

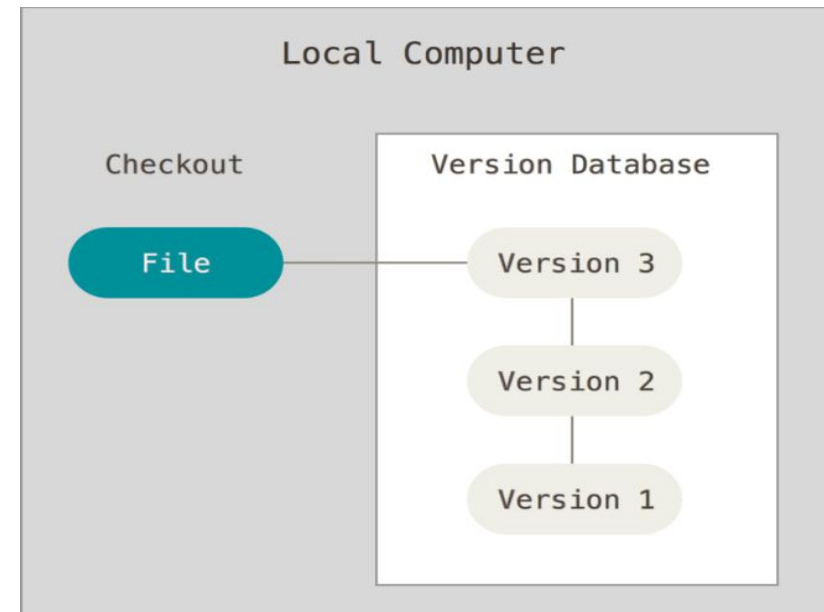
GIT, Version Control System, Branch, GIT Hub

Version Control System

- Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.
- If you are a graphic or web designer and want to keep every version of an image or layout (which you would most certainly want to), a Version Control System (VCS) is a very wise thing to use.
- It allows you to revert selected files back to a previous state, revert the entire project back to a previous state, compare changes over time, see who last modified something that might be causing a problem, who introduced an issue and when, and more. Using a VCS also generally means that if you screw things up or lose files, you can easily recover.

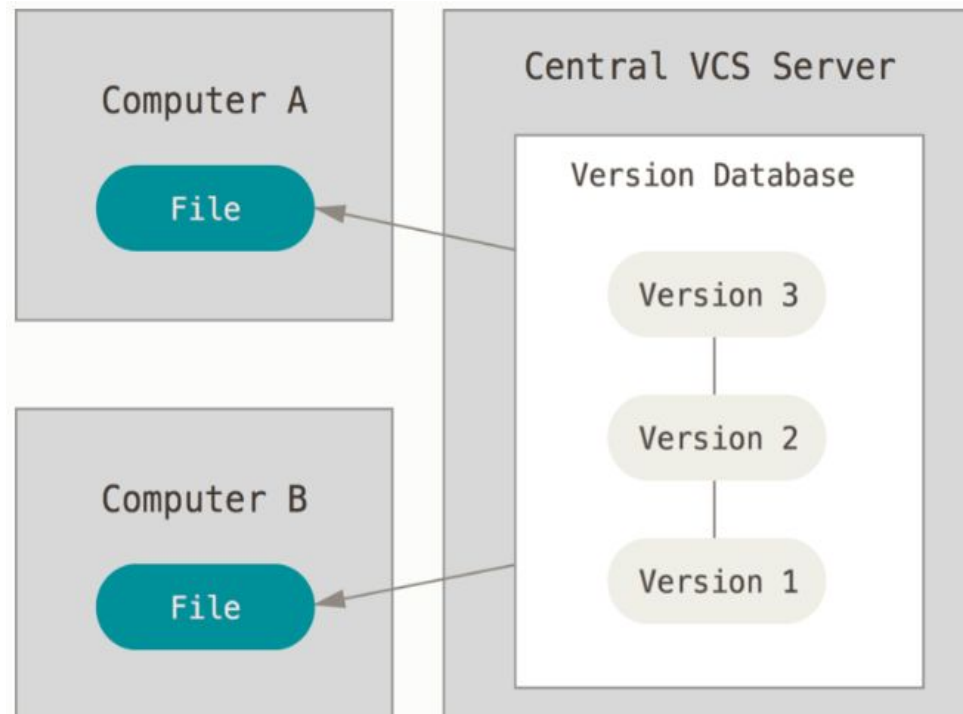
Local Version Control Systems

- Many people's version-control method of choice is to copy files into another directory (perhaps a time-stamped directory, if they're clever). This approach is very common because it is so simple, but it is also incredibly error prone.



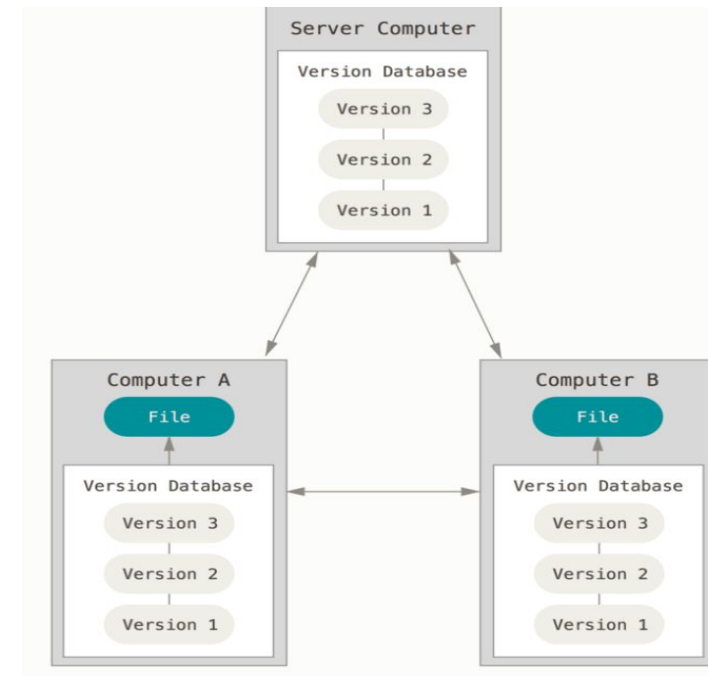
Centralized Version Control Systems

- These systems (such as CVS, Subversion, and Perforce) have a single server that contains all the versioned files, and a number of clients that check out files from that central place.



Distributed Version Control Systems

- In a DVCS (such as Git, Mercurial, Bazaar or Darcs), clients don't just check out the latest snapshot of the files; rather, they fully mirror the repository, including its full history.



- Git is a popular version control system. It was created by Linus Torvalds in 2005, and has been maintained by Junio Hamano since then.

It is used for:

- Tracking code changes
- Tracking who made changes
- Coding collaboration

What does Git do?

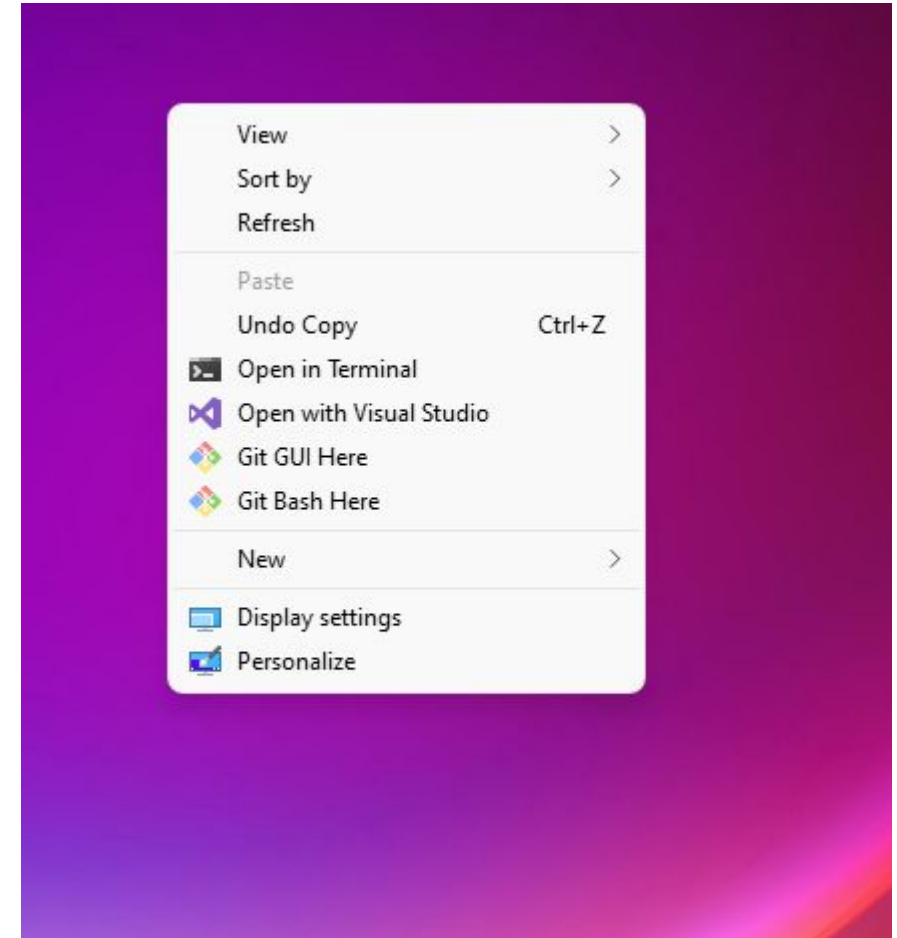
- Manage projects with **Repositories**
- **Clone** a project to work on a local copy
- Control and track changes with **Staging** and **Committing**
- **Branch** and **Merge** to allow for work on different parts and versions of a project
- **Pull** the latest version of the project to a local copy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

GIT Installation

- <https://git-scm.com/>





MINGW64:/c/Users/student/Desktop



student@DESKTOP-NVCKE78 MINGW64 ~

\$ cd Desktop

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop

\$ pwd

/c/Users/student/Desktop

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop

\$ git config --global user.name "nidhi"

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop

\$ git config --global user.mail "nidhi.2592@gmail.com"

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop

\$

```
student@DESKTOP-NVCKE78 MINGW64 ~/Desktop
```

```
$ pwd  
/c/Users/student/Desktop
```

```
student@DESKTOP-NVCKE78 MINGW64 ~/Desktop
```

```
$ git config --global user.name "nidhi"
```

```
student@DESKTOP-NVCKE78 MINGW64 ~/Desktop
```

```
$ git config --global user.mail "nidhi.2592@gmail.com"
```

```
student@DESKTOP-NVCKE78 MINGW64 ~/Desktop
```

```
$ git config --list  
diff.astextplain.textconv=astextplain  
filter.lfs.clean=git-lfs clean -- %f  
filter.lfs.smudge=git-lfs smudge -- %f  
filter.lfs.process=git-lfs filter-process  
filter.lfs.required=true  
http.sslbackend=openssl  
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt  
core.autocrlf=true  
core.fscache=true  
core.symlinks=false  
core.fsmonitor=true  
pull.rebase=false  
credential.helper=manager  
credential.https://dev.azure.com.usehttppath=true  
init.defaultbranch=master  
user.name=nidhi  
user.mail=nidhi.2592@gmail.com
```

```
student@DESKTOP-NVCKE78 MINGW64 ~/Desktop
```

```
$ |
```

Other

-z, --null	terminate values with NUL byte
--name-only	show variable names only
--includes	respect include directives on lookup
--show-origin	show origin of config (file, standard input, blob, command line)
--show-scope	show scope of config (worktree, local, global, system, command)
--default <value>	with --get, use default value when missing entry

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop

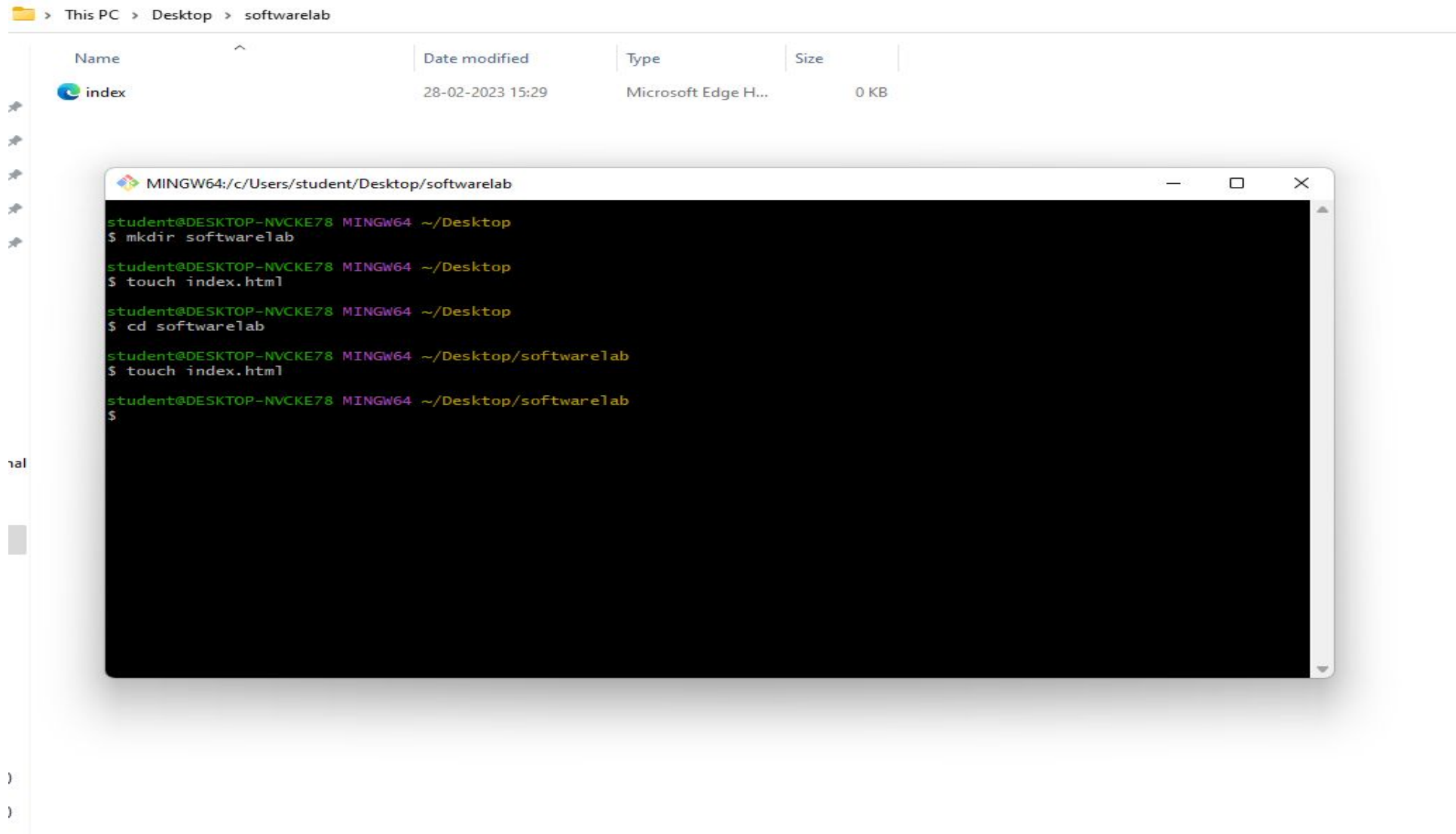
\$ git config -l

```
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
core.fsmonitor=true
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.name=nidhi
user.mail=nidhi.2592@gmail.com
```


student@DESKTOP-NVCKE78 MINGW64 ~/Desktop

\$ |

Git in Action (git - add, status, commit, checkout)



This PC > Desktop > softwarelab

Name	Date modified	Type	Size
 index	28-02-2023 15:31	Microsoft Edge H...	1 KB

index - Notepad

File Edit View

```
<!DOCTYPE html>
<html>
<head>
<title>Hello World!</title>
</head>
<body>

<h1>Hello world!</h1>
<p>This is the first file in my new Git Repo.</p>

</body>
</html>
```

Ln 1, Col 1

100%

Windows (CRLF)

UTF-8

we check the Git status and see if it is a part of our repo:

```
MINGW64:/c/Users/student/Desktop/softwarelab

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop
$ touch index.html

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop
$ cd softwarelab

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab
$ touch index.html

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab
$ git init
Initialized empty Git repository in C:/Users/student/Desktop/softwarelab/.git/

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (master)
$ ls index.html
index.html

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (master)
$ git status
On branch master

No commits yet

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

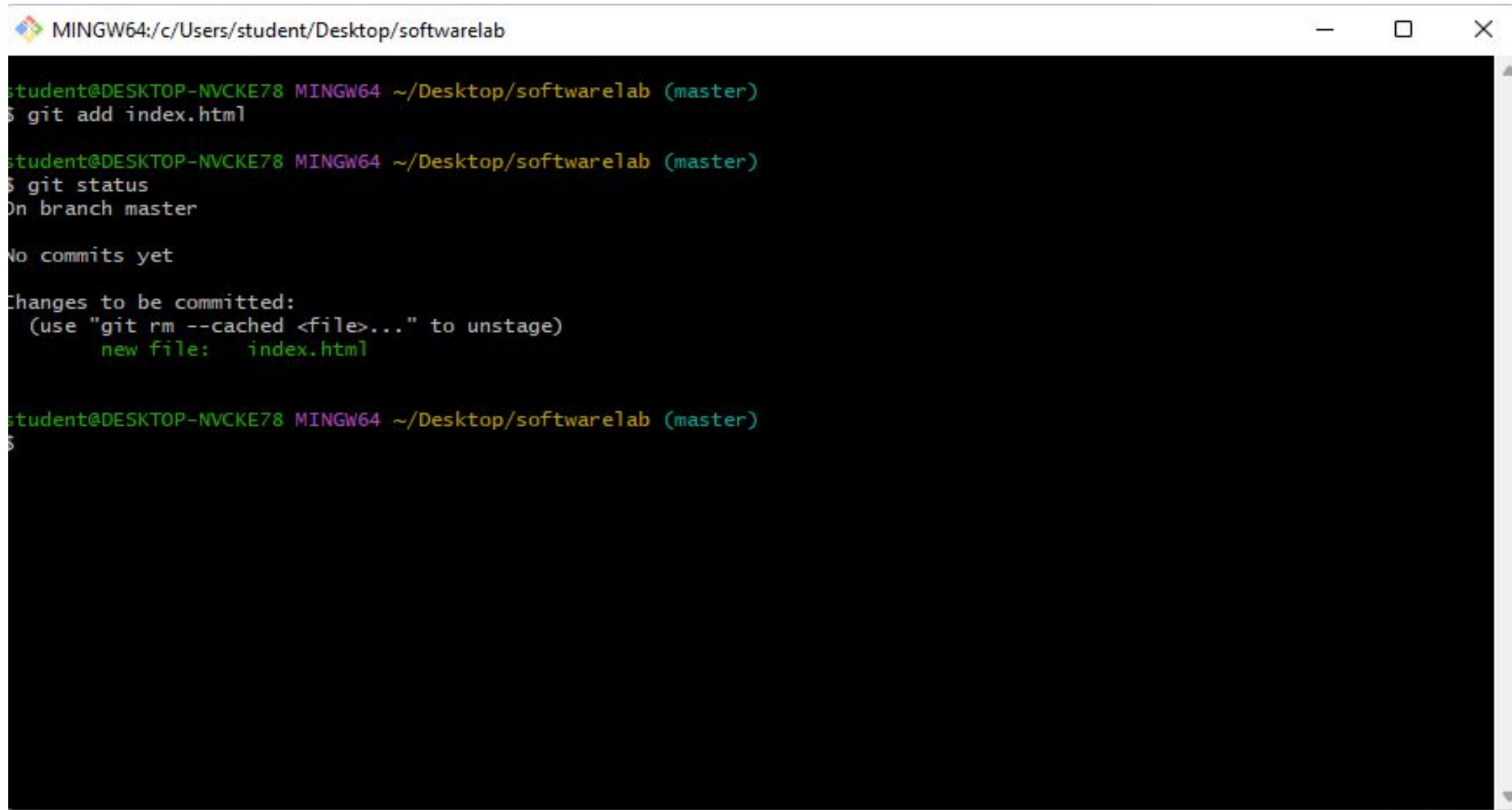
nothing added to commit but untracked files present (use "git add" to track)

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (master)
$
```

Git Staging Environment

- One of the core functions of Git is the concepts of the Staging Environment, and the Commit.
- As you are working, you may be adding, editing and removing files. But whenever you hit a milestone or finish a part of the work, you should add the files to a Staging Environment.
- **Staged** files are files that are ready to be **committed** to the repository you are working on.

- `git add index.html`
- `git status`



A screenshot of a Windows command prompt window titled "MINGW64:/c/Users/student/Desktop/softwarelab". The window shows the execution of two git commands. The first command is "git add index.html", which is followed by the prompt "student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (master)". The second command is "git status", which produces the following output: "On branch master", "No commits yet", and "Changes to be committed: (use 'git rm --cached <file>...' to unstage) new file: index.html". The prompt then returns to "student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (master)".

```
MINGW64:/c/Users/student/Desktop/softwarelab

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (master)
$ git add index.html

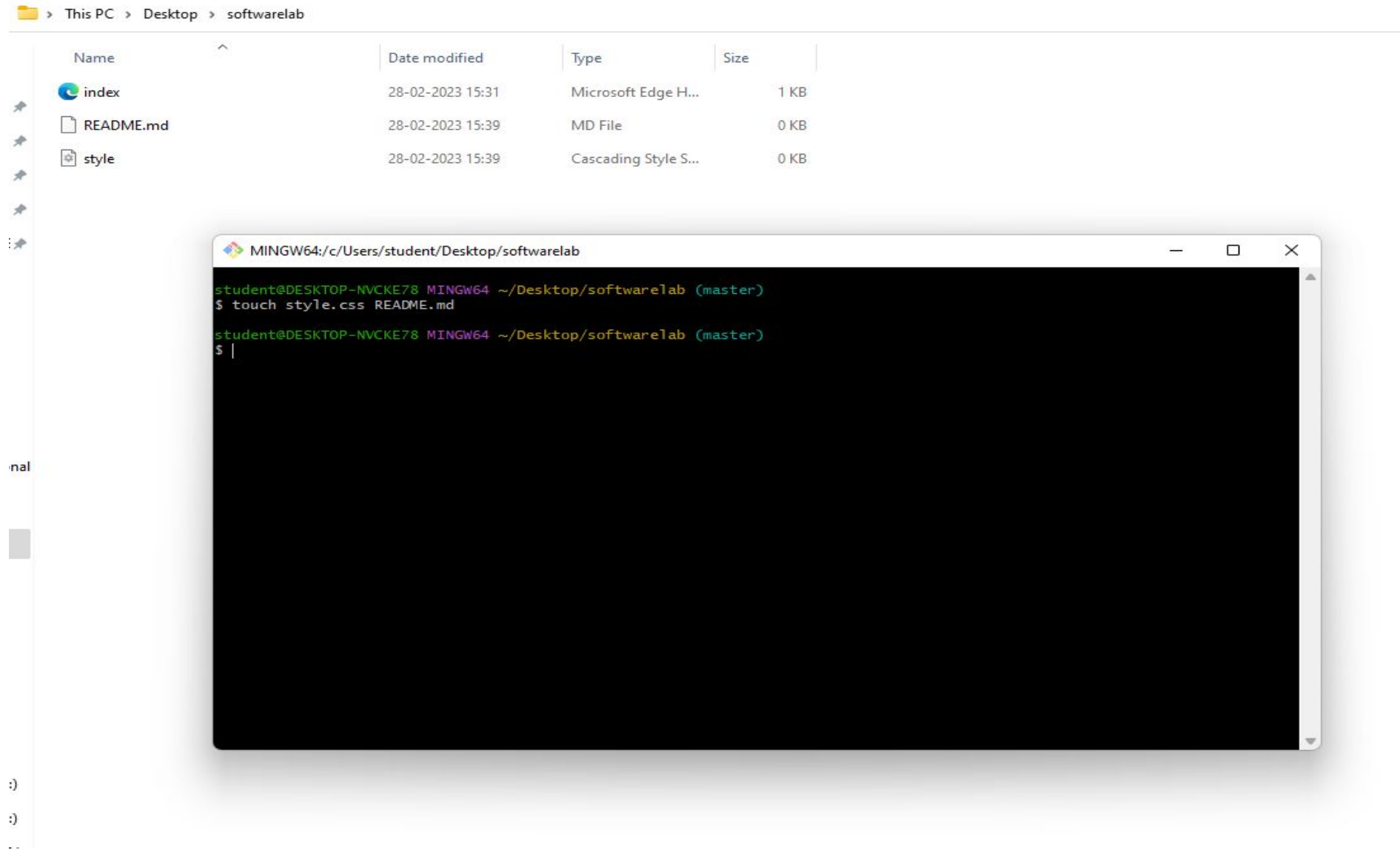
student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (master)
$ git status
On branch master

No commits yet

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

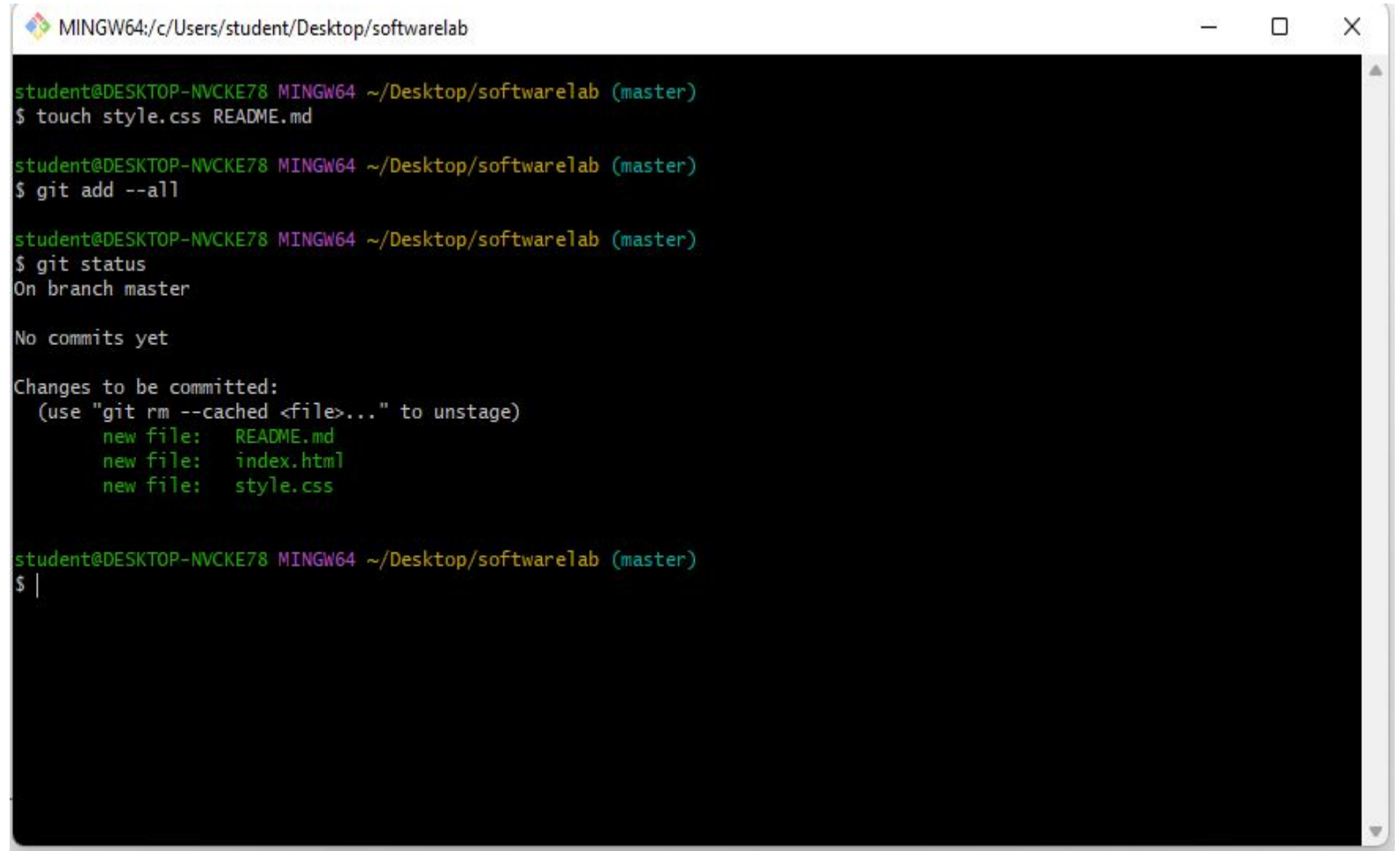
student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (master)
$
```

Git Add More than One File



Now add all files in the current directory to the Staging Environment:

- `git add -all`
- `git status`

A terminal window titled 'MINGW64:/c/Users/student/Desktop/softwarelab' with standard window controls. The terminal shows a sequence of commands and their outputs. First, 'touch style.css README.md' is executed. Then, 'git add --all' is run. Finally, 'git status' is executed, showing the current branch as 'master' with no commits yet, and listing three new files to be committed: README.md, index.html, and style.css. The prompt '\$ |' is visible at the bottom.

```
MINGW64:/c/Users/student/Desktop/softwarelab

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (master)
$ touch style.css README.md

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (master)
$ git add --all

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README.md
        new file:   index.html
        new file:   style.css

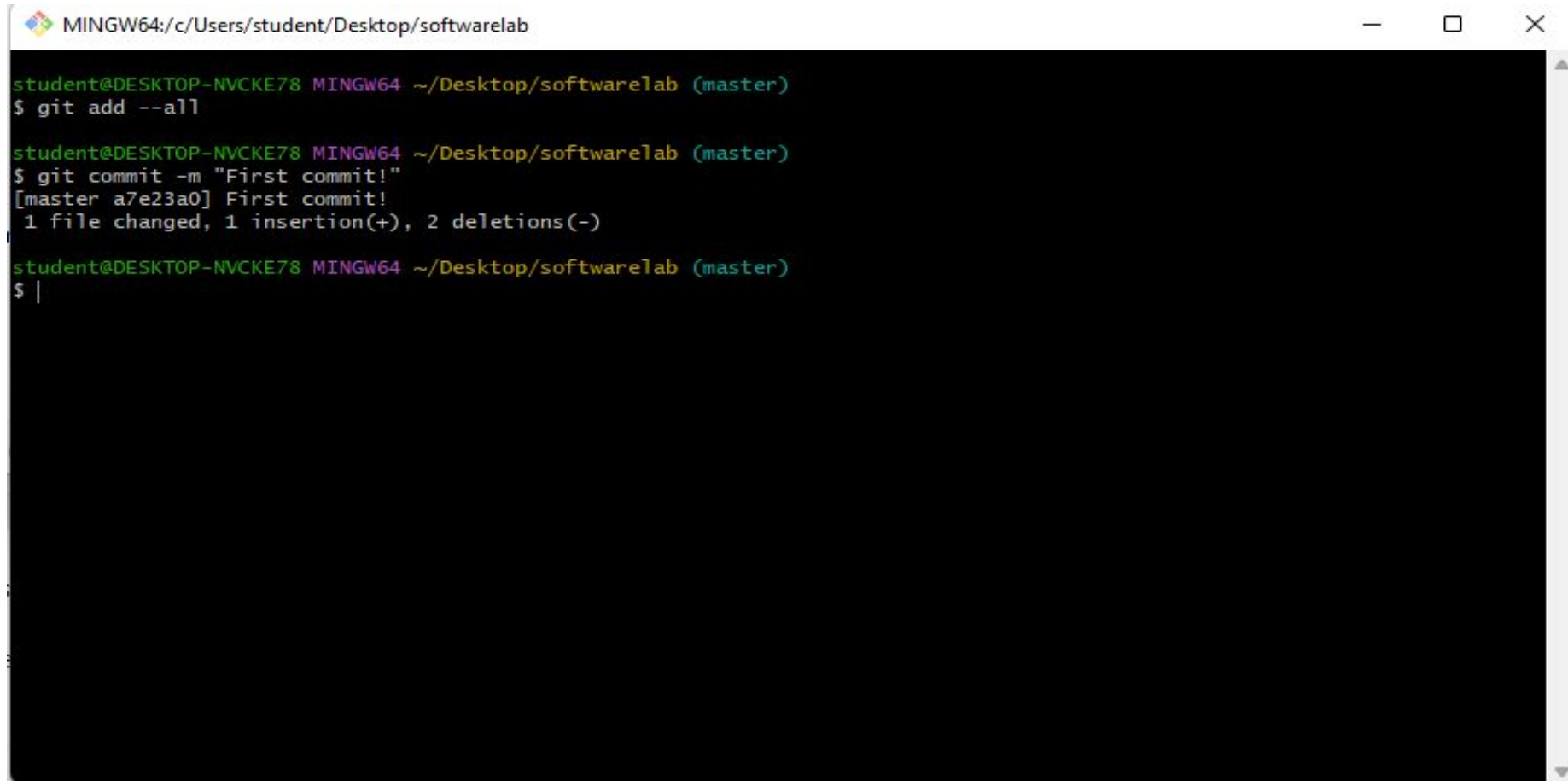
student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (master)
$ |
```

Git Commit

- Since we have finished our work, we are ready move from stage to commit for our repo.
- Adding commits keep track of our progress and changes as we work. Git considers each commit change point or "save point". It is a point in the project you can go back to if you find a bug, or want to make a change.
- When we commit, we should always include a message.
- By adding clear messages to each commit, it is easy for yourself (and others) to see what has changed and when.

Git Commit

- `git commit -m "First commit!"`

A screenshot of a Windows terminal window titled 'MINGW64:/c/Users/student/Desktop/softwarelab'. The terminal shows a sequence of Git commands and their output. The user is in the 'master' branch. They run 'git add --all', then 'git commit -m "First commit!"'. The output shows the commit hash 'a7e23a0' and a summary of changes: '1 file changed, 1 insertion(+), 2 deletions(-)'. The prompt returns to the shell.

```
MINGW64:/c/Users/student/Desktop/softwarelab

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (master)
$ git add --all

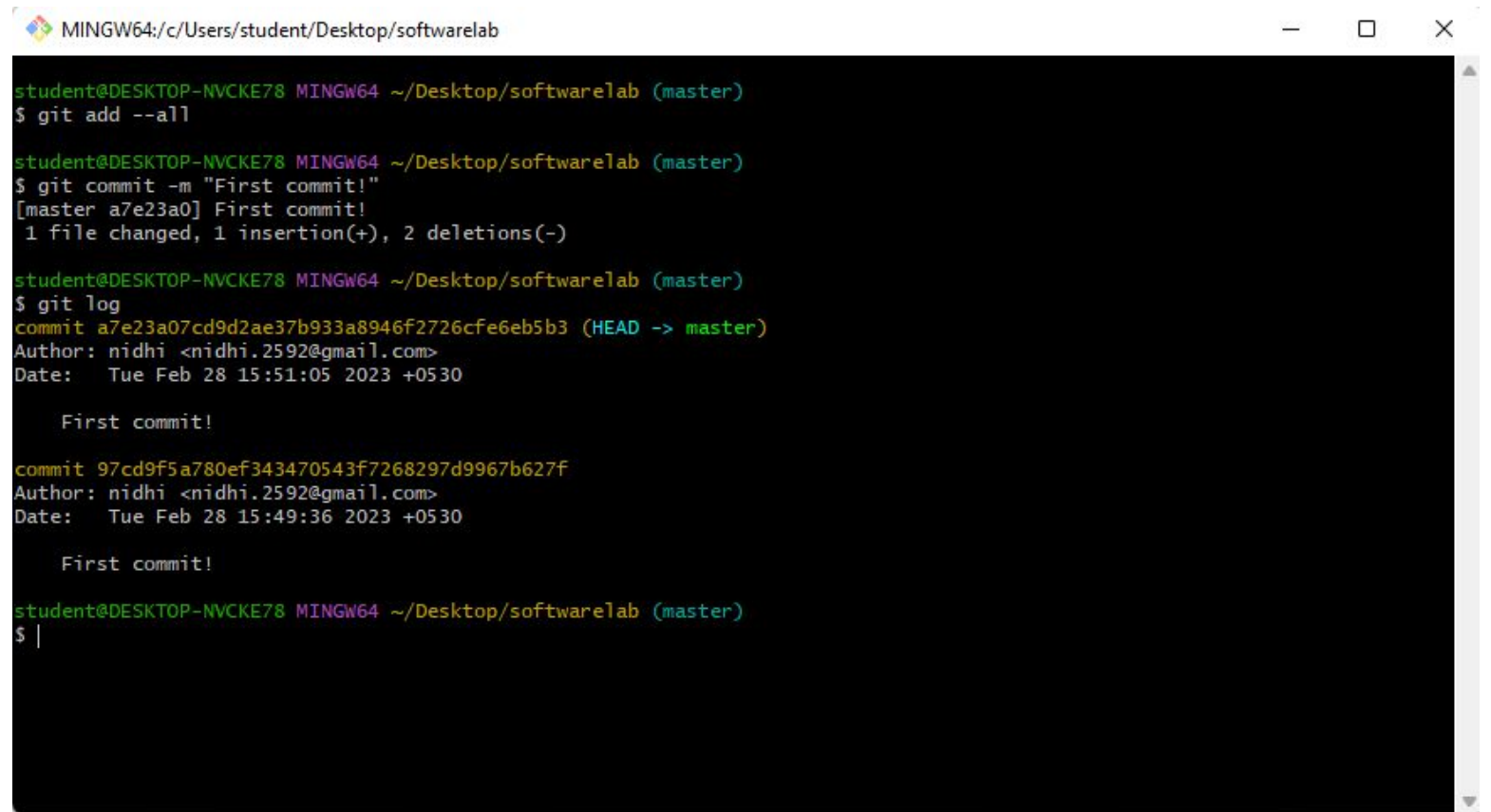
student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (master)
$ git commit -m "First commit!"
[master a7e23a0] First commit!
1 file changed, 1 insertion(+), 2 deletions(-)

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (master)
$ |
```

Git Commit Log

- To view the history of commits for a repository, you can use the log command:

- `git log`



```
MINGW64:/c/Users/student/Desktop/softwarelab

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (master)
$ git add --all

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (master)
$ git commit -m "First commit!"
[master a7e23a0] First commit!
1 file changed, 1 insertion(+), 2 deletions(-)

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (master)
$ git log
commit a7e23a07cd9d2ae37b933a8946f2726cfe6b5b3 (HEAD -> master)
Author: nidhi <nidhi.2592@gmail.com>
Date: Tue Feb 28 15:51:05 2023 +0530

    First commit!

commit 97cd9f5a780ef343470543f7268297d9967b627f
Author: nidhi <nidhi.2592@gmail.com>
Date: Tue Feb 28 15:49:36 2023 +0530

    First commit!

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (master)
$ |
```

Git Branch

- Branches allow you to work on different parts of a project without impacting the main branch.
- When the work is complete, a branch can be merged with the main project.
- You can even switch between branches and work on different projects without them interfering with each other.
- Branching in Git is very lightweight and fast!

New Git Branch

- Let add some new features to our index.html page.
- We are working in our local repository, and we do not want to disturb or possibly in the main project.
- So we create a new branch:
 - `git branch hello-world-images`
 - `git branch hello-world-images * master`

New Git Branch

```
MINGW64:/c:/Users/student/Desktop/softwarelab
bash: README.md: command not found

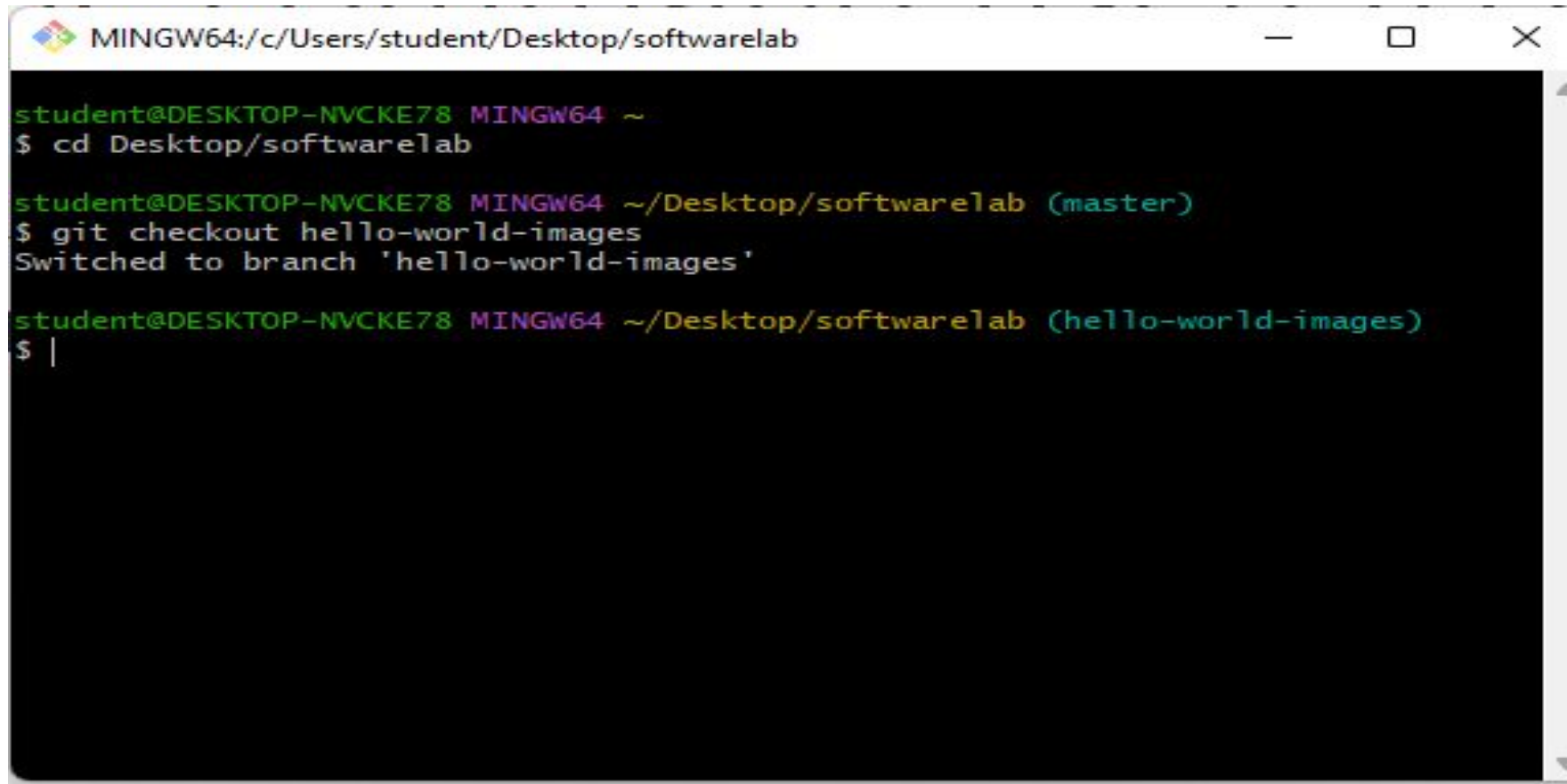
student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (master)
$ git branch hello-world-images * master
usage: git branch [<options>] [-r | -a] [--merged] [--no-merged]
      or: git branch [<options>] [-f] [--recurse-submodules] <branch-name> [<start-point>]
      or: git branch [<options>] [-l] [<pattern>...]
      or: git branch [<options>] [-r] (-d | -D) <branch-name>...
      or: git branch [<options>] (-m | -M) [<old-branch>] <new-branch>
      or: git branch [<options>] (-c | -C) [<old-branch>] <new-branch>
      or: git branch [<options>] [-r | -a] [--points-at]
      or: git branch [<options>] [-r | -a] [--format]

Generic options
  -v, --verbose          show hash and subject, give twice for upstream branch
  -q, --quiet            suppress informational messages
  -t, --track[=(direct|inherit)]
                        set branch tracking configuration
  -u, --set-upstream-to <upstream>
                        change the upstream info
  --unset-upstream       unset the upstream info
  --color[=<when>]       use colored output
  -r, --remotes          act on remote-tracking branches
  --contains <commit>   print only branches that contain the commit
  --no-contains <commit>
                        print only branches that don't contain the commit
  --abbrev[=<n>]         use <n> digits to display object names

Specific git-branch actions:
  -a, --all             list both remote-tracking and local branches
  -d, --delete          delete fully merged branch
  -D                    delete branch (even if not merged)
```

checkout is the command used to check out a branch

- `git checkout hello-world-images`

A screenshot of a Windows command prompt window titled "MINGW64:/c/Users/student/Desktop/softwarelab". The window has standard Windows window controls (minimize, maximize, close) in the top right corner. The terminal text shows the user navigating to the Desktop/softwarelab directory and then using the git checkout command to switch to the 'hello-world-images' branch. The prompt changes from '~' to '~/Desktop/softwarelab (master)' and then to '~/Desktop/softwarelab (hello-world-images)'.

```
MINGW64:/c/Users/student/Desktop/softwarelab

student@DESKTOP-NVCKE78 MINGW64 ~
$ cd Desktop/softwarelab

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (master)
$ git checkout hello-world-images
Switched to branch 'hello-world-images'

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (hello-world-images)
$ |
```

Adding file "icon.png" is not tracked.

```
MINGW64:/c/Users/student/Desktop/softwarelab

student@DESKTOP-NVCKE78 MINGW64 ~
$ cd Desktop/softwarelab

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (master)
$ git checkout hello-world-images
Switched to branch 'hello-world-images'

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (hello-world-images)
$ git status
On branch hello-world-images
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:   index.html

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

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

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (hello-world-images)
$
```

```
MINGW64:/c/Users/student/Desktop/softwarelab

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

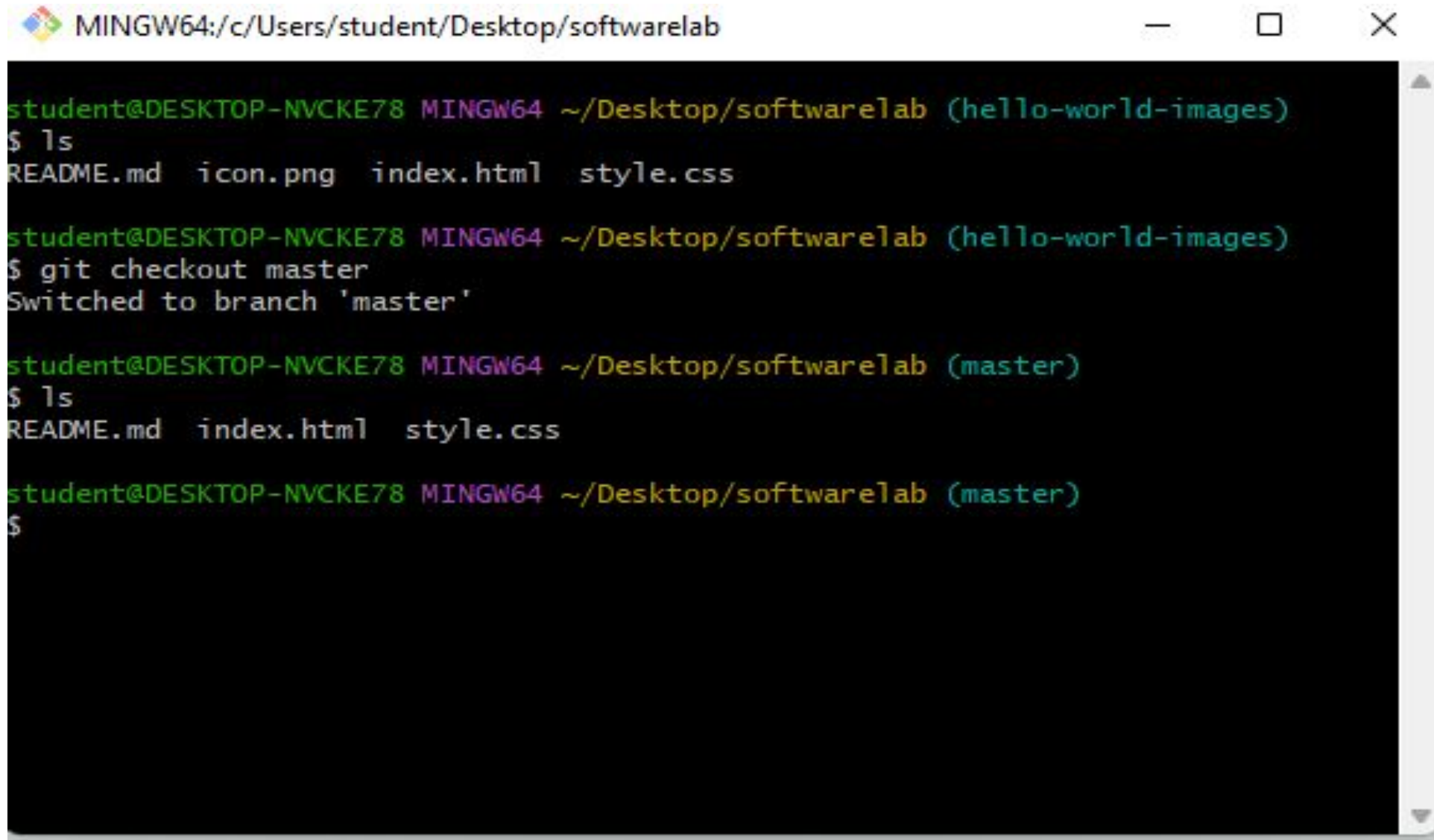
student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (hello-world-images)
$ git add --all

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (hello-world-images)
$ git status
On branch hello-world-images
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   icon.png
        modified:   index.html

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (hello-world-images)
$ git commit -m "Added image to Hello World"
[hello-world-images 5a87ef2] Added image to Hello World
2 files changed, 5 insertions(+), 1 deletion(-)
create mode 100644 icon.png

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (hello-world-images)
$
```

Switching Between Branches

A screenshot of a Windows terminal window titled 'MINGW64:/c/Users/student/Desktop/softwarelab'. The terminal shows a user switching from the 'hello-world-images' branch to the 'master' branch. The user runs 'ls' to see files in the current branch, then 'git checkout master' to switch branches. After running 'ls' again, the file list is updated to reflect the 'master' branch. The terminal text is as follows:

```
MINGW64:/c/Users/student/Desktop/softwarelab

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (hello-world-images)
$ ls
README.md  icon.png  index.html  style.css

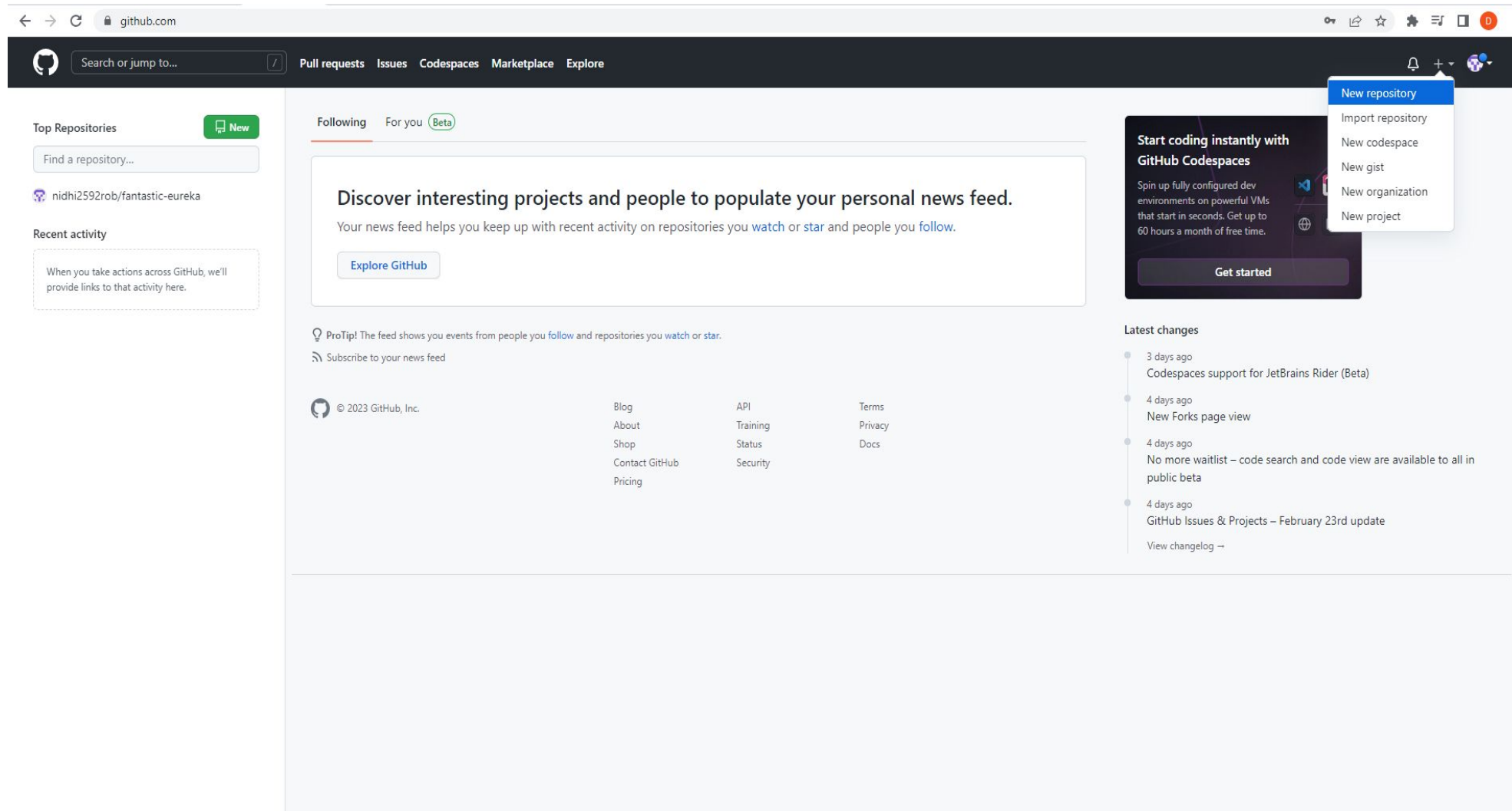
student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (hello-world-images)
$ git checkout master
Switched to branch 'master'

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (master)
$ ls
README.md  index.html  style.css

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (master)
$
```

Git and GitHub

- GitHub Account: Go to GitHub and sign up for an account:



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

[Set up in Desktop](#) or [HTTPS](#) [SSH](#) [Copy](#)

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 "# softwarelab" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/nidhi2592rob/softwarelab.git
git push -u origin main
```

[Copy](#)

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

```
git remote add origin https://github.com/nidhi2592rob/softwarelab.git
git branch -M main
git push -u origin main
```

[Copy](#)

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

[Import code](#)

- **git remote set-url origin URL** specifies that you are adding a remote repository, with the specified URL, as an origin to your local Git repo.
- URL: <https://github.com/nidhi2592rob/softwarelab.git>

```
MINGW64:/c/Users/student/Desktop/softwarelab


student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (master)
$ git remote set-url origin https://github.com/nidhi2592rob/softwarelab.git

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (master)
$ git remote -v
origin https://github.com/nidhi2592rob/softwarelab.git (fetch)
origin https://github.com/nidhi2592rob/softwarelab.git (push)





student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (master)
$ git help log


student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (master)
$ git push --set-upstream origin master
info: please complete authentication in your browser...
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 6 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (8/8), 900 bytes | 900.00 KiB/s, done.
Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/nidhi2592rob/softwarelab.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

student@DESKTOP-NVCKE78 MINGW64 ~/Desktop/softwarelab (master)
$ |
```

 master  1 branch  0 tags

Go to file Add file <> Code


 nidhi2592rob First commit!	a7e23a0 1 hour ago	🕒 2 commits
 README.md	First commit!	1 hour ago
 index.html	First commit!	1 hour ago
 style.css	First commit!	1 hour ago

README.md 


hello-world


Hello World repository for Git tutorial This is an example repository for the Git tutoial.


This repository is built step by step in the tutorial.


About 

No description, website, or topics provided.

 Readme

 0 stars

 1 watching

 0 forks



Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

Languages

 HTML 67.6%  CSS 32.4%