

# SE489 DevOps Engineering

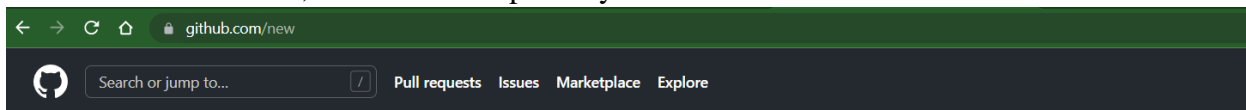
## Lab 3



## Lab 3: GitHub

**Objective:** After successful completion of this lab, students will understand concepts and commands of GitHub and would be able to develop code collaboratively.

1. Open URL <https://github.com/new> use your credentials if you have already registered, if not, sign on, and then it will bring you to this interface, create a new repository



### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner \*  / Repository name \*

Great repository names are short and memorable. Need inspiration? How about [refactored-disco?](#)

#### Description (optional)

- Public**  
Anyone on the internet can see this repository. You choose who can commit.
- Private**  
You choose who can see and commit to this repository.

#### Initialize this repository with:

Skip this step if you're importing an existing repository.

- Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)

#### Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

---

### Initialize this repository with:

Skip this step if you're importing an existing repository.

**Add a README file**

This is where you can write a long description for your project. [Learn more.](#)

### Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: **None** ▾

### Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

License: **None** ▾

This will set  **main** as the default branch. Change the default name in your [settings](#).

---

 You are creating a public repository in your personal account.

---

**Create repository**

With this you have created a public repository in your account

2. Let's make a SSH secure connection with this repository, this is necessary to secure your public repository from malicious or erroneous pushes. To achieve this, do followings
  - a. Open git bash terminal
  - b. Navigate to local repository
  - c. On \$ prompt, write **ssh-keygen**
  - d. It will ask for **file name** to which key will be stored, don't write anything and when it asks for passphrase also don't provide anything, by default, key will be saved into id\_rsa.pub(public key) and id\_rsa(private key)

```
MINGW64/d/DevOps Tools/Lab Manual
mzafa@LAPTOP-F33P6U06 MINGW64 /d/DevOps Tools/Lab Manual (master)
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/mzafa/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in id_rsa
Your public key has been saved in id_rsa.pub
The key fingerprint is:
SHA256:YMoFAZ3ODPe1AzaFLPzJizQJhMc3yYo8Nk4fio1Y1f0 mzafa@LAPTOP-F33P6U06
The key's randomart image is:
+---[RSA 3072]-----+
|O+.=O=..
|O */ /O. O
|. + / B+O .
|. **OB++ . E
|B+=O+ . .S
|=O...
+-----[SHA256]-----+
mzafa@LAPTOP-F33P6U06 MINGW64 /d/DevOps Tools/Lab Manual (master)
```

3. The key shown here is masked, to see the complete key, let's open the *sshkey* file, use command **cat file\_name** to print the contents of the file at terminal
  - a. To see the list of files in the repository, use **ls**.

```
MINGW64/d/DevOps Tools/Lab Manual
mzafa@LAPTOP-F33P6U06 MINGW64 /d/DevOps Tools/Lab Manual (master)
$ ls
DEMO.java demo111.java demo222.java id_rsa id_rsa.pub
mzafa@LAPTOP-F33P6U06 MINGW64 /d/DevOps Tools/Lab Manual (master)
$ |
```

- b. second last file contains private key, we don't need it, the last file with .pub extension is the sought file containing public key, use cat command to open it.

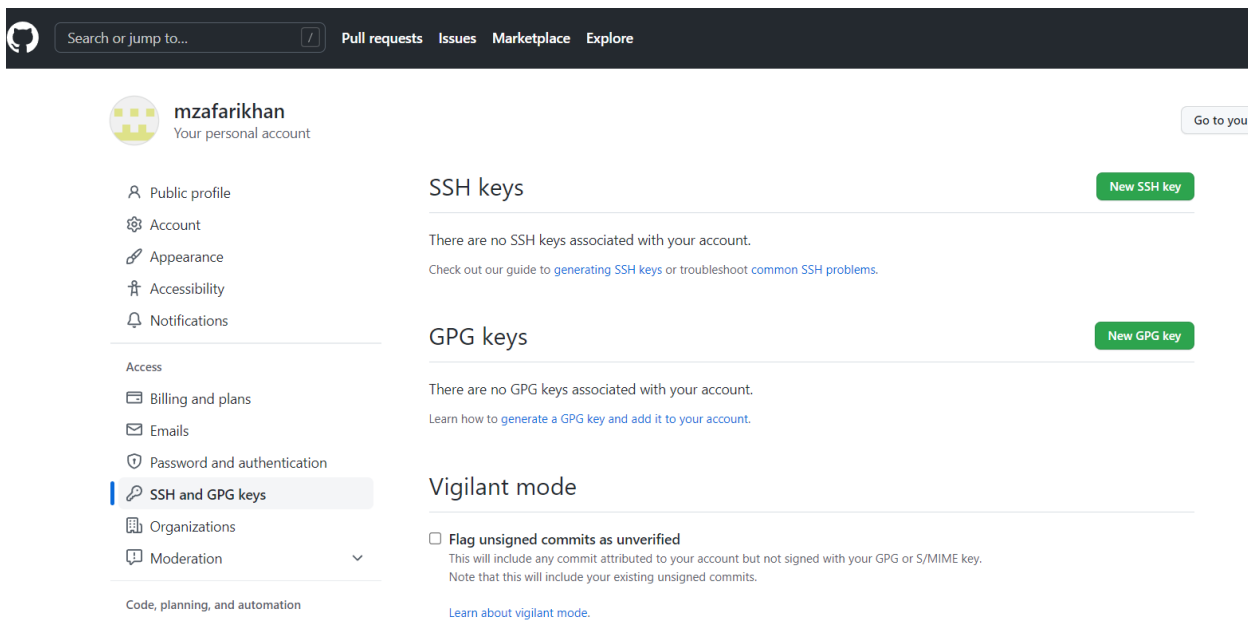
```
MINGW64/d/DevOps Tools/Lab Manual
mzafa@LAPTOP-F33P6U06 MINGW64 /d/DevOps Tools/Lab Manual (master)
$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGCfzspLybRVBoJivBJxCPcnQIC+veiWkKhkzSqEzngCc
kFebWY9sRe7iVPO6dtpMon9EQqmz092HDFdjavgPy6ozomZkjV3retJzkqmvJt7iLA9C/0gdBjk7txdF3g
t2YWWQxryZ0PhyQL+m1xHrxLwb5zstMkpOD07Ys8TZTT/9tb4X+8Vh6eu91hNrNT7yB500p0VJTNpyzU8B
d1/nD0Uc3PgtAcOfbvOx/dfH/R5RfL1CpBPjIv9WwV/Q7zZ/hQRluUGOM13mKoRue9L/gWj6BzEMm95j5B
YcgYqhrs5hIBdaZBPwtstGfY7tmFDv/4NM9xDpCdR61QsUCvJtgYPhRUhYGdQ04DRL+7eAUsFkPwY+mYX
308N+5lRF9rJMzb5WqnrRa/umiUQggmTwd3eTswOjq/miE8Fy0r1bi0MF/WEn4UcYhtxCUEK2DnuG2uvGU
6CK7zZS95gGmlawq2Z+tPi7pM0eKA5CTAZef4p143US8o2REvaBRslh1iic= mzafa@LAPTOP-F33P6U06
mzafa@LAPTOP-F33P6U06 MINGW64 /d/DevOps Tools/Lab Manual (master)
$
```

copy this key

Alternatively

`$cat id_rsa.pub | clip`, will copy the key into clipboard without displaying it

#### 4. Now go back to GitHub website → Settings → SSH and GPG keys → New SSH key



#### 5. Paste the copied key into the space given, assign it a name and then finally click on Add SSH key

## SSH keys / Add new

Title

DevOpsKey

Key

```
ssh-rsa
AAAAAB3NzaC1yc2EAAAADAQABAAQgQDlV58ySfUjUw7IW2BSg/xw4aCWzrAWJk8M9afHdcUQC+FOtYsk3FJKLVZThCUT
5V0AMQCOA6lw03ApZJhOILoZo2uyNRN7glnaKXDuy7ulUhd/WR4WW+3KZrO/qUyejD1kChBx7L9+cMCsehwZ57Sj05Qq
W2LHP5JLtvIjSdbqifVvYI2QxcsgaMj03LRZIXO+oOo1DybyBe0nXz/HXeerELH/TazlIGP5C88tHacvQTDc4XjdPgmhHTk1DI
+lKc8d/YbkAqmNw1KdRXe3QYXwmTu5c2BHE909Bif7+wXUz6EGg/SAnOzEZNNV9uCPFJ7S00Rcq1u9oHCMqF9sDHq+
3kha2o8IX/hfRC6KLz1hh8hKqZfzfAz5h8g5PDCYAW++ZHal7dQGmqwNS09ZKdf406/jXrUhgGqHcwg2/vZ8RsKvstPS1Nh
tUEuVR5LkUpzWHRQg5X+EbWbVnHsGGGLlQaOfpbo1rsuHr/r5noZakNwJWqwnKdEyktS= mzafa@LAPTOP-F33P6UO6
```

Add SSH key

6. Following screen appears, showing the key added

## SSH keys

New SSH key

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.



DevOpsKey

SHA256:06pM4a89wfsT1bM0N55RkijNgKE/+1k5kDNZGLYb1E4

SSH

Added on Jun 22, 2022

Never used — Read/write

Delete

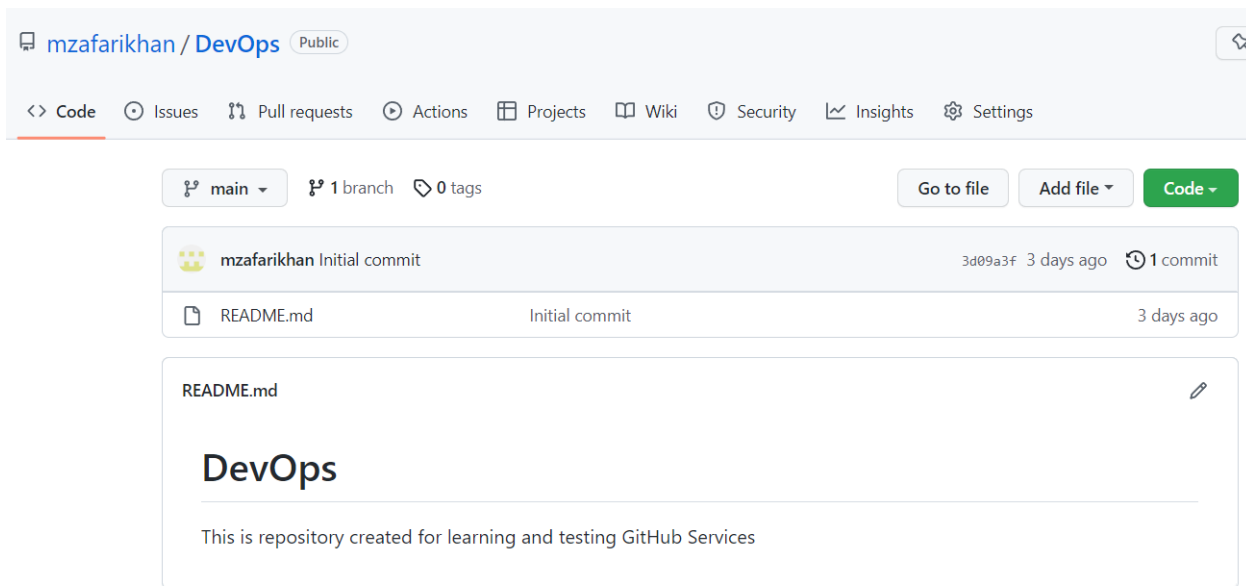
Check out our guide to [generating SSH keys](#) or [troubleshoot common SSH problems](#).

7. Let's authenticate this key, go back to the git terminal, and run following command

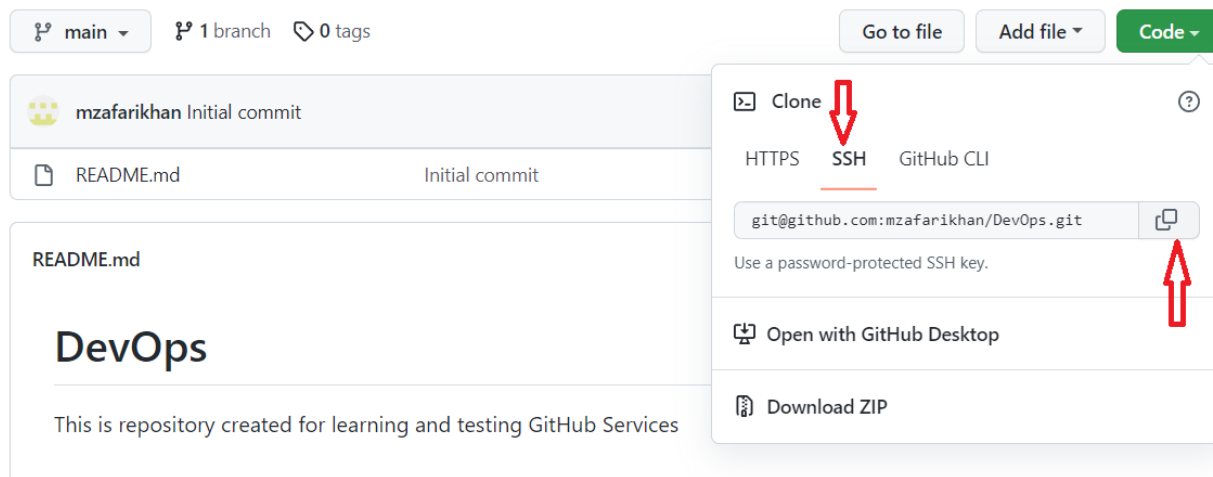
```
MINGW64:/d/DevOps Tools/Lab Manual
mzafa@LAPTOP-F33P6UO6 MINGW64 /d/DevOps Tools/Lab Manual (master)
$ ssh -T git@github.com
Hi mzafarikhan! You've successfully authenticated, but GitHub does not provide
shell access.
mzafa@LAPTOP-F33P6UO6 MINGW64 /d/DevOps Tools/Lab Manual (master)
$
```

Means we have authenticated the pair of keys we have generated. This can be verified from the website as well.

8. Let's go back to the website, go to the repository we have created viz. DevOps, it will be showing only one branch as we didn't do anything here



9. Copying the url of the repository, click on the code button at the right side, click on SSH, and then click on the copy icon to copy the url of the repository. This is url of the remote repository, in which we will be doing remote operations.



10. Adding remote repository, **git remote add origin** command is used to add a remote repository to the local repository

PS: "origin" in the command refers to the current repository, it may be any repository, it just adds a current repository into remote location, specified by url.

```
MINGW64:/d/DevOps Tools/Lab Manual
mzafa@LAPTOP-F33P6U06 MINGW64 /d/DevOps Tools/Lab Manual (master)
$ git remote add origin git@github.com:mzafarikhan/DevOps.git

mzafa@LAPTOP-F33P6U06 MINGW64 /d/DevOps Tools/Lab Manual (master)
$
```

11. Now let's push files from local repository to remote repository

```
MINGW64:/d/DevOps Tools/Lab Manual
mzafa@LAPTOP-F33P6U06 MINGW64 /d/DevOps Tools/Lab Manual (master)
$ git push --all origin
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 12 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (9/9), 1011 bytes | 252.00 KiB/s, done.
Total 9 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
To github.com:mzafarikhan/DevOps.git
```

12. To make a local working copy of the remote repository, clone the repository with **git clone** command

- a. Write **ls** to check the contents of the master repo
- b. Use **git clone** command to get the local working repo of the remote repo

c. Check contents again by ls

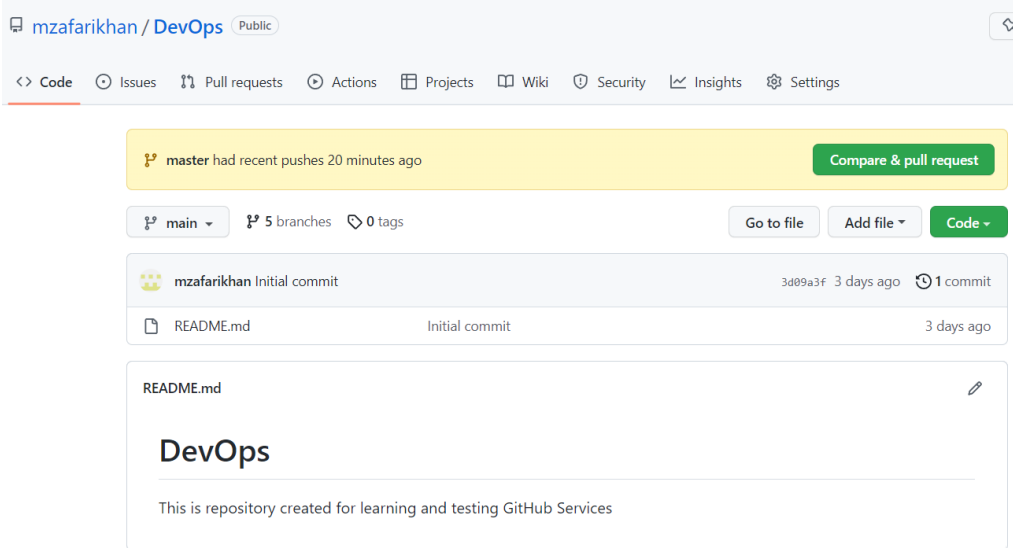
```
MINGW64/d/DevOps Tools/Lab Manual
mzafa@LAPTOP-F33P6U06 MINGW64 /d/DevOps Tools/Lab Manual (master)
$ ls
demo.java demo111.java demo222.java

mzafa@LAPTOP-F33P6U06 MINGW64 /d/DevOps Tools/Lab Manual (master)
$ git clone git@github.com:mzafarikhan/DevOps
Cloning into 'DevOps'...
remote: Enumerating objects: 40, done.
remote: Counting objects: 100% (40/40), done.
remote: Compressing objects: 100% (27/27), done.
remote: Total 40 (delta 6), reused 23 (delta 6), pack-reused 0
Receiving objects: 100% (40/40), 14.86 MiB | 1.06 MiB/s, done.
Resolving deltas: 100% (6/6), done.

mzafa@LAPTOP-F33P6U06 MINGW64 /d/DevOps Tools/Lab Manual (master)
$ ls
DevOps/ demo.java demo111.java demo222.java

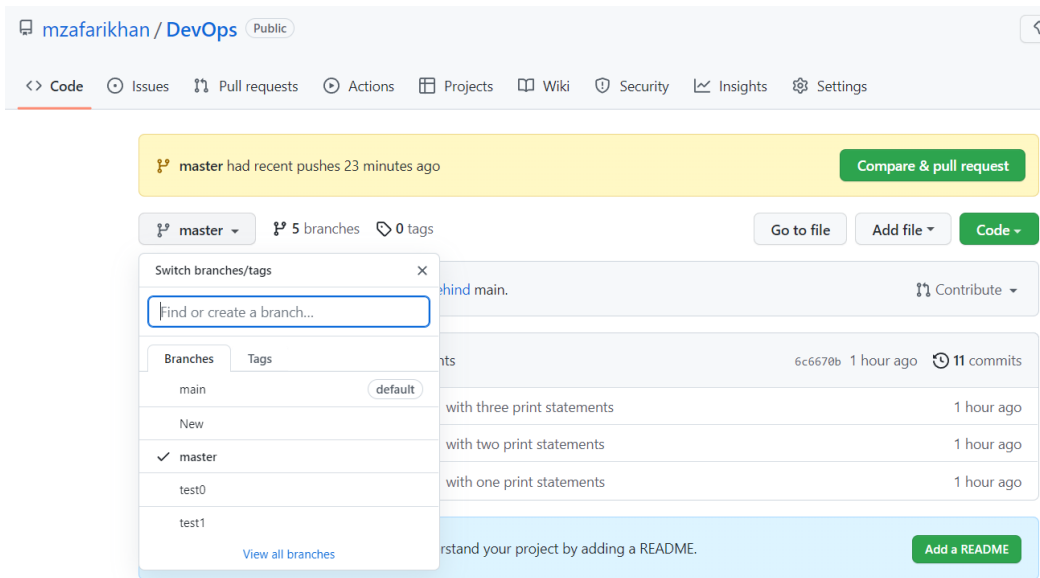
mzafa@LAPTOP-F33P6U06 MINGW64 /d/DevOps Tools/Lab Manual (master)
$ |
```

13. These changes can be observed at GitHub website

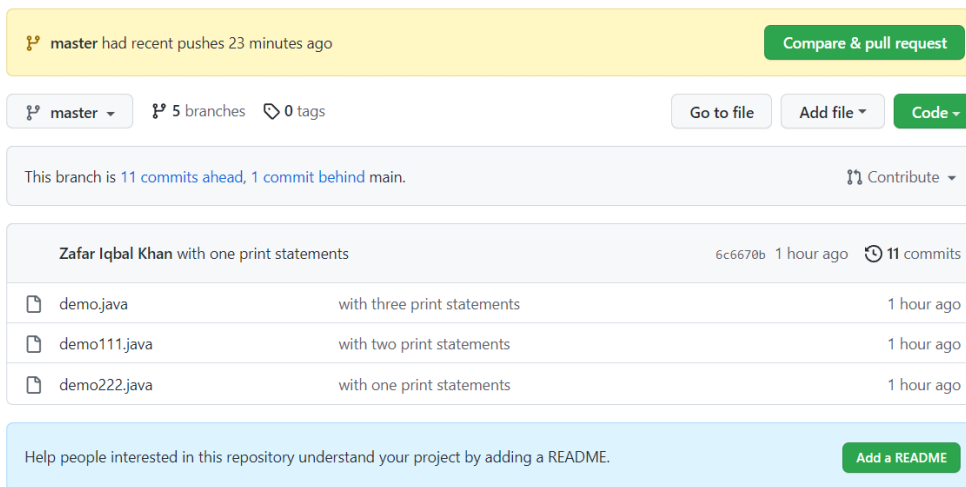


Change branch from *main* to *master*



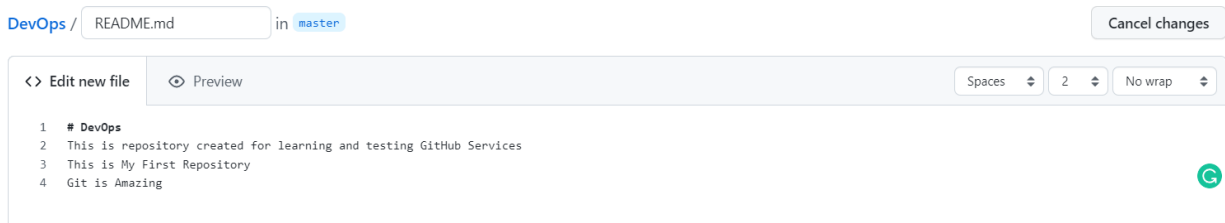


Clearly the commits are visible

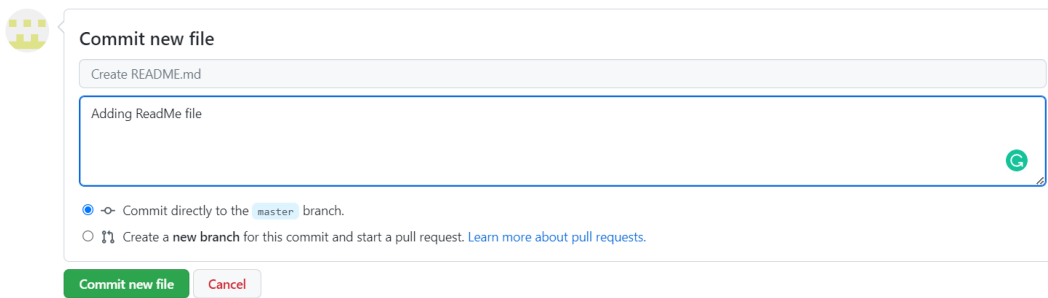


14. Let's make some changes into the remote repository

- a. Click at the **Add a README**
- b. Make some changes into the shown readme file, e.g.



### c. Commit new file



15. Now since we had made some changes in the remote repository (added README file), let's make a pull request, to reflect those changes into the local repository

```
MINGW64:/d/DevOps Tools/Lab Manual
mzafa@LAPTOP-F33P6U06 MINGW64 /d/DevOps Tools/Lab Manual (master)
$ ls
DevOps/ demo.java demo111.java demo222.java

mzafa@LAPTOP-F33P6U06 MINGW64 /d/DevOps Tools/Lab Manual (master)
$ git pull origin master
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 760 bytes | 76.00 KiB/s, done.
From github.com:mzafarikh/DevOps
 * branch                master      -> FETCH_HEAD
 6c6670b..471759a master  -> origin/master
Updating 6c6670b..471759a
Fast-forward
 README.md | 4 ++++
 1 file changed, 4 insertions(+)
 create mode 100644 README.md

mzafa@LAPTOP-F33P6U06 MINGW64 /d/DevOps Tools/Lab Manual (master)
$ ls
DevOps/ README.md demo.java demo111.java demo222.java

mzafa@LAPTOP-F33P6U06 MINGW64 /d/DevOps Tools/Lab Manual (master)
$
```

Clearly, we can see, README.md is now part of local repository.

16. Let's make some changes into demo222.java and subsequently commit and push it.
- At \$ prompt, write notepad demo222.java
  - Add a simple for loop

```
MINGW64:/d/DevOps Tools/Lab Manual
mzafa@LAPTOP-F33P6U06 MINGW64 /d/DevOps Tools/Lab Manual (master)
$ notepad demo222.java

*demo222.java - Notepad
File Edit View

public class demo222
{
    public static void main(String[] args)
    {
        for(int i=0;i<10;i++)
            System.out.println("Welcome to Java");
    }
}

Ln 1, Col 21 100% Windows (CRLF) UTF-8
```

c. Save the file

17. Add this file to the staging area, -u can be used to update only file that has changed

```
MINGW64:/d/DevOps Tools/Lab Manual
mzafa@LAPTOP-F33P6U06 MINGW64 /d/DevOps Tools/Lab Manual (master)
$ git add -u

mzafa@LAPTOP-F33P6U06 MINGW64 /d/DevOps Tools/Lab Manual (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   demo222.java

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

18. Now commit this file with message, "for loop added"

```
MINGW64:/d/DevOps Tools/Lab Manual
mzafa@LAPTOP-F33P6U06 MINGW64 /d/DevOps Tools/Lab Manual (master)
$ git commit demo222.java -m "for loop added"
[master 1ff14c6] for loop added
1 file changed, 2 insertions(+), 1 deletion(-)

mzafa@LAPTOP-F33P6U06 MINGW64 /d/DevOps Tools/Lab Manual (master)
$
```

19. Let's suppose after committing this change, user give their feedbacks, and we realize that we need to revert back to previous version of the code, then do the following steps
- Write **cat demo222.java**, it will print contents of the demo222.java file on the console
  - Run the code, \$ **git log**
  - Use arrow keys to navigate to the end of the file
  - Find the commit message "with one print statement"
  - Copy first 8 characters from the long string written just next to the commit

```
MINGW64:/d/DevOps Tools/Lab Manual
mzafa@LAPTOP-F33P6U06 MINGW64 /d/DevOps Tools/Lab Manual (master)
$ cat demo222.java
public class demo222
{
    public static void main(String[] args)
    {
        for(int i=0;i<10;i++)
            System.out.println("welcome to Java");
    }
}

mzafa@LAPTOP-F33P6U06 MINGW64 /d/DevOps Tools/Lab Manual (master)
$ git log
commit 1ff14c60e9acc11bc5e659e735d0ccc3907ff935 (HEAD -> master)
Author: Zafar Iqbal Khan <zkh@psu.edu.sa>
Date: Sat Jun 25 23:44:55 2022 +0530

    for loop added

commit 471759a87db6893b86276ff5793d436665954ca0 (origin/master)
Author: mzafarikhan <30715764+mzafarikhan@users.noreply.github.com>
Date: Sat Jun 25 23:29:29 2022 +0530

    Create README.md

    Adding ReadMe file

commit 6c6670b9f8e9b7babcef981b70ddc4188168338c
Author: Zafar Iqbal Khan <zkh@psu.edu.sa>
Date: Sat Jun 25 22:21:15 2022 +0530

    with one print statements
```

20. Write `git checkout <8 characters> <filename>` and subsequently `cat demo222.java` to verify the changes

```
MINGW64:/d/DevOps Tools/Lab Manual
mzafa@LAPTOP-F33P6U06 MINGW64 /d/DevOps Tools/Lab Manual (master)
$ git checkout 6c6670b9 demo222.java
Updated 1 path from 9410d33

mzafa@LAPTOP-F33P6U06 MINGW64 /d/DevOps Tools/Lab Manual (master)
$ cat demo222.java
public class DEMO
{
    public static void main(String[] args)
    {
        System.out.println("welcome to Java");
    }
}

mzafa@LAPTOP-F33P6U06 MINGW64 /d/DevOps Tools/Lab Manual (master)
$ |
```

Clearly, there is no more for loop.

**This is known as version restore.**