



Git And GitHub Session No.13

Summary 06-03-2022

- **git init (workspace_name)** – command to create a repository
- **cd ws21**
- **touch f1.txt** – command to create a file
- **git add .** – command to add the file to the staging area.
- **git commit . -m f** – command to commit the file
- **git log --oneline** – command to display a simplified one-line summary of commit history.

```
$ git log --oneline
810744a (HEAD -> master) f

$ cat >> a.txt
aaa

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/gitws-workshop
/lwosproj/ws21 (master)
$ git add a.txt
warning: LF will be replaced by CRLF in a.txt.
The file will have its original line endings in your working
directory
```

- In case our file is not completed, we don't commit the file because some triggers at the commit area automatically push the file to Git Hub.
- **git status -s** – command to see files in the working area.

```
/lwsproj/ws21 (master)
$ git status -s
A  a.txt
A  b.txt
```

- To commit all files exclude one or some (which we are working on currently), a temporary space is used called a stash to keep the working files.
- **git stash save "first"** – command to create a stash commented 'first'

```
/lwsproj/ws21 (master)
$ git stash save "first"
Saved working directory and index state On master: first
```

Now the previous files are no longer in the working area.

```
/lwsproj/ws21 (master)
$ git status
On branch master
nothing to commit, working tree clean
```

- **Git stash list** – command to see all the stashes. We can create more than one stash.

```
$ git stash save "sec"
Saved working directory and index state On master: sec

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/gitws-worksho
/lwsproj/ws21 (master)
$ git stash list
stash@{0}: On master: sec
stash@{1}: On master: first
```

- **git stash show stash@{1}** – command to see the content inside the stash with the ID of stash e.g., `stash@{1}`.

```

/lwosproj/ws21 (master)
$ git stash show stash@{1}
a.txt | 1 +
b.txt | 1 +
2 files changed, 2 insertions(+)

```

- **git stash apply stash@{1}** – command to get back the files from stash.

```

/lwosproj/ws21 (master)
$ git stash apply stash@{1}
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   a.txt
    new file:   b.txt

```

- **git status -s** – files are presented back on the staging area.

```

/lwosproj/ws21 (master)
$ git status -s
A  a.txt
A  b.txt

```

- **git stash drop stash@{1}** – Command to delete the stash.

```

/lwosproj/ws21 (master)
$ git stash drop stash@{1}
Dropped stash@{1} (3aea0d678b59826038cd5c6da8a84fb1d91d5883)

```

- **git stash pop stash@{0}** – a command with ‘pop’ option the files have come back and the stash is also deleted.

```

/1wosproj/ws21 (master)
$ git stash pop stash@{0}
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   a.txt
    new file:   b.txt
    new file:   z1.txt

Dropped stash@{0} (a14760de0f459ea20b24ef00991172098291a090)

```

- **git clean -i** – command to clean the working area

```

$ git clean -i
Would remove the following item:
  b.txt
*** Commands ***
  1: clean          2: filter by pattern  3: select
  by numbers
  4: ask each       5: quit           6: help
What now> 4
Remove b.txt [y/N]? y
Removing b.txt

```

- **git init ws22** – create the workplace repository.

Create a file, add, commit 'a1', and create a new branch

```
/lwsproj/ws22 (master)
$ touch a

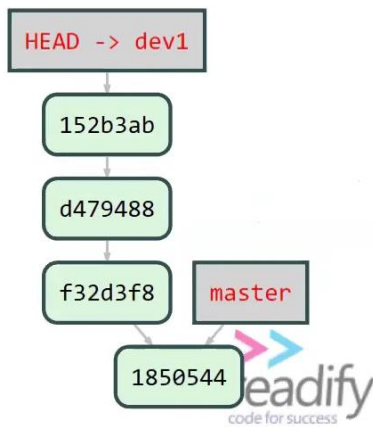
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/gitws
/lwsproj/ws22 (master)
$ git add .
git
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/gitws
/lwsproj/ws22 (master)
$ git commit . -m a1
[master (root-commit) 1850544] a1
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 a

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/gitws
/lwsproj/ws22 (master)
$ git checkout -b dev1
Switched to a new branch 'dev1'
```



Add some more files to do the lab-ready

Local Repo Path: >aga\Documents\gitws-workshop\lwi ... Resize



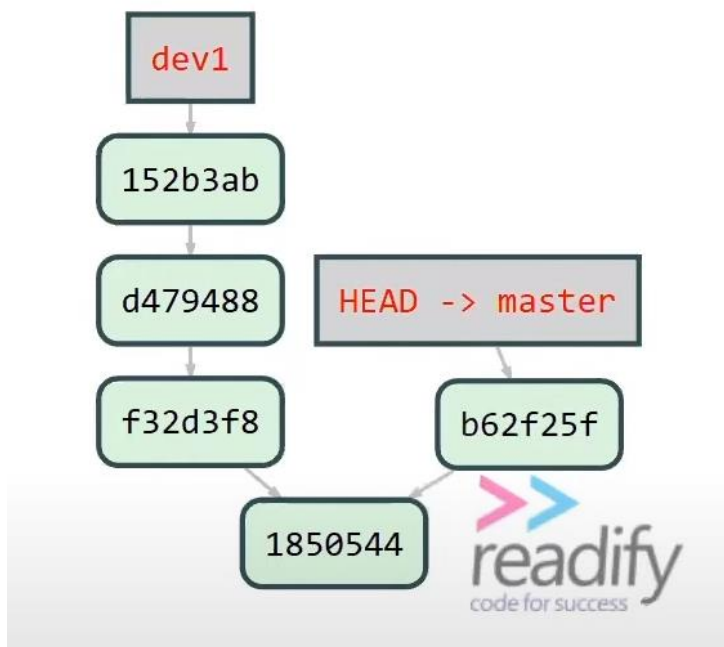
cp -r (source) (dest) – command to copy the data from one folder to another.

cp -r ws22 ws23

```
/lwsproj (master)
$ cp -r ws22 ws23
```

- **Cherry picking** (3-way merging strategy) - Cherry-picking is a Git strategy where you choose specific commits from one branch and apply them to another branch.

```
/lwsproj/ws22 (master)
$ git cherry-pick d479488
[master b62f25f] d2
Date: Sun Mar 6 12:29:04 2022 +0530
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 c
```

- **Restoring or rolling back** changes in Git can be risky, particularly when removing a commit or node from the history, as it may be possible that another branch is dependent on the same node.
- **Revert** - revert creates a new commit that reverts changes made by another commit. It's essentially a reversed **cherry-pick**, rolling your history forward.
- **git revert d479488** – command to revert the node.

```
/lwsproj/ws23 (dev1)
$ git revert d479488
[dev1 1e7b4ee] Revert "d2 my revety mgd"
1 file changed, 0 insertions(+), 0 deletions(-)
delete mode 100644 c
```

We can add some comments.

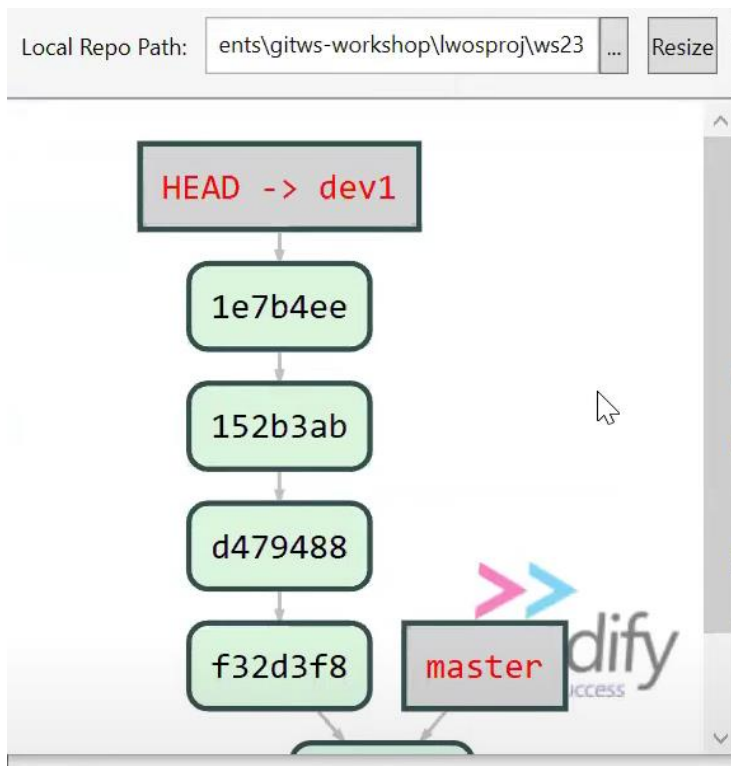
```
Revert "d2 my revety mgd"
```

```
This reverts commit d479488d0d12938663128f378d746c79f954e3b5.
```

```
# Please enter the commit message for your changes. Lines starting  
# with '#' will be ignored, and an empty message aborts the commit.  
#  
# On branch dev1  
# Changes to be committed:  
#   deleted:    c  
#
```

We can see that node 'c' is deleted.

```
/lwsproj/ws23 (dev1)  
$ ls  
a b d
```



- **git rebase -i HEAD~3** – command will start an interactive rebase for the last 3 commits.


```

/1wosproj/ws23 (dev1)
$ git rebase | -i HEAD~3
[detached HEAD 21f297c] d3
Date: Sun Mar 6 12:29:15 2022 +0530
2 files changed, 0 insertions(+), 0 deletions(-)
delete mode 100644 c
rename b => d (100%)
Successfully rebased and updated refs/heads/dev1.

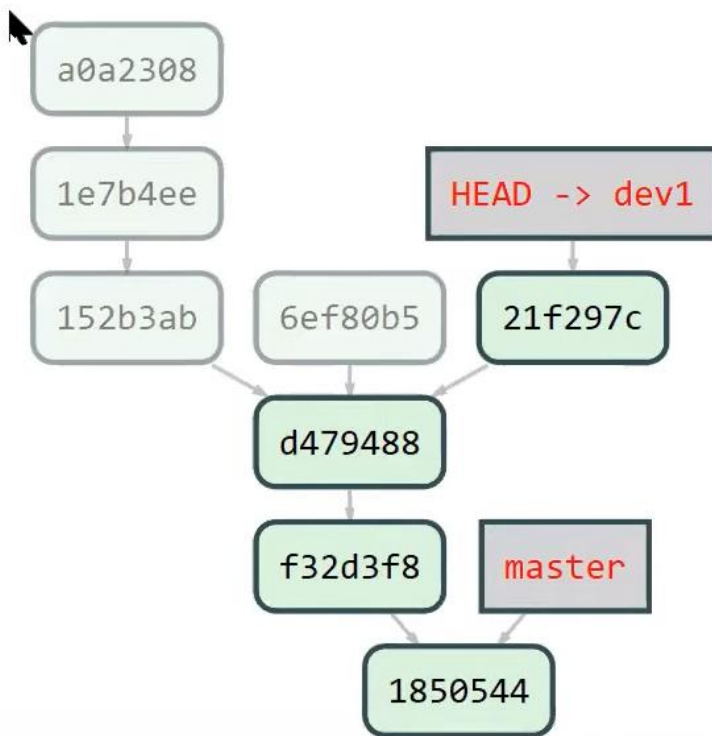
```

As this pop-up is opened along with adding the comments, we can change some commands here like below.

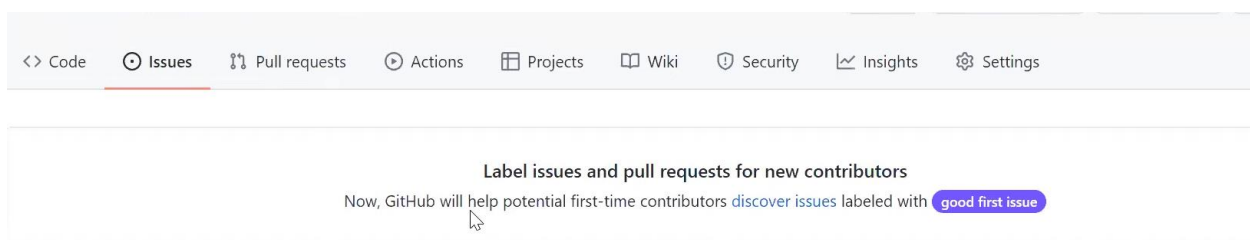
```

1 pick 152b3ab d3
2 pick 1e7b4ee Revert "d2 my revety mgd"
3 pick a0a2308 Revert "d1"
4
5 # Rebase d479488..a0a2308 onto d479488 (3 commands)
6 #
7 # Commands:
8 # p, pick <commit> = use commit
9 # r, reword <commit> = use commit, but edit the commit message
10 # e, edit <commit> = use commit, but stop for amending
11 # s, squash <commit> = use commit, but meld into previous commit
12 # f, fixup [-C | -c] <commit> = like "squash" but keep only the previous
13 #                      commit's log message, unless -C is used, in which case
14 #                      keep only this commit's message; -c is same as -C but
15 #                      opens the editor
16 # x, exec <command> = run command (the rest of the line) using shell
17 # b, break = stop here (continue rebase later with 'git rebase --continue')
18 # d, drop <commit> = remove commit
19
1 pick 152b3ab d3
2 squash 1e7b4ee Revert "d2 my revety mgd"
3 squash a0a2308 Revert "d1"
4
5 # Rebase d479488..a0a2308 onto d479488 (3 commands)
6 #
7 # Commands:
8 # p, pick <commit> = use commit
9 # r, reword <commit> = use commit, but edit the commit message
10 # e, edit <commit> = use commit, but stop for amending
11 # s, squash <commit> = use commit, but meld into previous commit
12 # f, fixup [-C | -c] <commit> = like "squash" but keep only the previous
13 #                      commit's log message, unless -C is used, in which case
14 #                      keep only this commit's message; -c is same as -C but
15 #                      opens the editor
16 # x, exec <command> = run command (the rest of the line) using shell
17 # b, break = stop here (continue rebase later with 'git rebase --continue')
18 # d, drop <commit> = remove commit
19 # l, label <label> = label current HEAD with a name

```



- **Issues** – Issuing a bug in the issues section means we are assigning it to the team for efficient resolution.
 - Click on ‘New issue’



- Give the name to the issue, and add some comments.
- Check the option on the right side, can assign to people particularly it will notify them, can add labels, can add milestones.

i find readme file not exists

Write Preview

comment from owner manager

Attach files by dragging & dropping, selecting or pasting them.

Styling with Markdown is supported

Submit new issue

Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

Assignees
vimallinuxworld13

Labels
critical

Projects
None yet

Milestone
ms2week

Development
Shows branches and pull requests linked to this issue.

- GitHub gives the **issue number** to the issue.

i find readme file not exists #96

Open vimallinuxworld13 opened this issue now · 0 comments

vimallinuxworld13 commented now

comment from owner manager

Edit New issue

- As the issue is resolved, the issue will be status closed.

Write Preview

Leave a comment

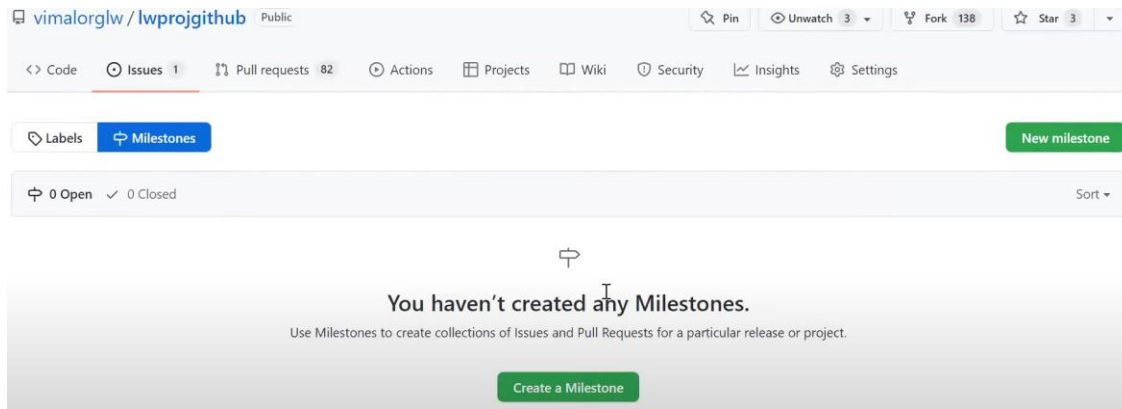
Attach files by dragging & dropping, selecting or pasting them.

Close issue Comment

Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

Assignees
vimallinuxworld13

- **Milestones**- we can create a milestone where we can add the timeline to fix the bug and can add some labels too.



- Give the name of the milestone and the date for the deadline.

New milestone

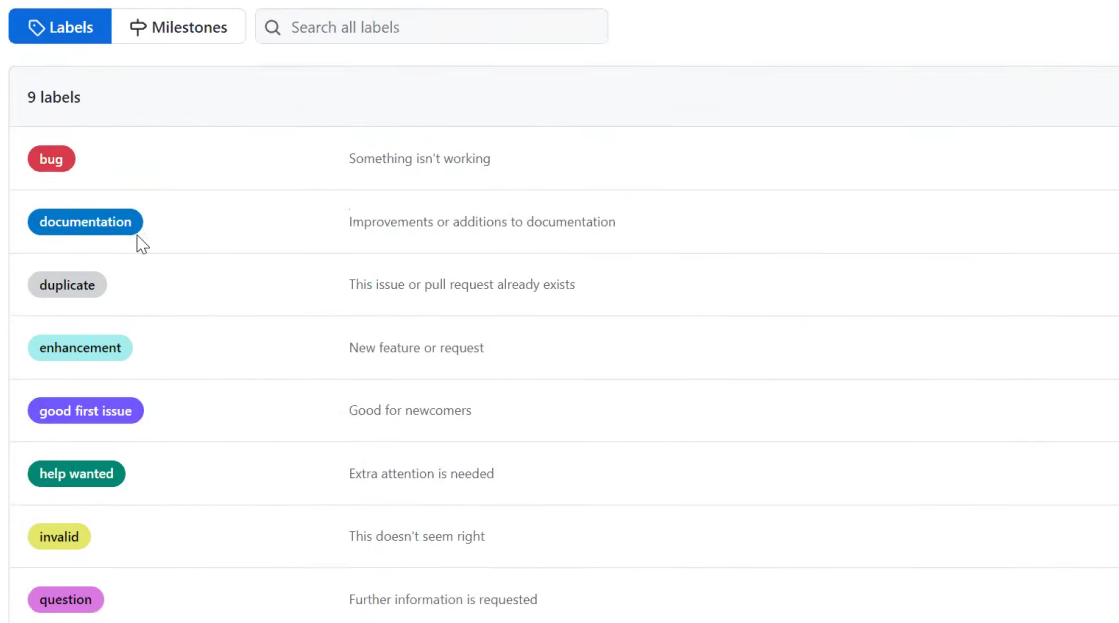
Create a new milestone to help organize your issues and pull requests. Learn more about [milestones](#) and [issues](#).

Title

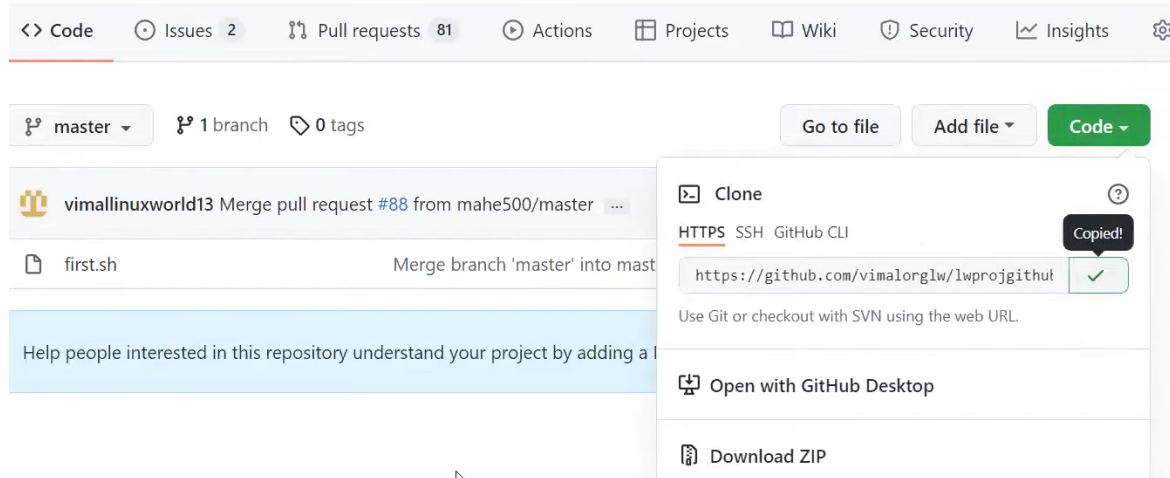
Due date (optional)

Description

- Labels according to the problem.



- If the employee to whom the issue is assigned is working on the command line. The issue can be cloned, copy the link as follows.



- Run the command **git clone** followed by the link and can work further.

```
/lwsproj/ws23 (dev1)
$ git clone https://github.com/vimalorglw/lwpr
Cloning into 'lwprojgithub'...
remote: Enumerating objects: 14, done.
remote: Counting objects: 100% (14/14), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 14 (delta 0), reused 6 (delta 0)
Receiving objects: 100% (14/14), done.
```

```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents
/lwsproj/ws23 (dev1)
$ ls
a d lwprojgithub/
```

```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents
/lwsproj/ws23 (dev1)
$ cd lwprojgithub/
```

```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents
/lwsproj/ws23/lwprojgithub (master)
$ |
```

```
$ git remote -v
origin https://github.com/vimalorglw/lwprojgithub.git (fetch)
origin https://github.com/vimalorglw/lwprojgithub.git (push)
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/gitws-worksh
/lwsproj/ws23/lwprojgithub (master)
$
```



```

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/gitws-works
/lwosproj/ws23/lwprojgithub (master)
$ ls
first.sh

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/gitws-works
/lwosproj/ws23/lwprojgithub (master)
$ cat > README.md
this i local

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/gitws-works
/lwosproj/ws23/lwprojgithub (master)
$ git add .
warning: LF will be replaced by CRLF in README.md.
The file will have its original line endings in your working
directory

```

- **git commit . -m “from local, close #96”** – in the commit command, we can add the status of the issue with the issue number.

```

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/gitws-works
/lwosproj/ws23/lwprojgithub (master)
$ git commit . -m "from local, close #96"
warning: LF will be replaced by CRLF in README.md.
The file will have its original line endings in your working
directory
[master fa16f16] from local, close #96
 1 file changed, 1 insertion(+)
 create mode 100644 README.md

```

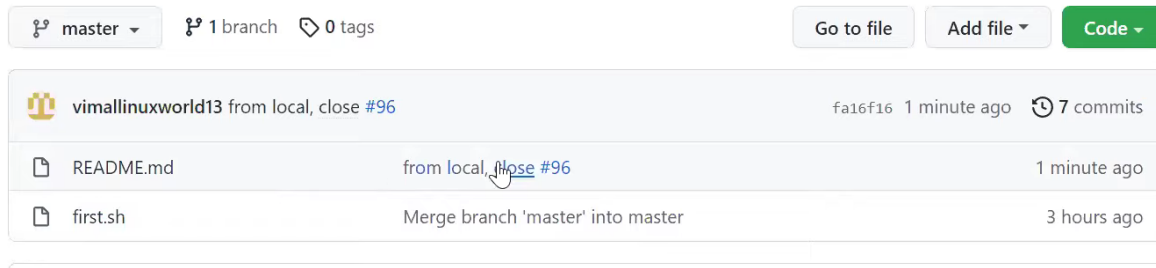
- **git push** – command to push the data on Git Hub, do refresh.

```

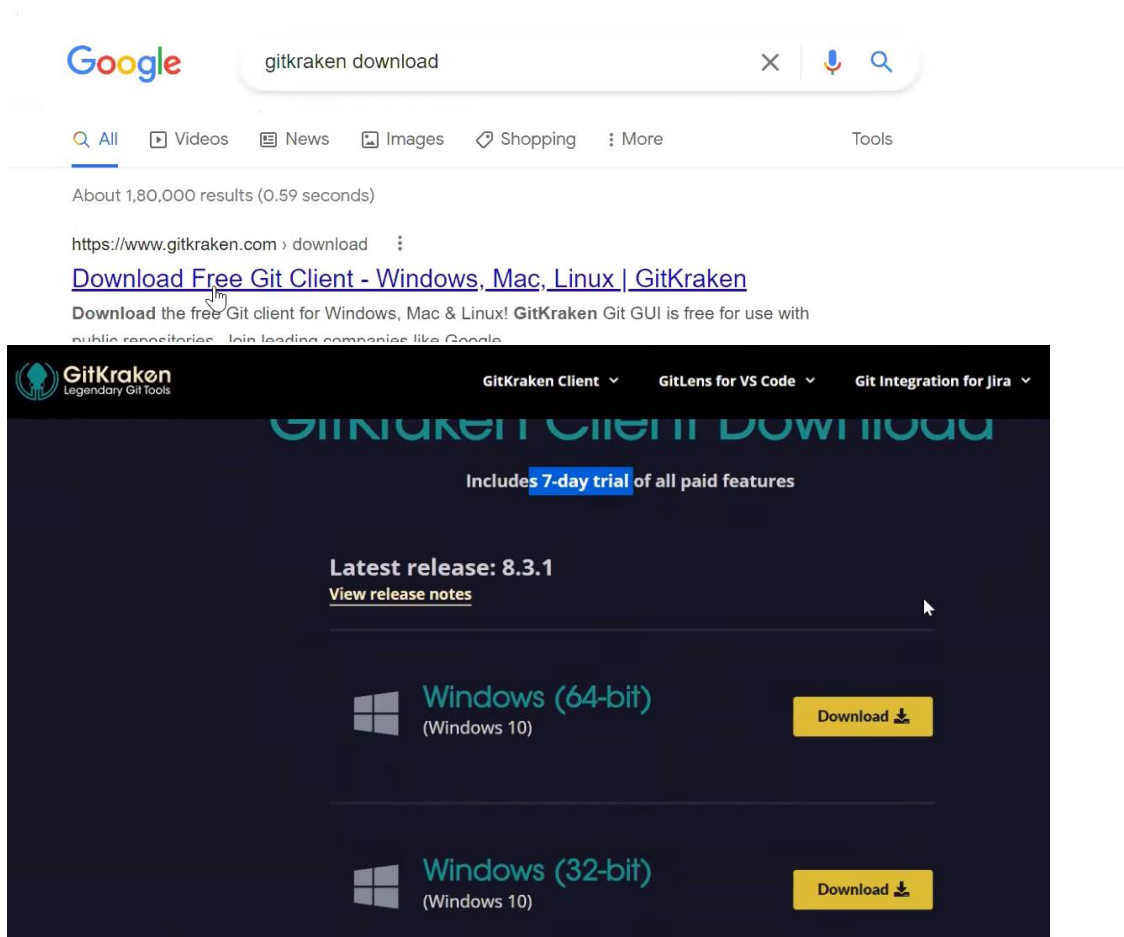
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 298 bytes | 99.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/vimalorglw/lwprojgithub.git
 5e34bd9..fa16f16  master -> master

```

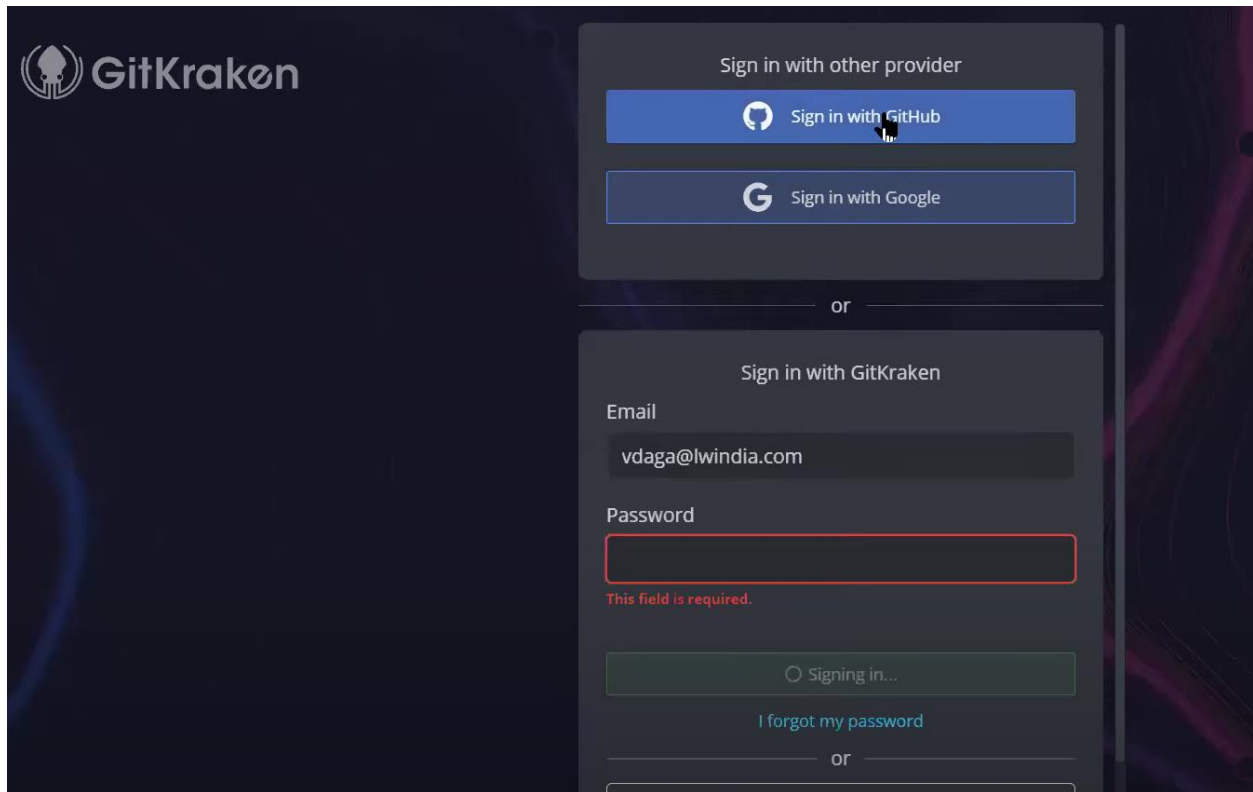
- We can see the issue is closed.



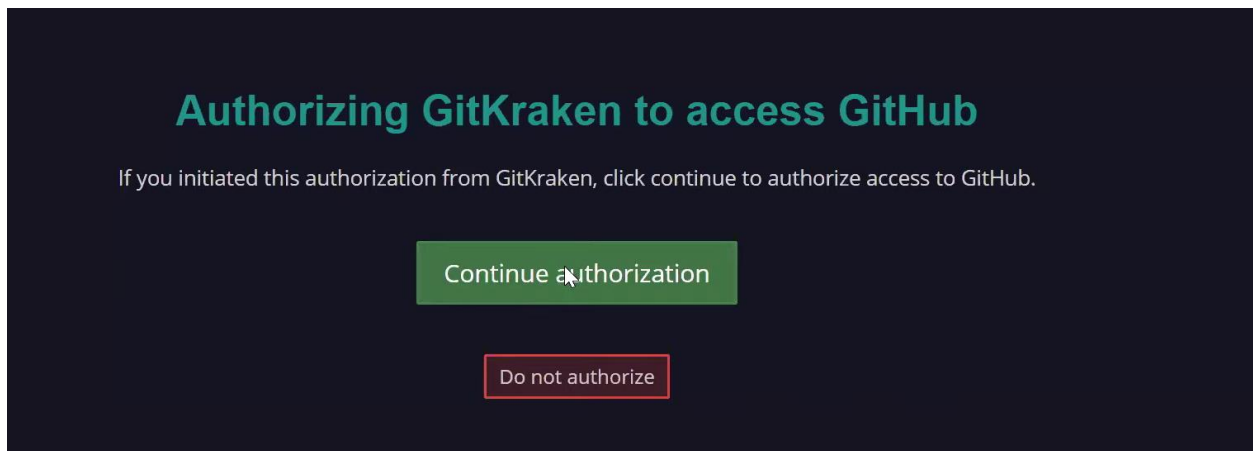
- **Gitkraken** - GitKraken is a graphical user interface (GUI) client for Git. It provides an intuitive way to interact with Git repositories, allowing users to visualize branches, commits, and changes, as well as perform common Git operations like committing, merging, and branching
- Download and install the gitkraken form www.gitkraken.com



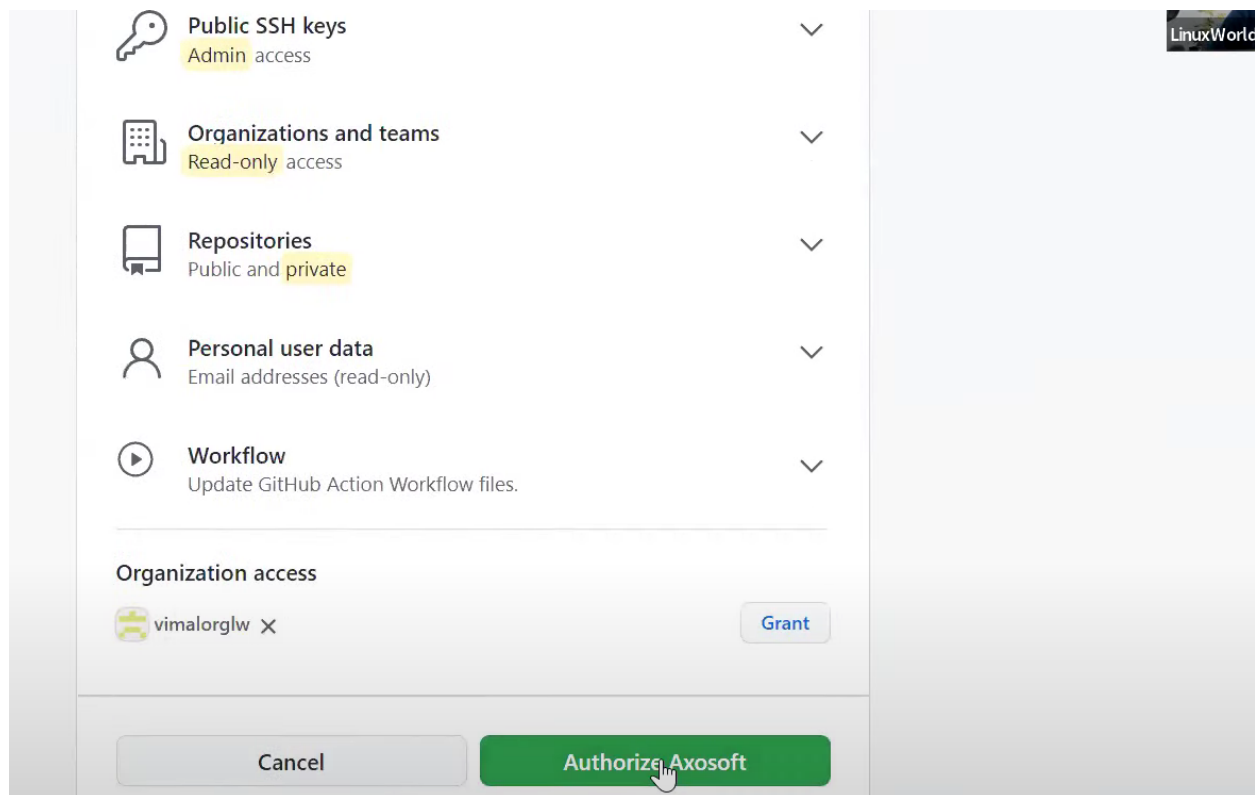
- Sign up on gitkraken with GitHub.



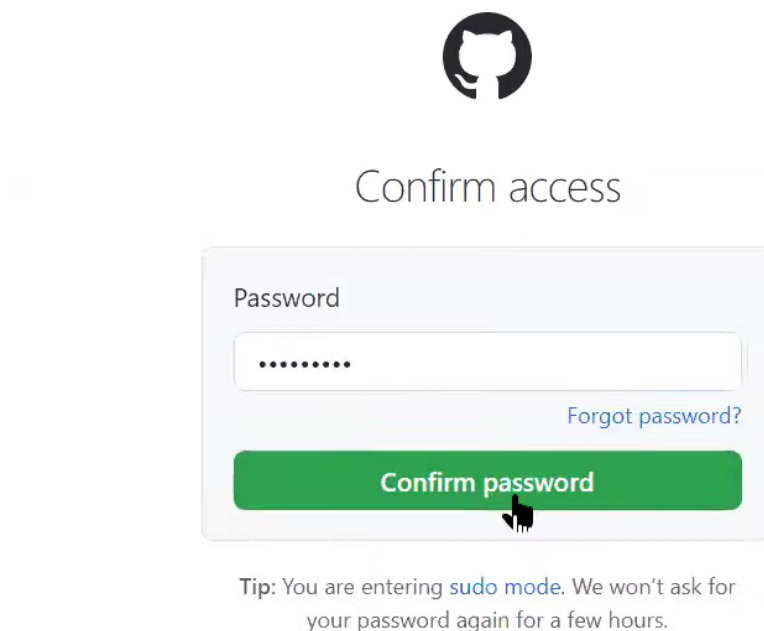
- Click on ‘continue authorization’



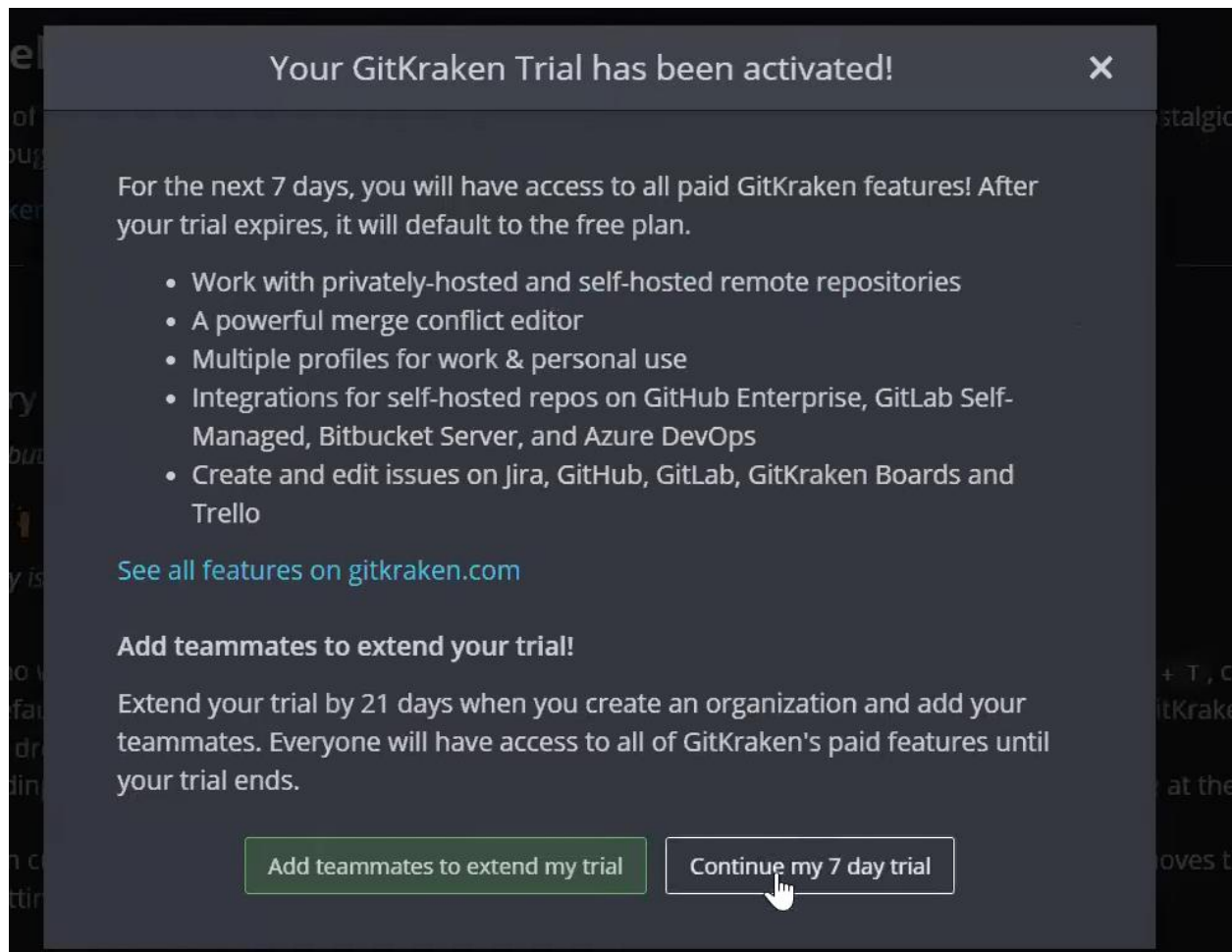
- Click on “Authorize Axosoft”



- Confirm the password.



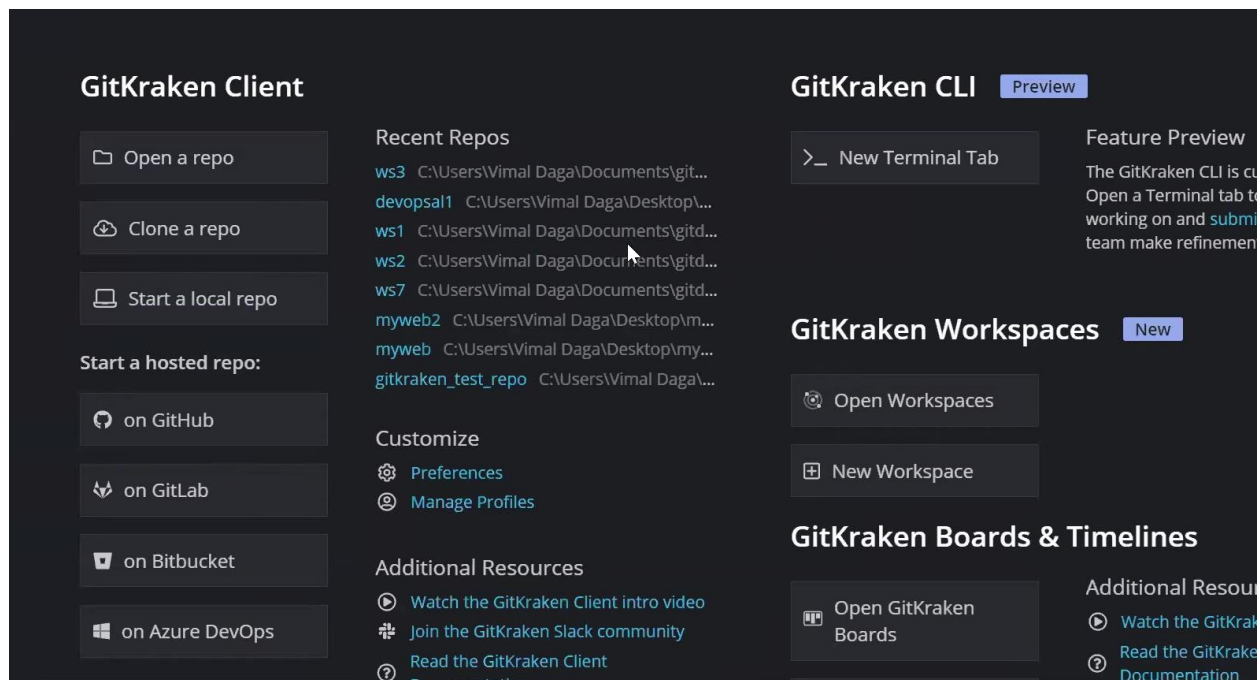
- Continue 7-day trial.



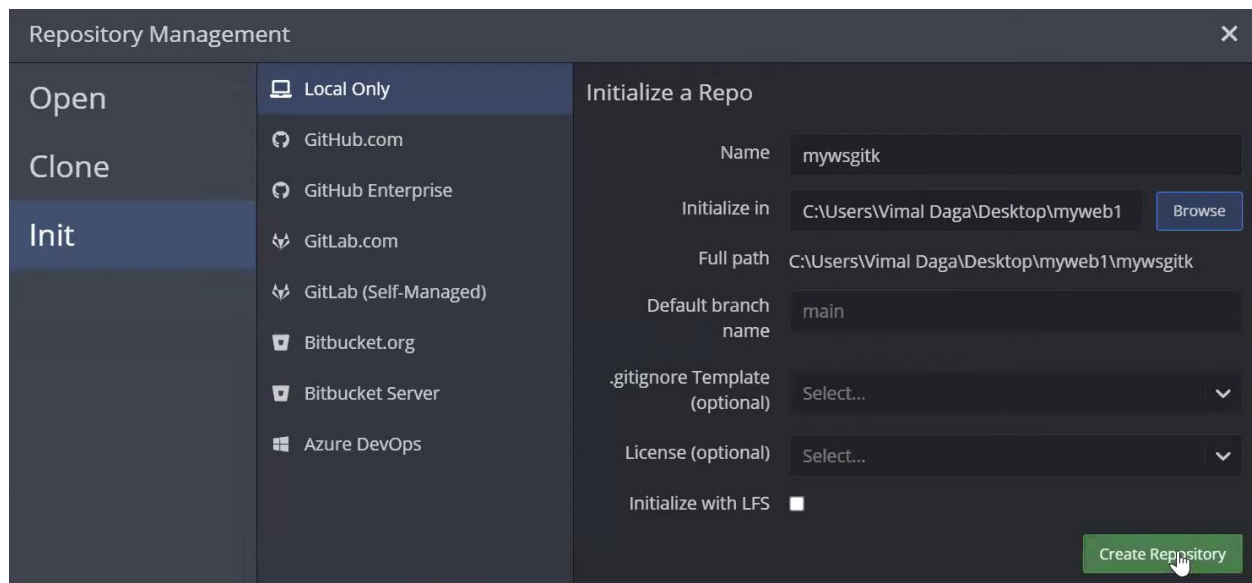
- Accept the agreement.



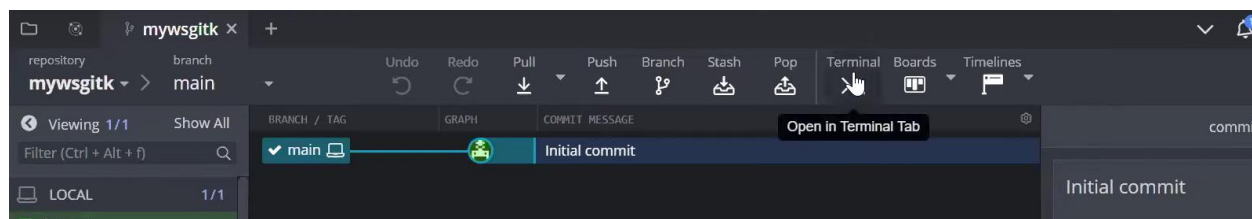
- We can create the open repository, a clone repository, or a local repository.



➤ Let's create a local repo. Click on Create Repository.



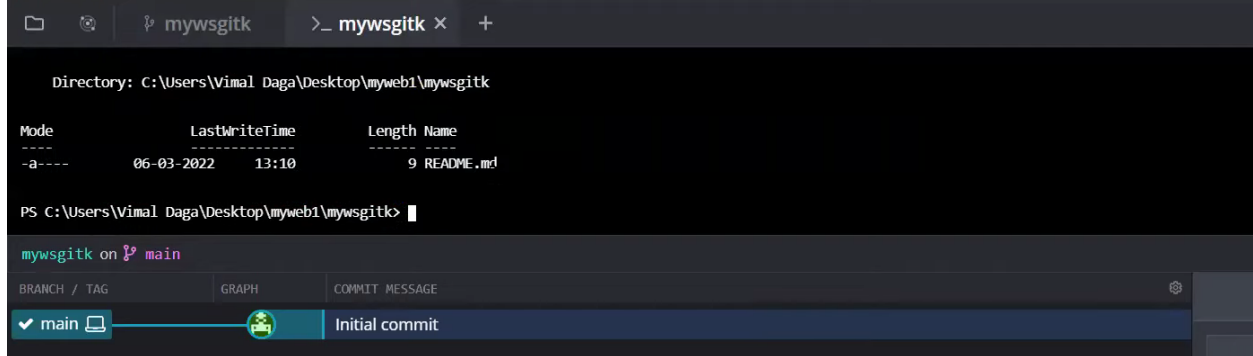
➤ Open the “Terminal”



➤ `cat > a.txt` – create a file.


```
PS C:\Users\Vimal Daga\Desktop\myweb1\mywsgitk> cat > a.txt

cmdlet Get-Content at command pipeline position 1
Supply values for the following parameters:
Path[0]: aaaaa
Path[1]:
```



Directory: C:\Users\Vimal Daga\Desktop\myweb1\mywsgitk

Mode	LastWriteTime	Length	Name
-a----	06-03-2022 13:10	9	README.md

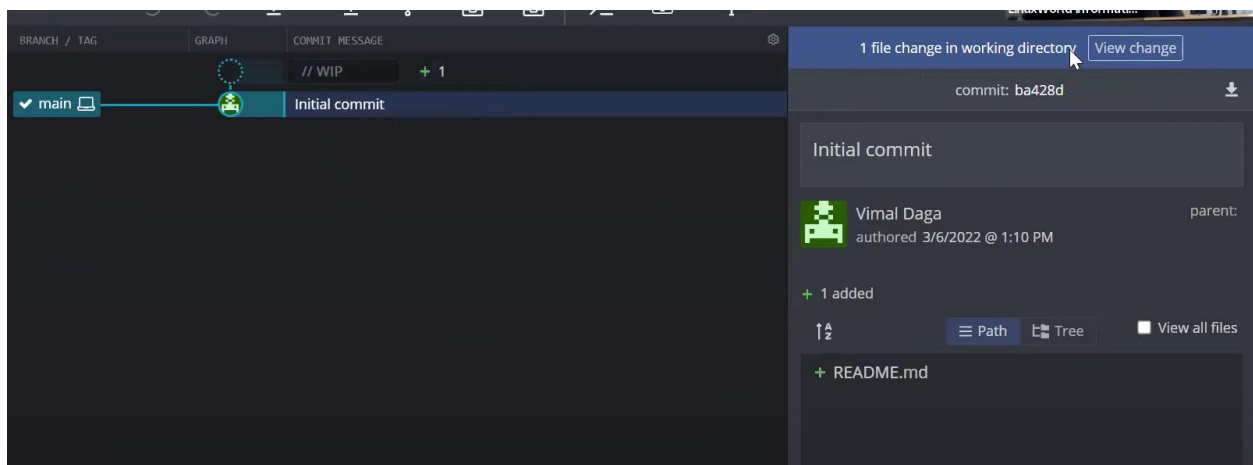
PS C:\Users\Vimal Daga\Desktop\myweb1\mywsgitk>

mywsgitk on **main**

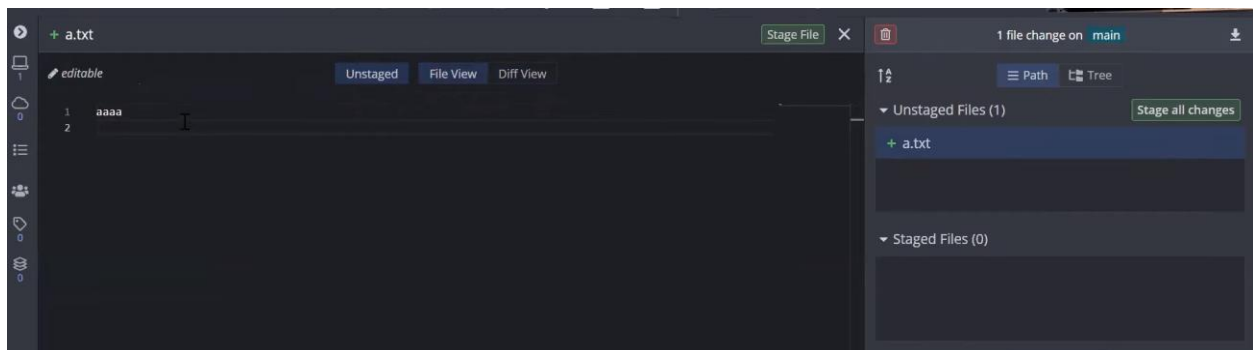
BRANCH / TAG GRAPH COMMIT MESSAGE

✓ main Initial commit

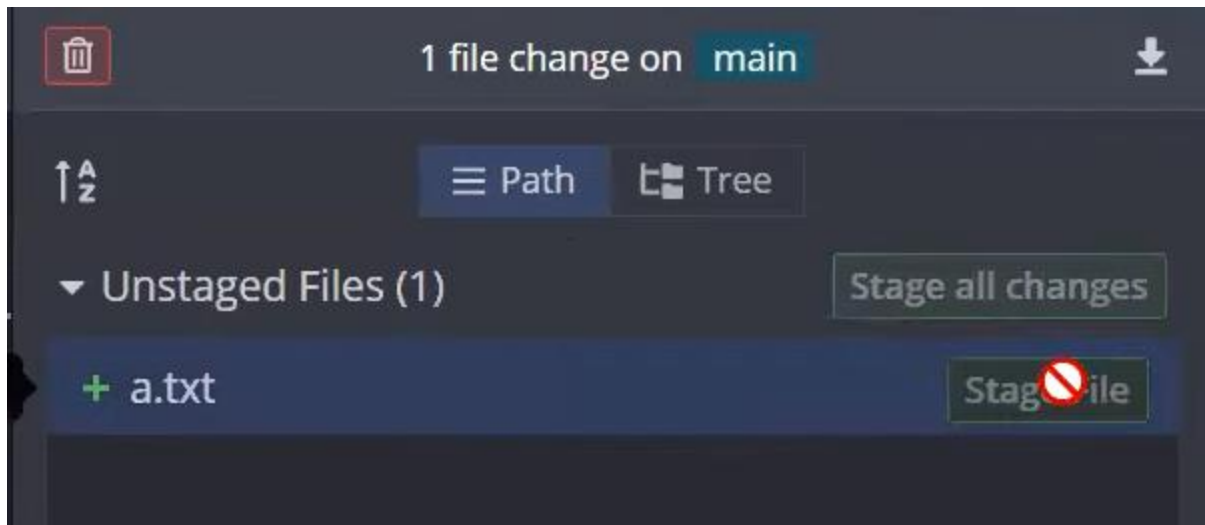
➤ Click on ‘view change’ to see the changes.



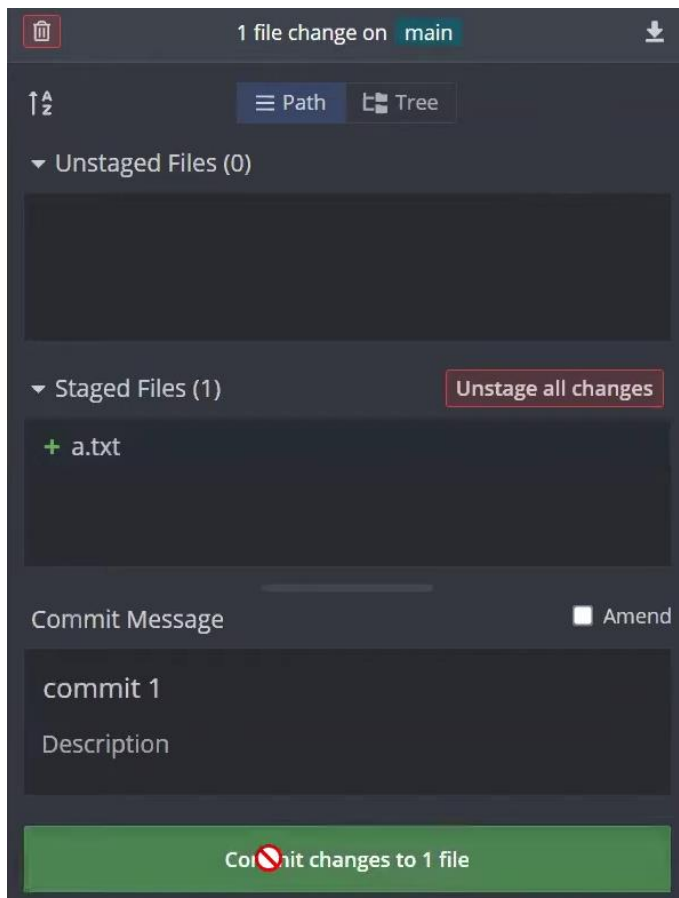
➤ We can see the file we created, which can be edited graphically here.



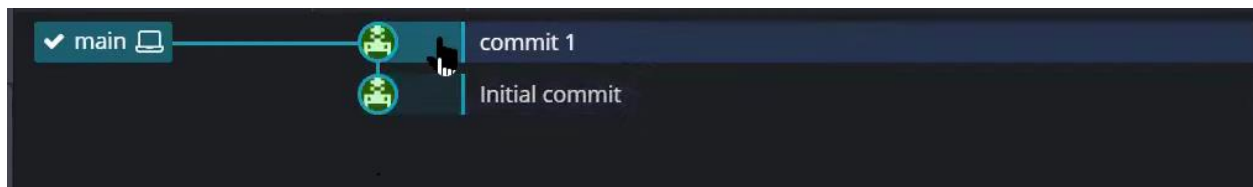
➤ Click on “stage file” to stage the file.



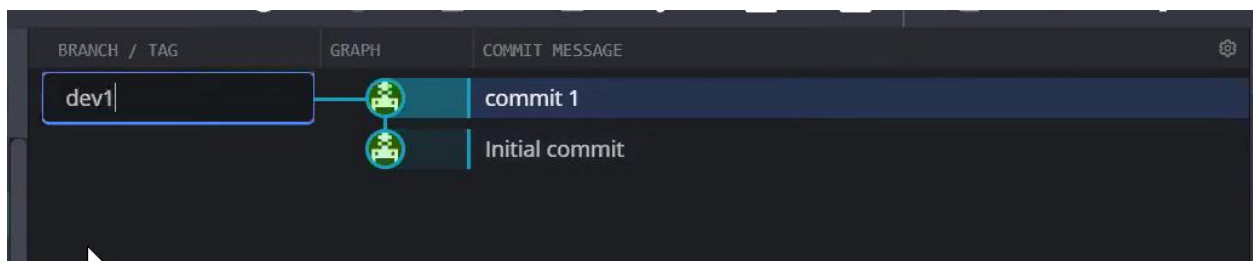
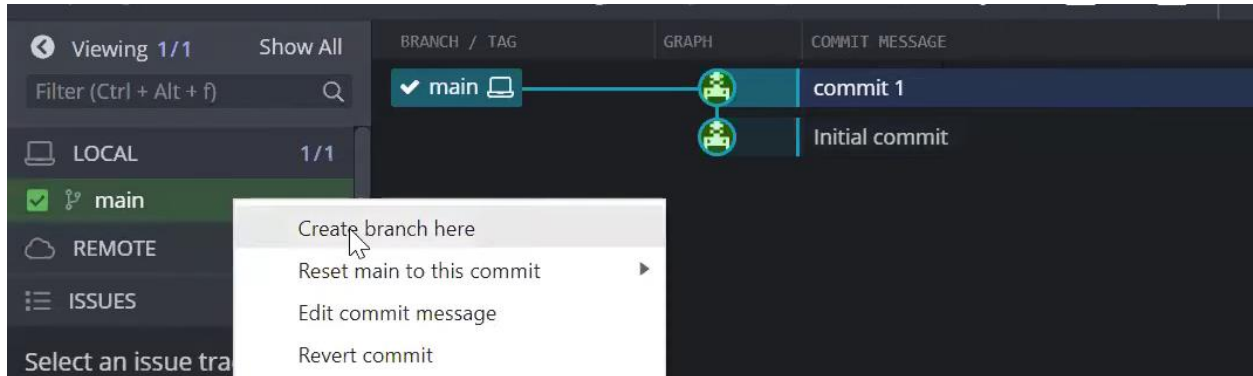
- To commit the file, on the right side, write “commit 1” in the summary and click on “commit changes to 1 file”.



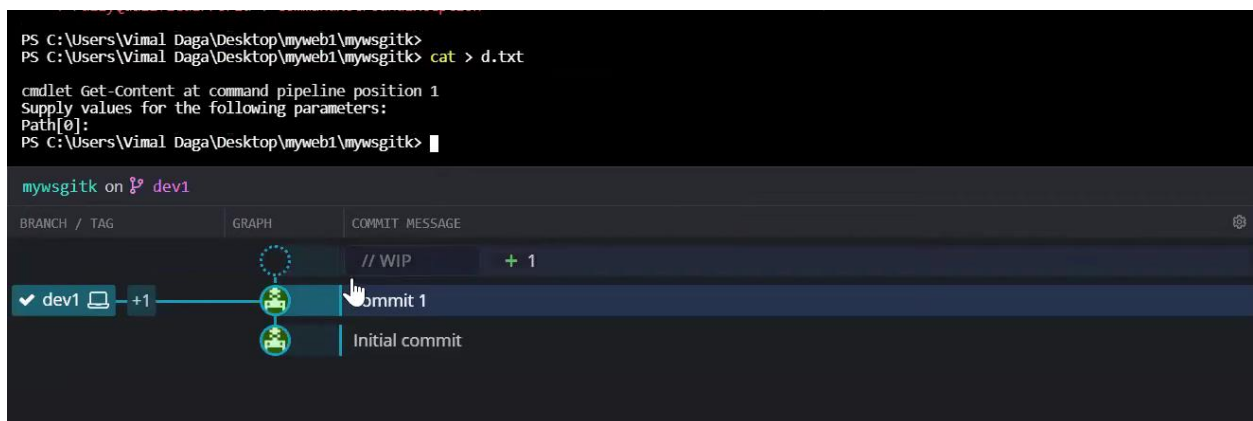
- Main head connected to commit 1.



➤ We can create the new branch from here



➤ Create a new file inside the dev1 branch.



➤ Do stage file or commit file similarly.

