Demo’s

# Steps for creating a repo and pull push process

1. Create a directory – learngit
2. Go inside the dir
3. Initialize the repo as git repo
4. Echo Welcome to Git Training > git.txt
5. Check the status
6. Git.txt in untracked stage also referred as red zone
7. Move the file to staging area
8. Check the status
9. File should be in green zone – the staged area.
10. Now commit the file with a message
11. Check the status
12. Modify the file
13. Run the status
14. File should show modified and in red zone
15. Move to staging area
16. Run the status
17. Run the commit
18. Check the status – Everything is clean in the status
19. Modify the file
20. Move to staging area
21. Edit the file
22. Run the status - 2 copies visible 1 in red and 1 in green
23. Move the red file to staged file
24. Run the status – now files are merged to one copy of two changes.
25. Commit the file
26. Go to github
27. Create a new repo
28. Copy the SSH URL
29. Go to git terminal
30. Connect the url of repo as remote repo with the name origin
31. Verify the remote connection
32. Check the status
33. Check if any new updates available in github
34. Push the local changes to remote repo
35. Verify the updates on github if they are visible now
36. If the updates are visible then we have successfully completed the process.
37. Go to github
38. Create a new file and commit the change in github
39. Go to local
40. Fetch the changes
41. Git status – it should reflect that the remote is 1 commit ahead of local
42. Pull the changes
43. Git status – everything is clean now.

# Remote + Local

1. Create a remote repo
2. Clone it to local
3. Add a file and Commit the file
4. Check the log
5. Run the status
6. Add another file and commit the change
7. Check the status
8. Check the remote repo connection
9. Fetch
10. Pull
11. Push
12. Check the log
13. Everything is updated to both the repo
14. Got to github repo and check the changes done from local

# Steps to setup SSH

1. Go to terminal window
2. Run the command cd ~/.ssh
3. Now you are inside the .ssh directory of your system.
4. Run the next command - ssh-keygen -t rsa -C “[email-id](mailto:simplilearngithubtraining@gmail.com)”
5. The above command will generate two ssh keys – Public and Private
6. Go to your system home directory and look for .ssh directory and there you will find one .pub extension file.
7. Open the file and you will see the public key.
8. Copy the public key as it is.
9. Open the GitHub setting page >> go to ssh keys tab >> click on new ssh key
10. Add the key and save it.
11. Now your local machine identity is verified by your GitHub account successfully.
12. Verify the connection by running the command ssh -T [git@github.com](mailto:git@github.com).
13. It will ask Are you sure you want to continue connecting (yes/no)? just say yes here.
14. You should receive a response as:

Hi *username*! You've successfully authenticated, but GitHub does not provide shell access.

# Steps for cloning and log check

1. Clone the following [URL](https://github.com/tabhresh/Selenium_Maven_Jenkins_Git) of my repo
2. Go to the repo and run the terminal
3. Check the status of the repo
4. Run the git log commands to see the various views of the log
5. Run and test your various log commands on this repo to get good view and understand the log commands.

# Steps for difference check

1. Go to learngit project
2. Check status
3. Create a new file – temp.txt
4. Move it to staging area
5. Commit the change
6. Check status
7. Now edit the temp.txt
8. Check the status
9. File should be in modified stage/unstaged area
10. Run the diff command
11. You can view the difference between the committed file and unstaged file
12. Now move the file to staging area
13. Run the diff command
14. You will see the difference between the committed file and staging file
15. Commit the file
16. Run the oneline log and take the latest two commit id’s
17. Run the diff command to view the difference between the two commits
18. You will be able to see the differences between two committed stage of the same file.

# Steps for Rename

1. Go to learngit repo
2. Run status
3. Take two files
4. Rename one file using git command
5. Check the status
6. File in staging area and git tracking it to be renamed
7. Now edit another file from UI side of the lab
8. Go to terminal
9. Run status
10. Git shows the recently renamed file as deleted with old name and new file with new name
11. Move all the file to staging area
12. Now run the status
13. All files are in staging area with the highlighter as renamed.
14. Commit the changes.

# Steps for Alias

1. Go to learngit repo
2. Run the status
3. Check the global configuration list
4. Now add the alias for oneline log command
5. Now add the alias for status command
6. Now run the global configuration list again.
7. It will show you the alias registered in the list.
8. Now verify the alias.

# Steps for Amend

1. Go to learngit repo
2. Check the status
3. Now create a file – amend.txt
4. Add a heading with mistakes in the text “ We are teaching git amend command”
5. Now check the status
6. Move the file to staging area
7. Commit the changes
8. Now you want to rectify the change with the awareness of others
9. Check the one line log
10. Now correct the text as - “ We are learning git amend command”
11. Go to repo and check the status
12. File is visible in unstaged stage
13. Move the file to staged
14. Commit the changes but with the help of using the amend command
15. Check the one line log
16. Your log will show the same no of commit, but the last commit will be updated with the correct information.

# Step for direct commit

1. Check the status
2. Modify any file from the repo
3. Check the status
4. File is currently in modified stage
5. Run the commit command
6. Check the log
7. Commit is done

# Steps for ignore

1. Configure the gitignore using configuration command to be done only one time in the same machine
2. Go to learngit
3. Run the status
4. Create a file with any name but with an extension as .gitignore
5. Add two files with test1.ppt and test2.xlsx
6. Run status
7. Files are in red zone i.e. untracked file stage
8. Now edit the .gitignore file and add the file extensions?
9. Run the status
10. Now files not visible in the status
11. Now mark the commit of changes in .gitignore file
12. Run the status
13. Now the .gitignore file is tracked but the files mentioned inside the file are not been tracked

# Steps for Checkout

1. Git status
2. Git checkout to remote repo and main branch
3. Go to file explorer and view the file an

# Steps for tags

1. Go to learngit repo
2. Check the status
3. Check the oneliner log
4. Find out which commit is to be added with a tag
5. Run the lightweight tag command with the commit id
6. Now run the oneliner log again
7. The tag is reflected with a yellow color shade
8. Find out one more commit id which is to be added with a tag
9. Run the Annotated tag command with the commit id and the message
10. Run the tag list command
11. Tags with messages will be displayed
12. Let’s delete one of the tag using tag deleting command
13. Run the oneliner log to verify if the tag is visible on the log
14. Run the tag list command to verify if we have successfully deleted the tag.

# Steps for Stashing

1. Go to learngit repo
2. Create a new file – stash1.txt
3. Add a heading in the file
4. Commit the file
5. Create a new file – stash2.txt
6. Add a heading in the file
7. Run the status
8. File is visible in untracked stage
9. Now run the stash command for a file
10. Git will reject it
11. Now add the stash2.txt to staging area
12. Now run the stash command for a file again
13. Your file is stashed
14. Now edit stash1.txt
15. Run the status
16. File in unstaged zone
17. Now run the stash command
18. Your file is stashed
19. Run the status
20. Now check the files in stashing space
21. Get the stash2.txt back to working directory using the pop command
22. Make the pending changes in the file and commit the file
23. Now get the stash1.txt popped out from stashing space
24. Make the pending changes and commit the file.
25. Now edit the stash2.txt
26. Run the status
27. File is in unstaged area
28. Run the stash command
29. See the stash list
30. Run the delete stash command
31. See the stash list
32. The modified changes and the stashed file are deleted
33. File is still in the repo but with old content.

# Steps for private GitHub repo

1. Go to github
2. Create a new private repo
3. Go to repo settings page
4. On the left Click on Manage access
5. Click on invite a collaborator
6. Type search and select the username of the contributor
7. Click on add
8. Now go to the contributor’s email id to accept the invitation
9. Once accepted you can access the repo

# Steps for Clean

1. Go to learngit repo
2. Run the status
3. Add a new file
4. Run the status
5. File in the untracked area
6. Now run the command to remove the file
7. Run the status
8. File removed
9. Add a directory and a file inside the directory
10. Add a file outside the directory
11. Now run the clean command to remove the directory and files
12. Run the status file and directory is removed successfully
13. Add a directory and a file inside the directory
14. Add a file outside the directory
15. Now run the clean command to remove the directory and files along with ignored files
16. Run the status file and directory is removed successfully

# Steps for RM

1. Go to learngit repo
2. Run status
3. Add a new file testrm1.txt
4. Commit the change
5. Run the status
6. Run the rm command to delete testrm1.txt
7. File is deleted
8. Run the status
9. Now add a new file testrm2.txt and commit it
10. Now run the status
11. Now run the command for removing the file but without deleting it from the repo
12. Run the status
13. Same file now visible in unstaged and staged area
14. Just commit the file from staging area keeping the unstaged file as it is
15. Run the status
16. Record of deletion moved to the commit
17. New file is in untracked area
18. Add the same file name testrm2.txt in the .gitignore
19. Now run the status
20. Now the file is not visible instead .gitignore is visible as modified
21. Run the commit for .gitignore
22. Run the status
23. Now the file is never going to be a part of the repo further.
24. History of that file is over.

# Steps for Reset

1. Go to learngit repo
2. Add a new file reset.txt
3. Add a heading – “Welcome to Git and Github Course”
4. Check the status
5. Commit the changes
6. Modify reset.txt by adding second line – “Instructor-led Learning!”
7. Check the status
8. Commit the changes
9. Now you have two commits for the same file
10. Add another commit with line 3 – “By Abhresh”
11. Run the mixed reset command and go to 1st commit
12. Run the status – File should be in unstaged area
13. Move the file back to the committed stage by committing the file
14. Run the soft reset command
15. Run the status – File should be in staged area
16. Move the file back to the committed stage by committing the file
17. Run the hard reset command
18. Run the status – File is deleted
19. Now commit the changes of deletion of the file.

# Steps for reset part 2

1. Go to learngit repo
2. Run the status
3. Run the oneliner log
4. Copy any old commit where a new file was created
5. Run the reset command to move the repo to that commit
6. This will reset the history to that commit and now all the newer commits are deleted.

# Steps for revert command

1. Make a new dir – revertgit
2. Add a new file – a.txt
3. Commit the change with a msg as m1
4. Add a new file – b.txt
5. Commit the change with a msg as m2
6. Add a new file – c.txt
7. Commit the change with a msg as m3
8. Run the log
9. Run the revert with commit id of 3rd commit
10. Your c.txt gets deleted and a new commit is created for reverting to the stage of 2nd commit completion.
11. Run the log
12. The repository is back to the state of 2nd commit

# Steps for revert command part2

1. To bring back the latest commit back to the repo
2. Run the log
3. Run the reset to commit 3
4. But the file is not back into the repo
5. Run the checkout c.txt
6. This will bring the file back to our repo.

# Steps to Reset for old records using Reflog clear

1. Run the Reflog
2. Copy the commitid
3. Checkout weather that commit is correct
4. Run the reset command with the expected commit
5. If there is any file which was deleted, then system will show you the file name with D mark
6. Recover the file using checkout and file name command

# Steps for Forking a repo

1. Go to the login environment of the GitHub
2. Navigate to the respective repository which is to be forked
3. Find out the fork button on the repository page and click on it
4. Now you will be navigated to a repository which is a forked copy of the project

# Steps for Creating a pull request

1. Make some changes in the forked copy
2. Commit the same
3. Go to Pull request menu
4. Select the repo from where you want to create a pull request
5. Now click on create merge request button
6. Once this process is done wait for the repository owner to accept or reject your request
7. Based on the decision the acceptance or rejection will be visible

# Steps to pull the updates from the main repo to forked repo

1. Got to forked repo of our account on GitLab
2. Click on pull requests
3. Click on button of new pull request
4. Click on the base repo drop down
5. Change it to your repo
6. Page reloads
7. Click on compare across forks
8. Now you will be able to select base as your and head as the original repo
9. Once the page refreshes >> you will see the updates to be fetched from the original repo
10. Click on create pull request and pull the changes to your repo.

# Steps for Branching

1. Create a new repo
2. Add a new file and commit the changes
3. Run the command to see the list of branches
4. Only main branch visible
5. Create a new branch with the name develop
6. Move inside the develop branch
7. Create a new branch feature1
8. Create a new branch feature2
9. Run the command to see the list of branches
10. 4 branches with their names visible
11. Now delete the feature1 branch

# Steps for branching practice

1. Create a new Branch.
2. Now switch to the main branch and push the updates to the remote repo.
3. Now switch back to the new branch
4. Make the necessary changes on the file.
5. Now add and commit the changes within new branch
6. Check the status
7. Check the log
8. Push the changes to the remote repo
9. Git status
10. Once the manager confirms the changes are good to move to the final copy
11. We will now merge the new branch to the main branch

# Steps for project work

1. Make directory
2. Initialize the repo
3. Add a new file to the repo and commit it
4. Now run the command to see the branches
5. You will see only main branch
6. Now run the command to create a new branch develop
7. Move inside the develop branch
8. Now create a new branch dev1 and dev2
9. Now go to dev1
10. Add a file with the name as loginfeature.txt with Heading as “This is a login feature code”
11. Now commit the changes
12. Check the status
13. Now move to dev2
14. Add a file with the name as logoutfeature.txt with Heading as “This is a logout feature code”
15. Now commit the changes
16. Check the status
17. Now move to develop branch
18. Run the status
19. Now rebase the dev1 into develop
20. Run the log
21. All the commits of dev1 are visible in the develop branch
22. Now merge the changes from dev2 branch to develop
23. Check the status
24. Run the log which will show all the commits created in both the branches
25. Now go to main
26. Run the merge develop branch command
27. And now see the log which will show all your commits into the main branch.

# Steps for Git Workflow

1. Create a GitHub Repository with a name as GitProject do not forget to add default Redme.md file.
2. Clone the repo into your local machine
3. Move inside the directory
4. Now check the status
5. If everything is clear perform the following steps
6. Create a new branch as develop,
7. Checkout to develop branch
8. Create dev1 and dev2 branch from develop branch
9. Now update the complete project to the GitHub repo where all the branches are also updated
10. Now go to GitHub and verify the updates
11. Now go to local machines of dev1 and dev2 and clone the repo with respective branch names
12. Now start working on your dev1/dev2 branch and make the necessary development and push the updated to remote branches
13. All the changes to be pushed to your own repo branch only

# Merge

1. Create a repo
2. Add readme.txt and commit it as “m1”
3. Modify the readme.txt and commit it as “m2”
4. Create a branch as “feature”
5. Checkout to feature branch
6. Add feature.txt and commit it as “f1”
7. Modify the feature.txt and commit it as “f2”
8. Go to main
9. Run merge feature

# Steps with recursive merge without conflict

1. Create a repo
2. Add readme.txt and commit it as “m1”
3. Modify the readme.txt and commit it as “m2”
4. Create a branch as “feature”
5. Checkout to feature branch
6. Add feature.txt and commit it as “f1”
7. Modify the feature.txt and commit it as “f2”
8. Now checkout main add new file and commit it as “m3”
9. Git merge feature
10. Vim tool will open and it will ask you for a merge commit message
11. Once done run the log and now you can see the merge commit.

# Steps for Merge with conflict

1. Create a repo
2. Add readme.txt and commit it as “m1”
3. Modify the readme.txt and commit it as “m2”
4. Create a branch as “feature”
5. Checkout to feature branch
6. Add feature.txt and commit it as “f1”
7. Modify the feature.txt and commit it as “f2”
8. Now checkout main and edit the readme.txt and commit it as “m3”
9. Checkout to feature
10. Edit the readme.txt and make some major changes and commit it as “f3”
11. Checkout to main
12. Git merge feature
13. Now you will see a conflict response
14. Now open the readme.txt file and you will see the conflict response
15. Edit the file with the finalized content
16. Go to git bash and run status and commit the change.

# Steps for Rebase

1. Let’s create a new project
2. Now let’s create a new file – abhi.txt
3. Git init
4. Git status
5. Git add
6. Git commit – m “main commit 1”
7. Now make some changes
8. Git status
9. Git add
10. Git commit – m “main commit 2”
11. Create a new feature branch
12. Move to the new feature branch
13. Git log --oneline
14. Git checkout main
15. Git log --oneline
16. Both the branches are at same level of commits
17. Now create a new commit in main branch
18. Git commit -n “main commit 3”
19. Git log –oneline
20. Git checkout feature branch
21. Git log –oneline
22. Differences are there in both the branches
23. Add a new file into the feature branch
24. Commit two changes “feature commit 1” and “feature commit 2”
25. Git log –oneline
26. Git rebase main
27. Git log –oneline
28. Git checkout main
29. Git merge feature
30. Git log –oneline
31. Now the history will look like this

# Steps for Cherrypick

1. Let’s create a new project
2. Create a new file main.txt
3. Add come content into it
4. Add the file to staging
5. Commit the file as M1
6. Edit the file
7. Add and commit the changes as M2
8. Now create a new branch Dev and checkout
9. Add a new file login.txt and commit as f1
10. Now edit the file and commit as f2
11. Now checkout to main and add a new file as dashboard.txt
12. Add and Commit the changes as M3
13. Now edit the main.txt and commit it as M4
14. Now checkout to Dev
15. You want M3 commit in dev but you don’t want M4 commit.
16. Now using the Cherrypic command bring the M3 commit
17. Now you can see the log and the files showing data of M3 commit in dev as well as master branch
18. Now make the changes to login.txt and commit it as f3
19. Now checkout to master branch
20. Run the merge command
21. Check the log you will see M3 visible twice.

# Steps for practice

1. Fork a remote repo
2. Go to your forked repo
3. Copy the forked repo url
4. Go to your local machine
5. Clone he remote repo
6. Check the no of branches available
7. Create your branch with your name
8. Add a new yourname.txt file to it
9. Commit the changes
10. Push the changes to repo
11. Push the branches as well as changes inside your branch
12. Check the updates into your repo
13. Now create a merge request for the changes made by you
14. I will accept the same and we will merge the data into one project

Steps for practice:

1. Go to Gitlab
2. Create a repo named as nikasio\_teamA/B/C
3. Invite 5 members of your team with proper rights to them
4. Now all 5 will go to gitlab and take ssh URL
   1. Team A
      1. Surendera
      2. Dakshayini
      3. Parikshith
      4. Sushmitha
      5. Mythra
   2. Team B
      1. Rama
      2. Mohanraj
      3. Abhishek
      4. Rakshitha
      5. Eshwara
   3. Team C
      1. Megha
      2. Mayur
      3. Vijaylaxmi
      4. Shreekar
      5. Rakshit
5. Clone the repo to local
6. Now check the status
7. Create a new file with yourname.txt
8. Enter Your Name, Designation, Work Role as an employee, Total years with Utthunga.
9. Write Q:- What is best about Utthunga work culture?
10. Write A:- Mention your answer in short.
11. Now save the file
12. Check the status
13. Commit the change with your message as “Your Name”
14. Following the process do push the changes to the remote repo.

Demo Project

1. Fork the project
2. Pull the project to Git
3. Project consists of only Readme file
4. Create a branch – YourFirstName\_Master - Here this the master branch for your project
5. Checkout to that branch
6. Now create a branch - Develop – is for Stable release and is created from master branch
7. Now move to develop branch
8. Create two feature branch – yourname1 & yourname2 – is to work on various features and once stable merge them to Develop branch
9. Now move to the yourname1 and Create an Tutorial.html file
10. Add Heading as – Welcome to Git and GitHub Training
11. Commit the changes
12. Now move to yourname2 and Create an Tutorial.html file
13. Add Heading as – From Nikasio!
14. Commit the changes
15. Now merge the changes to Develop Branch
16. Now go to your Master Branch and create a HotFix branch
17. Move to HotFix branch and create a new File Tutorial.html
18. Commit the changes.
19. Merge the changes to your Master Branch
20. Now merge the changes of your develop branch to your master branch using rebase
21. Push all the changes to remote repo including branches
22. Go to Issue section of the parent repository (Abhresh’s repo)
23. Create a new issue and past your project url.

# Bonus project: https://github.com/AbhreshSugandhi/PracticeProject

# New Idea:-

M1 commit = Line 1

M2 Commit = Line 2

M3 Commit = Line 3

Now reset to m1 commit but now I don’t want to create a new commit as m2 to club all the data to m1 commit