Topic : Git Commands
Name : Talha
Surname : Demir
Submitted to : Ali Cihan Keleş
Date : 11.04.2025
Section : 1

Purpose: To help us understand Git Bash commands and what they do.

B: Before The Problem
A: After The Problem
SS: Screen Shot
SX: Step X

==========

Steps	Procedure	Observations
1	Creating GitHub account. SS	Go to GitHub.com. On the page, click “Sign Up” and enter your e-mail address, and create a password.
2	Creating our first repository. S1 & S2	In your profile click the “Repositories” tab. Then click on the “New” Button. Enter your repository name then click “Create repository”.
3	Copying our first created repository’s HTTPS address. SS	Click the “Copy” button in the top-right corner then copy the HTTPS address of your repository. Then open Git Bash and run the next provided commands.
4	Cloning our repository to our local system. S1 & S2	By typing given commands we can clone our repository and push to GitHub by typing given commands. I already logged in with my GitHub account so no password was required. Anyways by typing “**git clone repoLink**” we can successfully clone the repository.
5	Instead of cloning we will start with a local repository. SS	This is an alternative way to create a Git repository. With “**git init**” command we can make the file repository.
6	Forking other users repositories to our local system. SS	We can again use “**git clone**” command to fork other users repositories.
7	Synchronization of local and remote Repositories. SS & SS & RESULT	What we want here is synchronize remote repositories to our local repository. As you can see there is a missing file in our local repository we can simply use **“git pull origin main”** command to synchronize and you might be asking “*what is the difference between ‘git add’ and ‘git commit*’”. The “**git add**” command adds changes to the staging area but the changes are not permanent on the other hand “**git commit**” command is making these changes permanent. Basically it saves everything. We will learn more in the next steps.
8	Files followed by Git . S1 / S2 / S3	At this stage we are learning “**git commit -am**” command. By typing “git ls-files” you can see which files are followed by git. As you can see at “**S2**” our new code.py file is not followed by git. So we have to use “**git add**” command first.

9	Staging area in Git. S1 / S2	For this step we are understanding the basic mechanics of "**git add**". Firstly I added a line to the notepad and then sent it to the staging area, at this point if we type git status we can see it is at staging area. But if we add new line to the notepad and then type git status again we can see the first change is on staging are but our new line is not on the staging area yet. If we commit at this point only the first changes will be committed as we can see at screenshots.
10	Undo changes made in Git. S1 / S4	We can simply use "**git restore --staged**" command to unstage the file but this command doesn't change the file, but if we type "**git restore**" this will undo all changes at file so file will be reverted to the previous default form. (As we can see the new line that we added has gone. S2 → S3)
11	Renaming files in Git. S1 / S2	With "**git mv ./**" command we can change the names of file and this will be automatically added to staging area and ready for commits. But if we use manual method Git cannot identify the filename change. It will say "file has been deleted and an existing new file has been added". To solve this we can simply use "**git add -A**" command to add all changes to staging area and this command is only for **CMD**. But we can use "**cmd //c**" to use this command at Git Bash.
12	Deleting files in Git. S1 /	We can delete files by using traditional way but this method is not recommended so we will use a more general method.
12.1	A problem SS → SOVLED	As we can see I accidentally deleted the whole repository by typing "**./f**". However we can restore all of them since we didn't commit the changes.

Problems & Log;

These commands are not working on Git Bash however we can use some other commands instead of using PowerShell commands. You can find all of them at "**solution**" part.

Command	Problem	Solution
del "fileName.txt" [1]	There is no such a command like del at Git Bash. B	We can use the **"git rm "fileName.txt"** to delete files. A (or we can just add cmd //c to the begining)
rmdir /s "repoName" [3]	The problem is this command is not avalible on Git Bash so we need to use different commands. B	To fix this issue we need to delete the files inside our repositories with the **"rm -rf repoName"** command. (**r** -> Deletes directories and their contents recursively. **f** -> Forces deletion without asking for confirmation. A
tree /f	The command is not working on Git Bash but it's working on Windows CMD. B	However the alternatives for this command on GitBash is **"cmd //c tree"**. A
idle "name.py"	The command is not working on Git Bash. B	We have multiple solutions for this but simply we can use this command **"py -m idlelib talhaDmeirDone.py"**. A
ren "file.txt" "selam.txt" [2]	The command is not working on Git Bash B	We can use the **"git mv oldName.txt newName.txt"** command to fix this issue. A
cls	The command is not working on Git Bash	Instead we can just use "**clear**" command to clear command prompt.

LOG	Rename Git Commands And View Commit History in Git.	With log commands we can see the changes that has been commited etc. If we use "**git help log**" command it will send us to the website and we can see all the commands that we need. (S1)
1.1	git show	Shows the last commit with fully detailed. (S1)
1.2	git log	Shows the commit history with ID's, date and more. (S1)
1.3	git diff	Shows changes that haven't been committed yet. I don't have anything to commit so it shows nothing. (S1)
2.1	git darwin S1 → S2	Instead of typing "**git log –all –graph –decorate –oneline**" we can write that code more shorter with assigning configs. It's pretty much doing the same thing.

Conclusion:

In summary, we learned how to use Git commands, like adding a file, committing and restoring. By understanding these commands, we can avoid common errors and fix our mistakes easily and this helps us to work with Git more smoothly. Knowing the basics of the Git commands will save a lot of time for us, like assigning commands and we can use Git more efficiently.

References

[1] [Online]. Available: https://stackoverflow.com/questions/2047465/how-do-i-delete-a-file-from-a-git-repository.

[2] [Online]. Available: https://stackoverflow.com/questions/59998804/renaming-a-file-in-git-bash-on-windows.

[3] [Online]. Available: https://stackoverflow.com/questions/1213430/how-can-i-fully-delete-a-git-repository-created-with-init.