Install GIT & make sure it is added into PATH.

Section 0 -Use GIT as local VCS. Steps to follow:

* Create a directory ‘project\_dir’ & cd to ‘project\_dir’.

$ mkdir project\_dir

$ cd project\_dir

* Initialize git version database. (git init)

$ git init

* Create a new file index.html.

$ touch index.html

* Check the git status. You should find index.html as untracked file.

$ git status

On branch master

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

index.html

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

* Stage the index.html file.

$ git add index.html

* Commit index.html

$ git commit -m "First Commit"

* Make few changes in index.html & create a new file info.txt file.

$ touch info.txt

* Check git status. You should find index.html & info.txt as untracked files.

$ git status

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: index.html

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: index.html

Untracked files:

(use "git add <file>..." to include in what will be committed)

info.txt

* Configure GIT to ignore all txt files.

touch .gitignore

* Again check the git status. You should find only index.html as untracked file.

$ git status

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: .gitignore

new file: index.html

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: index.html

* State & commit index.html

$ git add index.html

$ git commit -m "After ignoring text file"

* Log all your comments so far.

$ git log

commit 98467b585518a00ae367ae6c6d5b0dfc5101fee4 (HEAD -> master)

Author: Rahul <rahul15159999@gmail.com>

Date: Wed Sep 29 14:44:10 2021 +0530

After ignoring text file

* Make some changes in index.html.
* Revert the change made in the previous step using git command.

$ git revert HEAD

* Again change index.html.
* Stage index.html

$ git add index.html

* Revert back the last stage.

$ git checkout -- .

* Rename ‘add’ command to ‘my-add’.

$ alias my\_add="git add ."

* Using my\_add command Stage index.html again & commit the changes.

$ my\_add

$ git commit -m "Last commit"

* Revert the last commit.

$ git revert HEAD

*GIT Branching*

Objective: Commit HTML, CSS & JavaScript assignments into GIT.

SECTION-1 (HTML assignments) - Steps to follow:

* First take a backup of your assignments & projects. This is required because due to incorrect GIT operation you may lose your files.
* Create an empty directory ‘Assignments’ & cd to ‘Assignments’.

$ mkdir Assignments

$ cd Assignments

* Create a file README.txt inside ‘Assignments’ & write few lines about the contents of ‘Assignments’ folder.

$ touch README.txt

* Commit README.txt file.

$ git add README.txt

$ git commit -m "README.txt file added"

* Now create a new branch ‘html-assignments’.

$ git branch html-assignments

* Switch to ‘html-assignments’ branch.

$ git checkout html-assignments

* Copy all HTML assignments inside ‘Assignments’ folder.

$ touch html\_assignments.html

* Commit HTML assignments into ‘html-assignments’ branch.

$ git add html\_assignments.html

$ git commit -m "First commit of html assignments"

* Make minor changes into few files belonging to ‘html-assignments’ branch.
* Commit those changed files.

$ git add html\_assignments.html

$ git commit -m "Second commit after modify"

* Switch to master branch.

$ git checkout master

* Make minor changes into README.txt file & commit those changes into master.

$ git add README.txt

$ git commit -m "First commit for ReadMe File"

* Again switch to ‘html-assignments’ branch.

$ git checkout html-assignments

* Make minor changes into few files belonging to ‘html-assignments’ branch.
* Commit those changes.

$ git add html\_assignments.html

$ git commit -m "Third commit after second Modification"

* Switch to master.

$ git checkout master

* Merge ‘html-assignments’ branch into master. Confirm all html assignments are shown in master.

$ git merge html-assignments

* Finally delete the ‘html-assignments’ branch.

$ git branch -d html-assignments

SECTION-2 - (CSS assignments) Steps to follow:

* Create a new branch ‘css-assignments’.

$ git branch css-assignments

* Switch to ‘css-assignments’ branch.

$ git checkout css-assignments

* Copy all CSS assignments inside ‘Assignments’ folder.

$ touch css\_assignments.txt

* Commit CSS assignments into ‘css-assignments’ branch.

$ git add css\_assignments.txt

$ git commit -m "CSS assignments added"

* Make minor changes into README.txt file on line 1 belonging to ‘css-assignments’ branch.

$ touch README\_for\_css.txt

* Commit those changed files.

$ git commit -m "Readme for css modified"

* Switch to master branch.

$ git checkout master

* Make minor changes into README.txt file on line 3 & commit those changes into master.

$ git add README.txt

$ git commit -m "Readme for master is modified"

* Again switch to ‘css-assignments’ branch.

$ git checkout css-assignments

* Make minor changes into few files belonging to ‘css-assignments’ branch.
* Commit those changes.

$ git add css\_assignments.txt

$ git commit -m "Modified css assignment file"

* Switch to master.

$ git checkout master

* Merge ‘css-assignments’ branch into master. Confirm all css assignments are shown in master.

$ git merge css-assignments

* Finally delete the ‘css-assignments’ branch.

$ git branch -d css-assignments

SECTION-3 - (JavaScript assignments) Steps to follow:

* Create a new branch ‘js-assignments’.

$ git branch js-assignments

* Switch to ‘js-assignments’ branch.
* $ git checkout js-assignments
* Copy all JavaScript assignments inside ‘Assignments’ folder.

$ touch java\_assignments.txt

* Commit JavaScript assignments into ‘js-assignments’ branch.

$ git add java\_assignments.txt

$ git commit -m "First commit"

* Make minor changes into README.txt file on line 1 belonging to ‘js-assignments’ branch.

$ touch README\_for\_java.txt

* Commit those changed files.

$ git add java\_assignments.txt

$ git commit -m "First commit"

* Switch to master branch.

$ git checkout master

* Make minor changes into README.txt file on line 1 & commit those changes into master.

$ git add java\_assignments.txt

$ git commit -m "Second commit"

* Again switch to ‘js-assignments’ branch.

$ git checkout js-assignments

* Make minor changes into few files belonging to ‘js-assignments’ branch.
* Commit those changes.

$ git add java\_assignments.txt

$ git commit -m "Last commit"

* Switch to master.

$ git checkout master

* Merge ‘js-assignments’ branch into master. Confirm all JavaScript assignments are shown in master.

$ git merge js-assignments

* Finally delete the ‘js-assignments’ branch.

$ git branch -d js-assignments

*GIT Remoting*

Objective: Pushing source code into GITHUB & collaborate team members.

SECTION-3 (Pushing assignments to remote repository) - Steps to follow:

* Create a github account if you do not have already.
* Login on into github account.
* Create new public repository ‘freshersbatch-oct16’.
* Commit & push any sample file to this repository under ‘Assignments’ directory.

SECTION-4 (Pushing source code to remote repository using Eclipse GIT plugin) - Steps to follow:

* One developer from project team will create eclipse projects ‘SampleProj’ & add sample source code files. Then commit all files through eclipse GIT plugin.
* Collaborate other team members with your github account so that they can also modify the committed files.
* Other developers from same team will checkout all files from remote repository. This might get conflicts since certain files fail to merge. In such case, merge it manually.
* Commit & push the ‘SampleProj’ project.