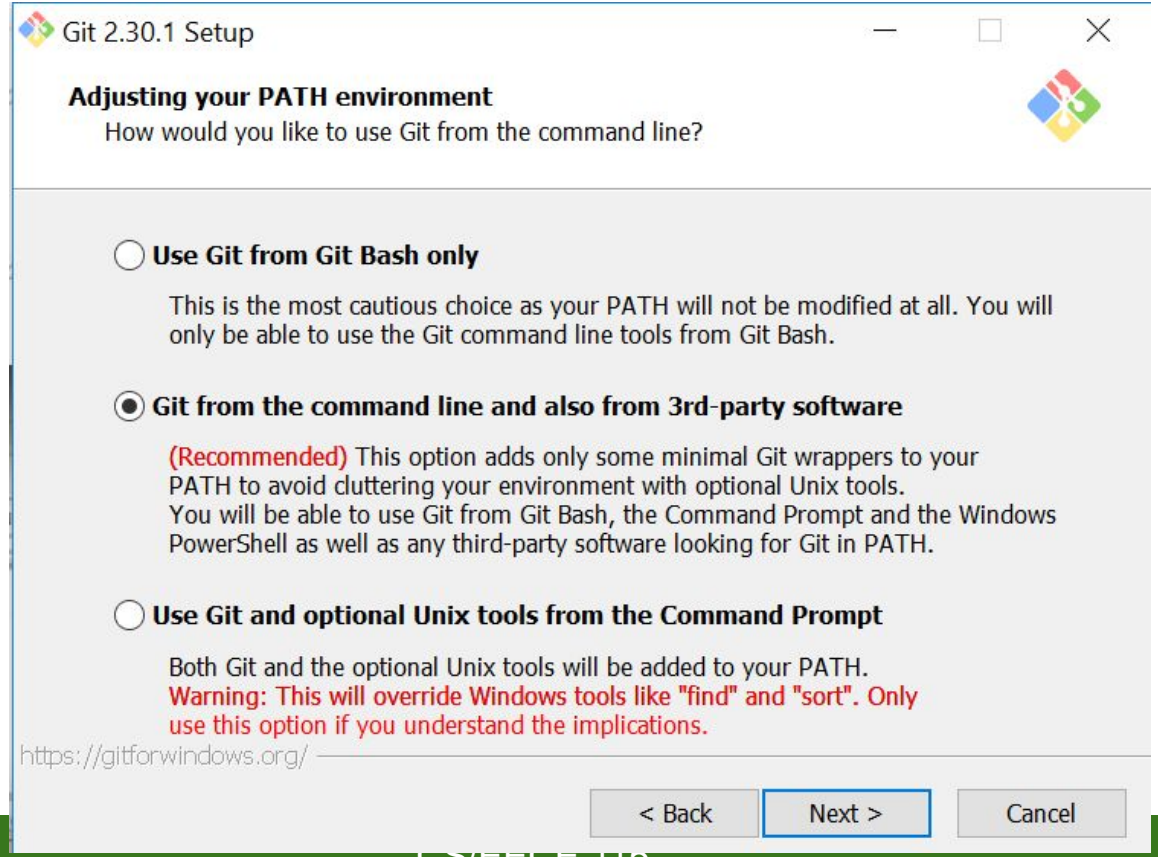
Installation on windows



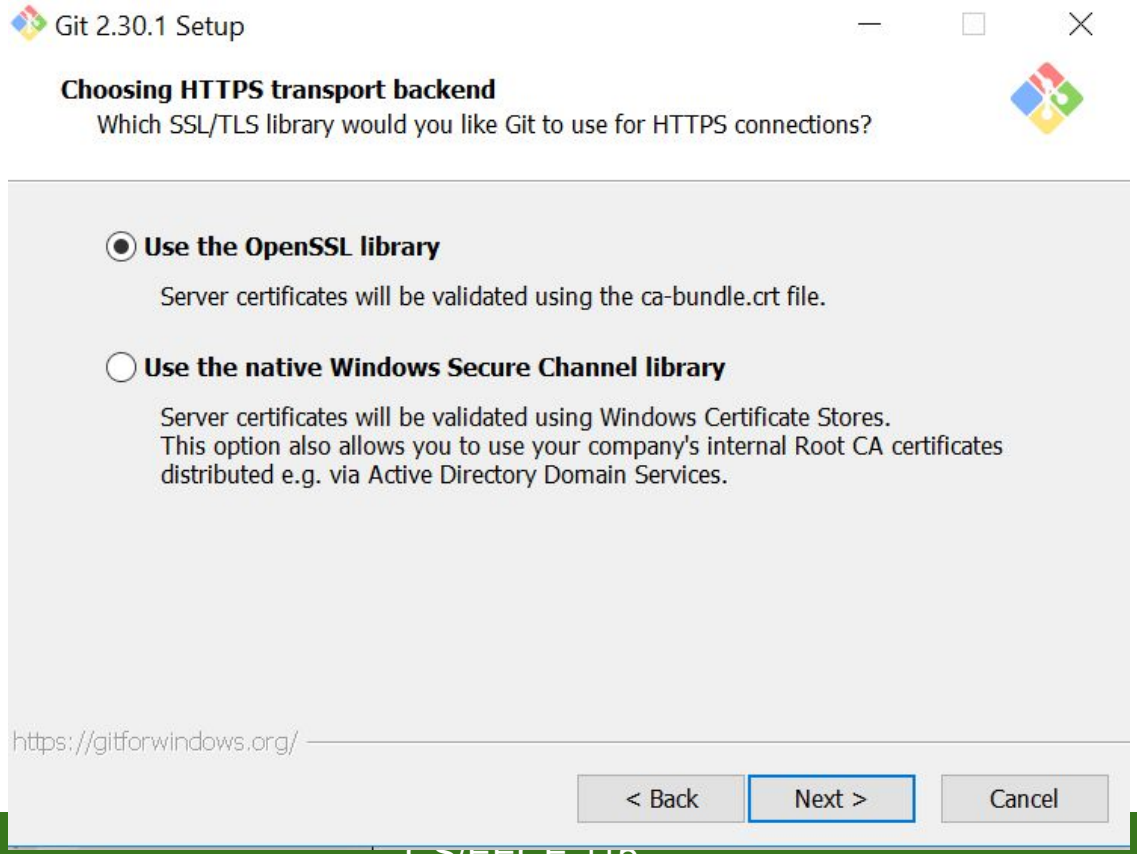
Installation on windows



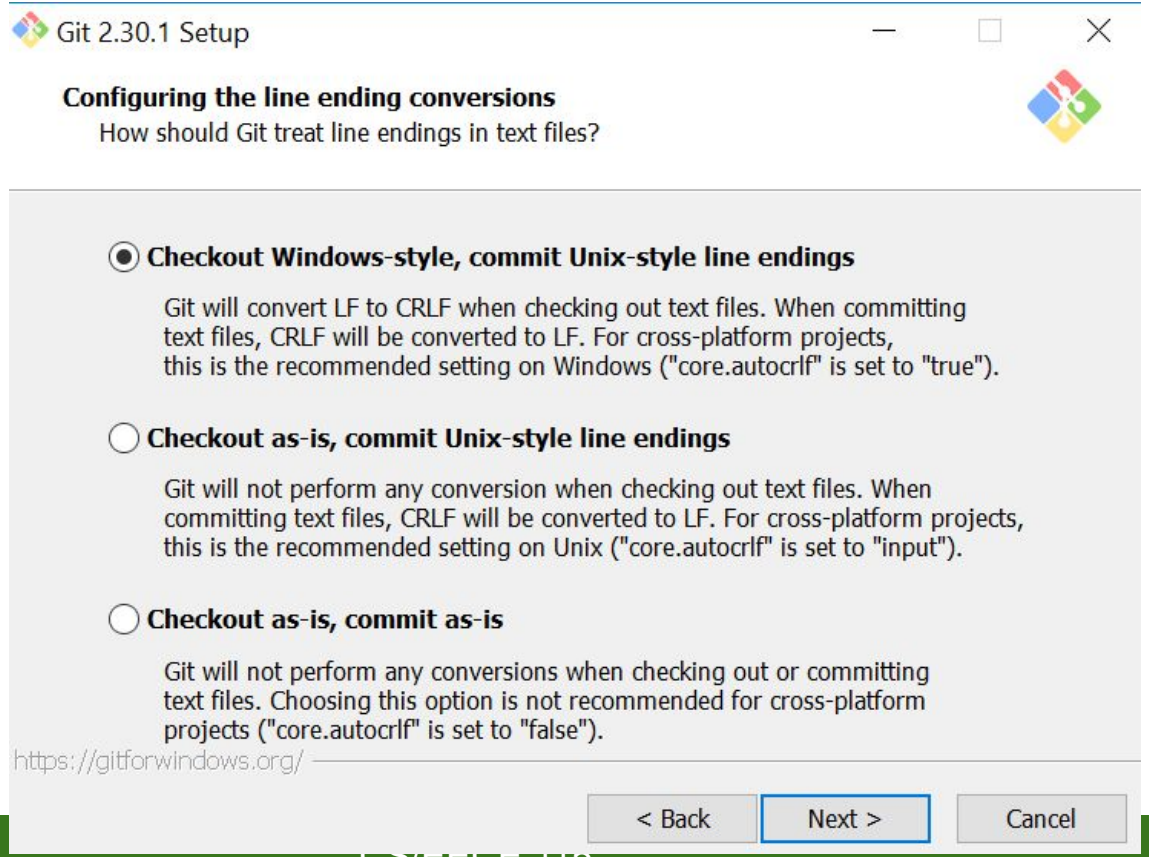
Installation on windows



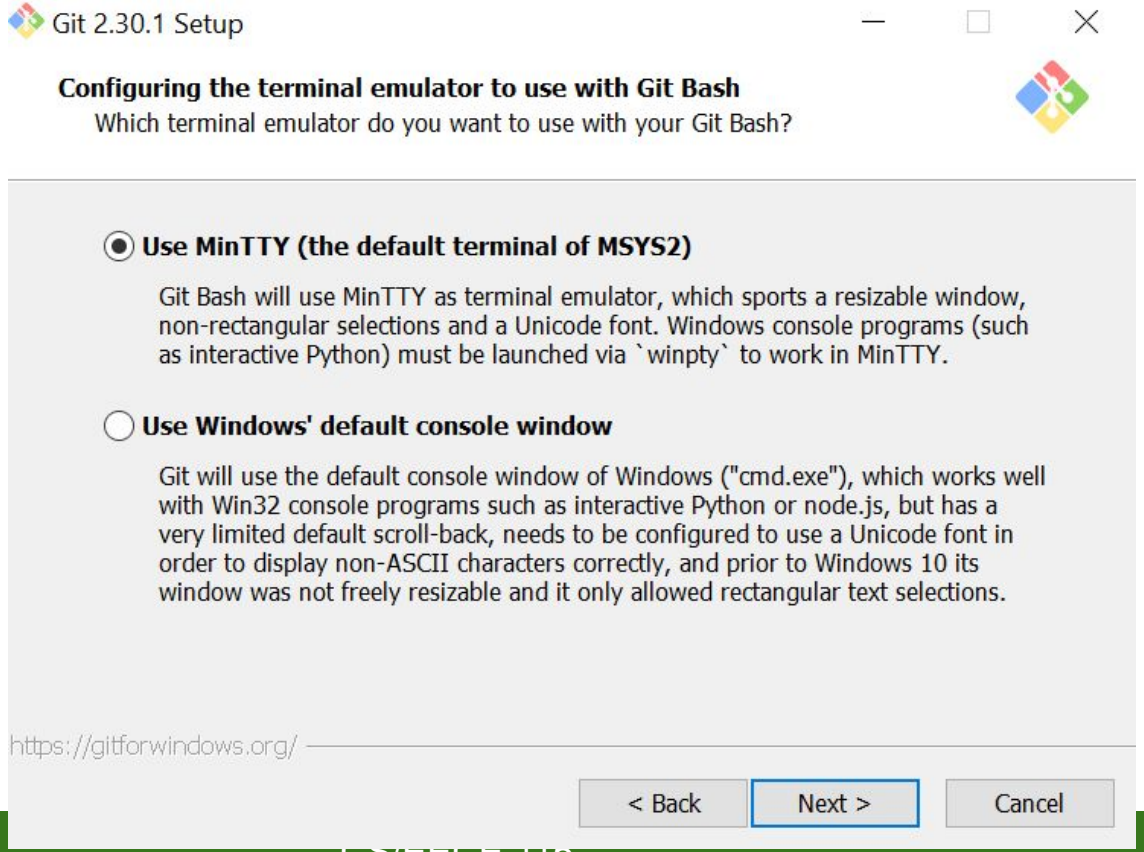
Installation on windows



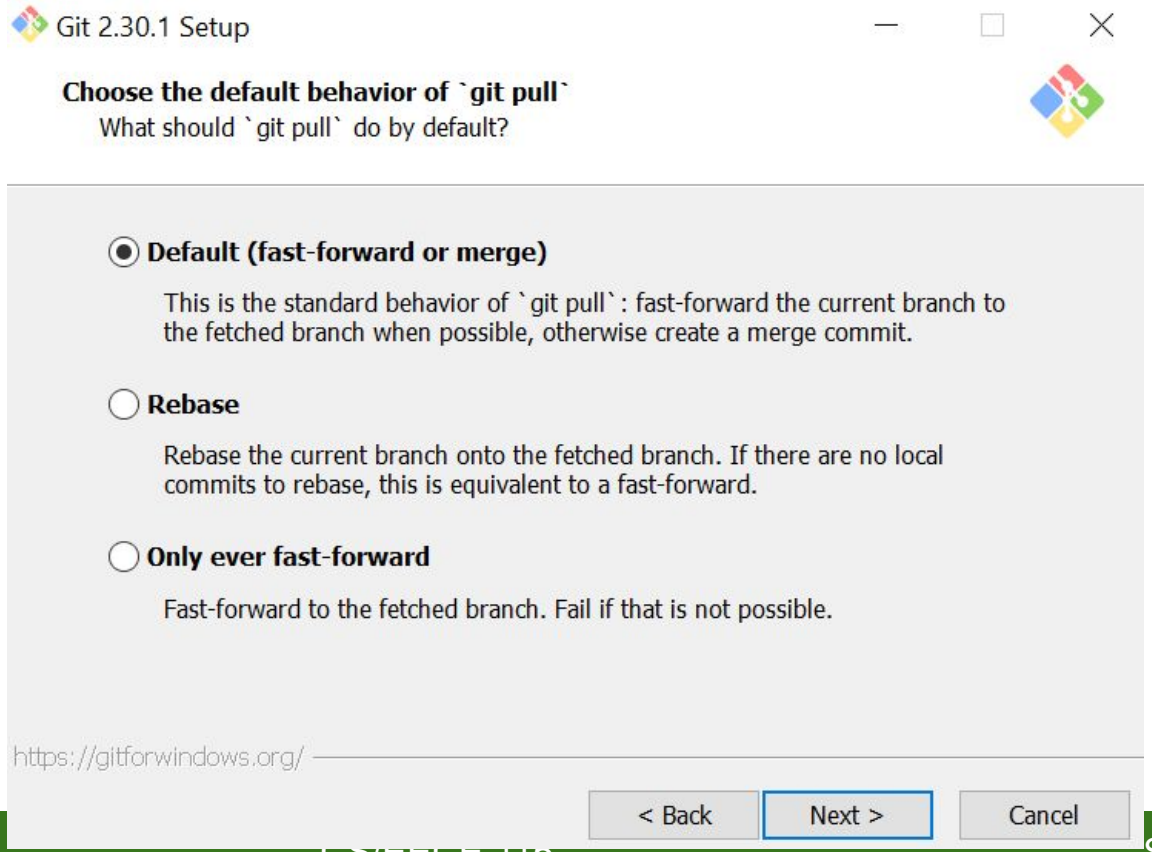
Installation on windows



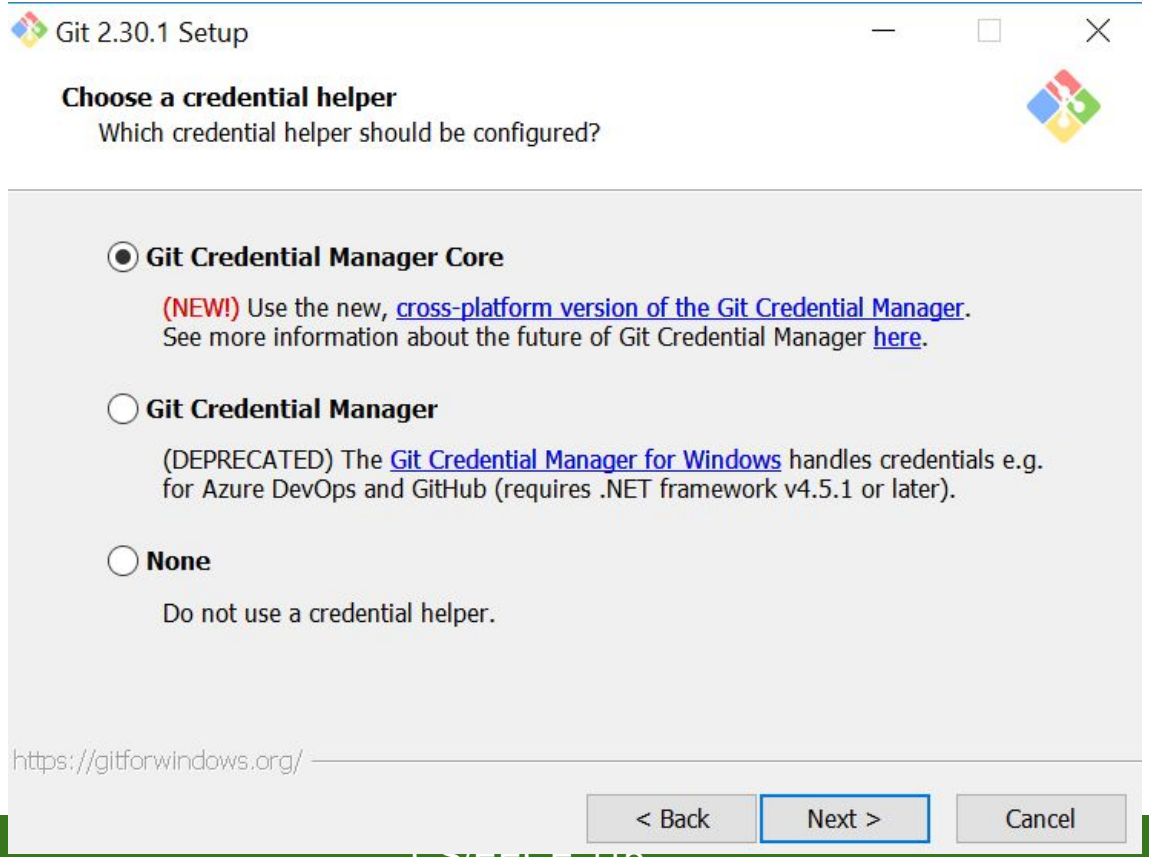
Installation on windows



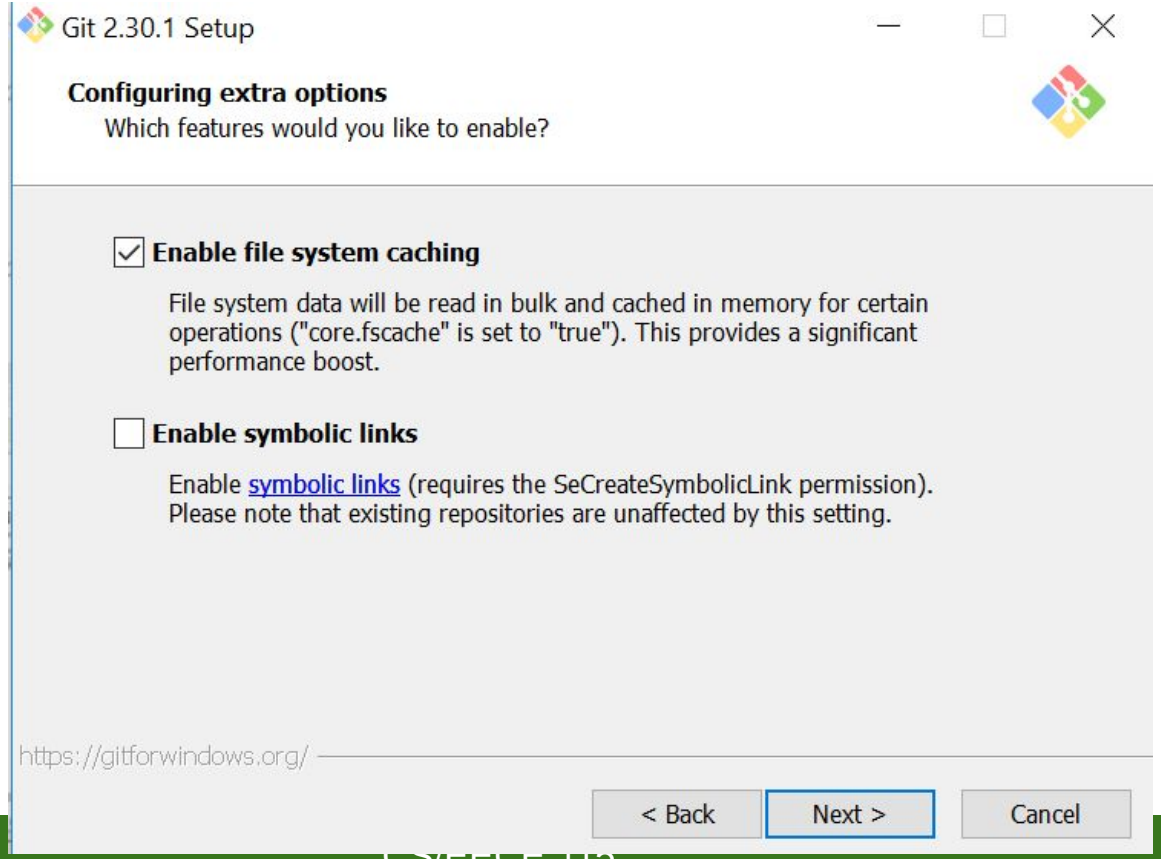
Installation on windows



Installation on windows



Installation on windows

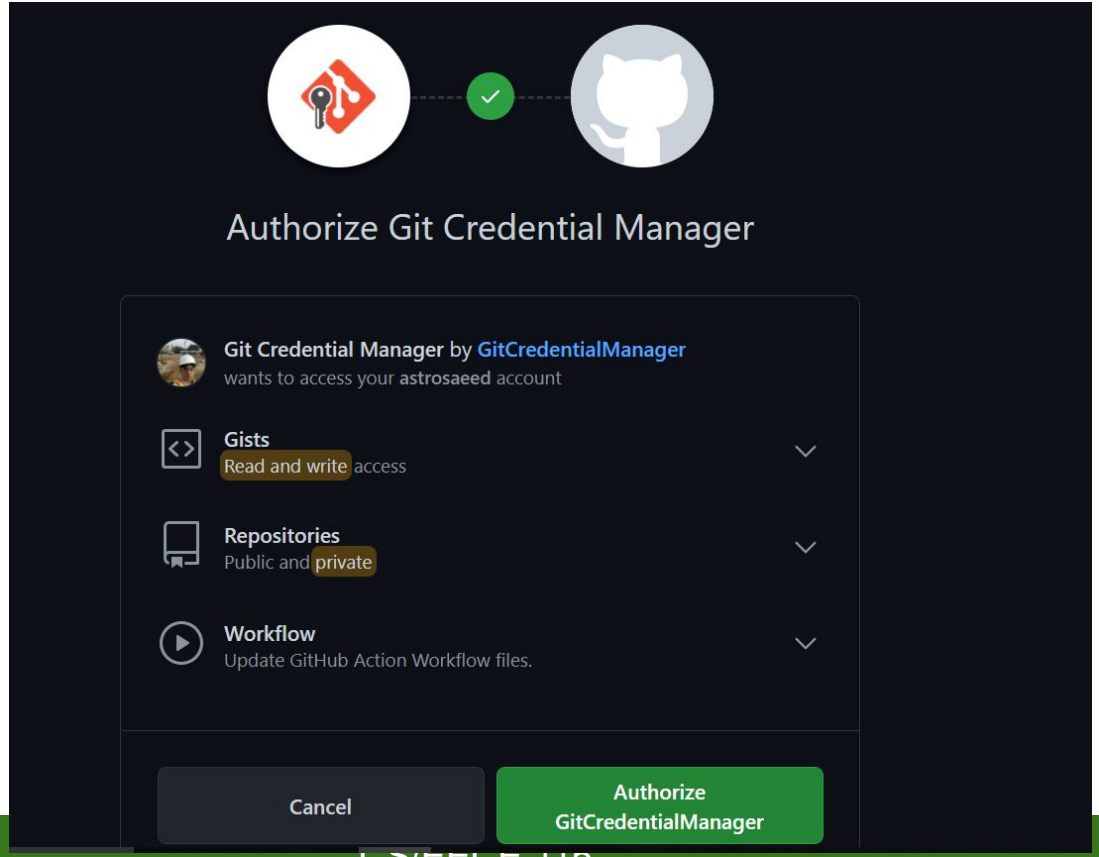


Installation on windows

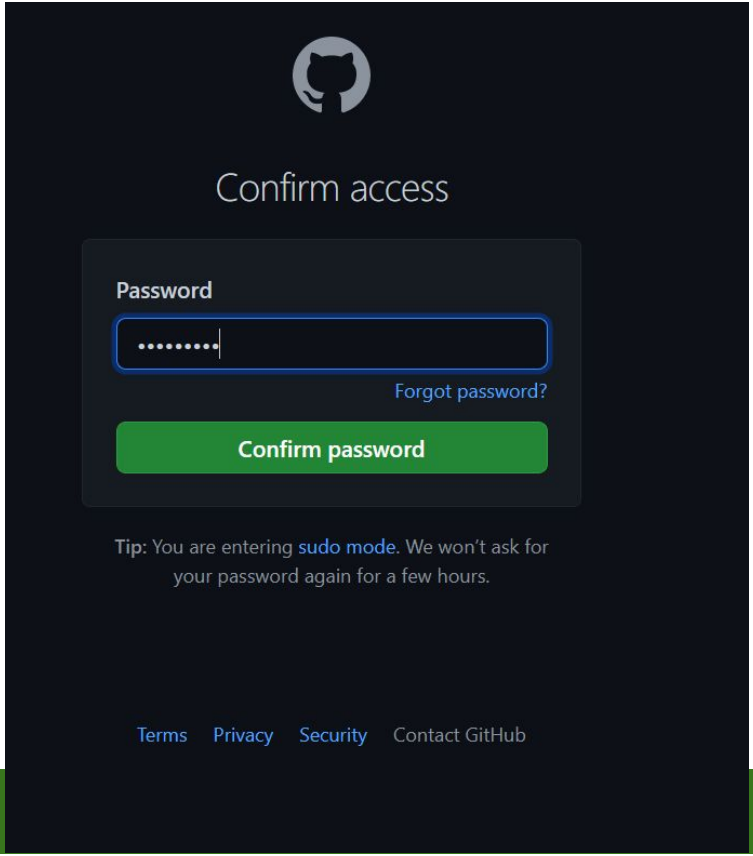
```
*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'turtlebot@LAPTOP-G31DO3MG.(none)')

(friteaching) C:\Users\turtlebot\teaching\week3>
```

Installation on windows

Installation on windows



Installation on windows



Github Desktop

https://desktop.github.com/

https://guides.github.com/introduction/git-handbook/

ReadMe

So

Some useful git commands

Git clone

Git status

Git add

Git commit

Git push

Git pull

There are more commands, but you will learn it as-needed by googling/stack-overflow