Install GIT & make sure it is added into PATH.

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

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

Ans: $ mkdir project\_dir

$ cd project\_dir

1. Initialize git version database. (git init)

Ans: $ git init

Initialized empty Git repository in C:/Users/CHAILAKS/project\_dir/.git/

1. Create a new file index.html.

Ans: $ touch index.html

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

Ans: $ 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)

1. Stage the index.html file.

Ans: $ git add index.html

$ git status

On branch master

No commits yet

Changes to be committed:

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

new file: index.html

1. Commit index.html

Ans: $ git commit -m "Added file index.html"

[master (root-commit) 9f5a7a1] Added file index.html

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 index.html

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

Ans: $ git add index.html

$ git commit -m "changes in html file"

$ git commit

$ touch info.txt

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

Ans: $ git status

On branch master

Untracked files:

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

index.html

info.txt

1. Configure GIT to ignore all txt files.

Ans: $ touch .gitignore

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

Ans: $ git status

1. State & commit index.html

Ans: $ git add

$ commit -m “changes in index.html”

$ git commit

1. Log all your comments so far.

Ans: $ git log

1. Make some changes in index.html.

Ans: $ vi index.html

$ git add .

$ commit -m “changes in index.html”

$ git commit

1. Revert the change made in the previous step using git command.

Ans: $ git diff head

1. Again change index.html.

Ans: $ vi index.html

1. Stage index.html

Ans: $ git add .

1. Revert back the last stage.

Ans: $ git log -one line

$ git log 9f5a7a106b96f8f0066ae2d5ff524e92bb030365

1. Rename ‘add’ command to ‘my-add’.

Ans: $ git mv add my-add

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

Ans: $ git add .

$ commit -m “changes by adding”

$ git commit

1. Revert the last commit.

Ans: git revert 9f5a7a106b96f8f0066ae2d5ff524e92bb030365

*GIT Branching*

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

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

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

Ans: $ mkdir Assignments

$ cd Assignments

$ git init

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

Ans: $ touch README.txt

1. Commit README.txt file.

Ans: $ git add README.txt

$ git commit -m "Added readme file"

$ git commit

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

Ans: $ git branch html-assignments

1. Switch to ‘html-assignments’ branch.

Ans: $ git checkout html-assignments

1. Copy all HTML assignments inside ‘Assignments’ folder.

Ans: Create Html files

$ cp \*.html Assignments

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

Ans: $ git add

$ git commit -m "changes in html files"

$ git commit

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

Ans : $ git commit -m "i changes"

$ git commit

1. Switch to master branch.

Ans: $ git checkout master

Switched to branch 'master’

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

Ans: $ vi README.txt

$ git add .

$ git commit -m "changes in readme files"

$ git commit

1. Again switch to ‘html-assignments’ branch.

Ans: $ git checkout html-assignments

Switched to branch 'html-assignments'

1. Make minor changes into few files belonging to ‘html-assignments’ branch.

Ans: $ vi a.html

1. Commit those changes.

Ans: $ git commit -m "i changes"

1. Switch to master.

Ans: $ git checkout master

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

Ans: $ git merge html-assignments

1. Finally delete the ‘html-assignments’ branch.

Ans: $ git branch -d html-assignments

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

1. Create a new branch ‘css-assignments’.

Ans: $ git branch css-assignments

1. Switch to ‘css-assignments’ branch.

Ans: $ git checkout css-assignments

Switched to branch 'css-assignments'

1. Copy all CSS assignments inside ‘Assignments’ folder.

Ans: Create css files

$ cp \*.css Assignments

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

Ans: $ git add .

$ git commit -m "changes in css file"

$ git commit

1. Make minor changes into README.txt file on line 1 belonging to ‘css-assignments’ branch.
2. Commit those changed files.

Ans: $ git add .

$ git commit -m "changes in readme file"

$ git commit

1. Switch to master branch.

Ans: $ git checkout master

Switched to branch 'master'

1. Make minor changes into README.txt file on line 3 & commit those changes into master.
2. Again switch to ‘css-assignments’ branch.

Ans: $ git checkout css-assignments

Switched to branch 'css-assignments'

1. Make minor changes into few files belonging to ‘css-assignments’ branch.

Ans: $ vi a.css

$ git status

On branch css-assignments

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: a.css

1. Commit those changes.

Ans: $ git add .

$ git status

On branch css-assignments

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

new file: Assignments

modified: a.css

1. Switch to master.

Ans: $ git checkout master

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

Ans: $ git merge css-assignments

1. Finally delete the ‘css-assignments’ branch.

Ans: $ git branch -d css-assignments

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

1. Create a new branch ‘js-assignments’.

Ans: $ git branch js-assignments

1. Switch to ‘js-assignments’ branch.

Ans: $ git checkout js-assignments

Switched to branch 'js-assignments

1. Copy all JavaScript assignments inside ‘Assignments’ folder.

Ans: Create javascript files

$ cp \*.js Assignments

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

Ans: $ git add .

$ git commit -m "changes in js file"

$ git commit

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

1. Commit those changed files.

Ans: $ git add .

$ git commit -m "changes in readme file"

$ git commit

1. Switch to master branch.

Ans: $ git checkout master

Switched to branch 'master'

1. Make minor changes into README.txt file on line 1 & commit those changes into master.
2. Again switch to ‘js-assignments’ branch.

Ans: $ git checkout js-assignments

Switched to branch 'js-assignments'

1. Make minor changes into few files belonging to ‘js-assignments’ branch.

Ans: $ vi a.js

$ git status

On branch js-assignments

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: a.js

1. Commit those changes.

Ans: $ git add .

$ git status

On branch js-assignments

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

modified: a.js

1. Switch to master.

Ans: $ git checkout master

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

Ans: $ git merge js-assignments

1. Finally delete the ‘js-assignments’ branch.

Ans: $ 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:

1. Create a github account if you do not have already.

Ans : Created

1. Login on into github account.

Ans: Done

1. Create new public repository ‘freshersbatch-oct16’.

Ans: My repository is test\_demo

1. Commit & push any sample file to this repository under ‘Assignments’ directory.

Ans: $ git add Firstdemo.txt

$ git commit -m “Commiting Firstdemo.txt file”

$ git config –global user.username Chaitali0301

Create Repository

$ git remote origin <https://github.com/Chaitali0301/test-demo.git>

Push local file into remote repository

$ git push origin master

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

1. One developer from project team will create eclipse projects ‘SampleProj’ & add sample source code files. Then commit all files through eclipse GIT plugin.

Ans🡪 1. Open eclipse IDE then shift to git repository.

2. Select add on existing local repository.

3. Browse the repository foder and added to it.

4. Select to clone a repository and enter url and finish.

1. Collaborate other team members with your github account so that they can also modify the committed files.
2. 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.
3. Commit & push the ‘SampleProj’ project.

Ans🡪 1. Right click on project and select Commit.

2. Select commit.

3. We can git stagung view and then select commit.

4.then right click on the project and select team.

5. In that team menu select push branch master.