Full Stack Development
Lab Management 2

more git commands

1. Lab objectives

This lab is intended for you to get a solid understanding of git basics

2. Staging and unstaging

After committing a file, the file in the working directory is being tracked by git. Any changes to it are noted. To commit those changes, first the changes have to be staged. This is what the git add command does. However, if you want to change your mind, you can unstage the changes by removing them from the staging area. This does not change your working file.

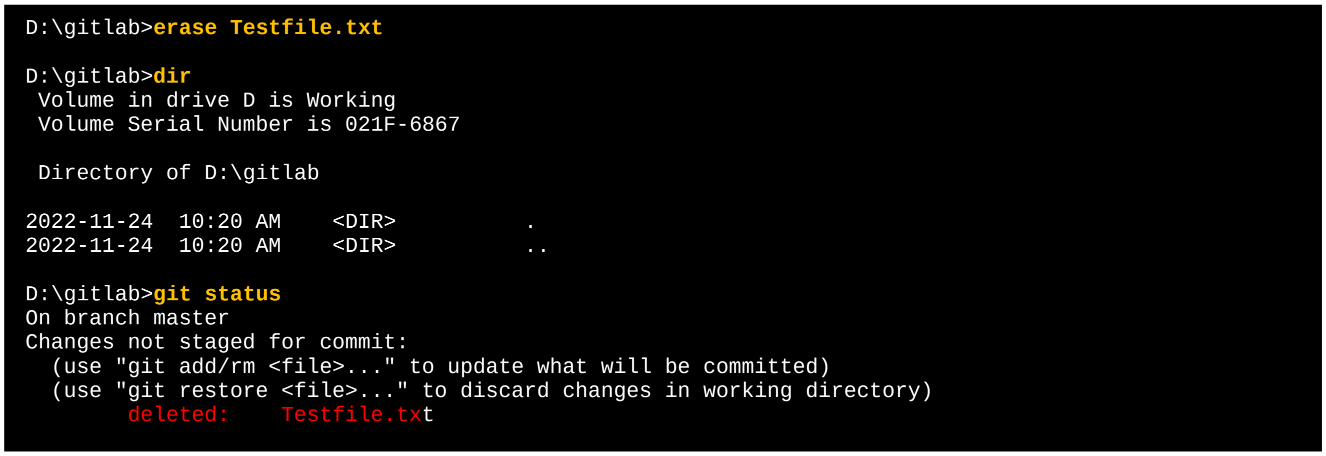
Make some changes to your test file that you committed earlier and run git status. Add the changes to the staging area and rerun git status

```
D:\gitlab>git add Testfile.txt

D:\gitlab>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   Testfile.txt
```

Now unstage the changes.

```
D:\gitlab>git restore --staged Testfile.txt

D:\gitlab>git s
On branch master
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:   Testfile.txt
```

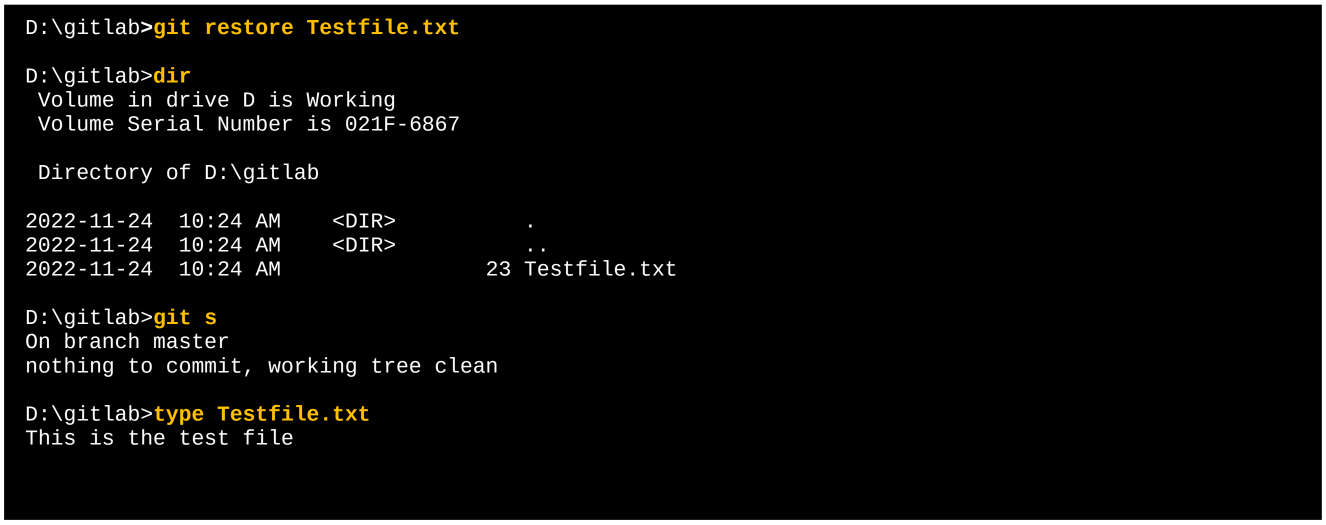
It is possible to revert a file in the working directory to the version in the repository. Delete your test file and run git status



```
D:\gitlab>erase Testfile.txt

D:\gitlab>dir
 Volume in drive D is Working
 Volume Serial Number is 021F-6867

 Directory of D:\gitlab

2022-11-24  10:20 AM    <DIR>          .
2022-11-24  10:20 AM    <DIR>          ..

D:\gitlab>git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    Testfile.txt
```

Git is advising you that you have removed a file it is tracking. To delete the file in the next commit, you can use the git add or git rm command. However, if you deleted the rile by accident, then you can get the copy from the repository with git restore.

However, notice that what was restored was the version that was committed, not what you were working on before you deleted the file.



```
D:\gitlab>git restore Testfile.txt

D:\gitlab>dir
 Volume in drive D is Working
 Volume Serial Number is 021F-6867

 Directory of D:\gitlab

2022-11-24  10:24 AM    <DIR>          .
2022-11-24  10:24 AM    <DIR>          ..
2022-11-24  10:24 AM                23 Testfile.txt

D:\gitlab>git s
On branch master
nothing to commit, working tree clean

D:\gitlab>type Testfile.txt
This is the test file
```

3. Tracking commits and changes

Make several changes and commits to your test file.

```
D:\gitlab>git ll
* 1e76e5c (HEAD -> master) Third commit
* 4a20398 second change
* bdac64d first change
* 805993f Inital Commit
```

Note that the commits are shown starting with the most recent. The HEAD pointer is showing you what is currently being displayed in the working directory – the third commit on the master branch.

Make a change to your test file in the working directory but do not stage the changes. To see how your current working file is different than the one in the repository, use the git diff command

```
D:\gitlab>git diff Testfile.txt
diff --git a/Testfile.txt b/Testfile.txt
index 35fe996..b407f1a 100644
--- a/Testfile.txt
+++ b/Testfile.txt
@@ -1,3 +1,3 @@
 This is the test file
 Change 3
-Change 2
\ No newline at end of file
+Change 4
\ No newline at end of file
```

4. Ignoring files

The .gitignore file lists all the file types that should not be tracked by git. Suppose for example that you want to ignore all log files that are generated by an application. Create a file call app.log and run git status .

```
D:\gitlab>dir

 Directory of D:\gitlab

2022-11-24  11:09 AM    <DIR>          .
2022-11-24  11:09 AM    <DIR>          ..
2022-11-24  11:09 AM                18 app.log
2022-11-24  11:09 AM                41 Testfile.txt

D:\gitlab>git s
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        app.log

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

To do this, create a file called **.gitignore** and place the correct file pattern in it. Be sure to add this file to the repository as well. The ignore file is correctly thought of as part of the repository, not your current working directory. Note that the log file is now ignored by git

```
D:\gitlab>type .gitignore
*.log

D:\gitlab>git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
```

5. Resetting commits

After a bad commit that has messed up your repository, you can direct git to go back and reset to an earlier commit and ignore all the commits since that earlier commit.

Create a bad commit by deleting the .ignore file and adding the log file.

```
D:\gitlab>git rm .gitignore
rm '.gitignore'

D:\gitlab>git add .

D:\gitlab>git s
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    .gitignore
        new file:   app.log

D:\gitlab>git commit -m "Bad commit"
[master f5b7d4f] Bad commit
 2 files changed, 1 insertion(+), 1 deletion(-)
 delete mode 100644 .gitignore
 create mode 100644 app.log

D:\gitlab>git ll
* f5b7d4f (HEAD -> master) Bad commit
* 30226bb Add gitignore
* 1e76e5c Third commit
* 4a20398 second change
* bdac64d first change
* 805993f Inital Commit
```

To make it as if that last commit never happened. You reference the commit you want to go back to by referencing its hash. Note that git is also telling you that the deleting of the .gitignore file is not staged.

```
D:\gitlab>git ll
* f5b7d4f (HEAD -> master) Bad commit
* 30226bb Add gitignore
* 1e76e5c Third commit
* 4a20398 second change
* bdac64d first change
* 805993f Inital Commit

D:\gitlab>git reset 30226bb
Unstaged changes after reset:
D       .gitignore
```

Looking at git status shows the changes that were staged for the bad commit. Now you can just unstage those and the working directory will be good to go

```
D:\gitlab>git s
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    .gitignore

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

D:\gitlab>git restore .gitignore

D:\gitlab>git s
On branch master
nothing to commit, working tree clean

D:\gitlab>dir

2022-11-24  11:36 AM    <DIR>          .
2022-11-24  11:36 AM    <DIR>          ..
2022-11-24  11:36 AM                 7 .gitignore
2022-11-24  11:09 AM                18 app.log
2022-11-24  11:09 AM                41 Testfile.txt
```

End Lab