GIT

GIT:Global Information Tracker.

Distributed version control system.

Vcs-managing and storing the different versions of the code

BENEFITS OF VCS:

- Control
- Security
- Fast delivery
- Availability

BENEFITS OF GIT:

- System compatibility
- Version compatibility
- Speed
- Distribute System and readability
- Security

WORKFLOW OF GIT:

1. Working repository
2. Initialization
3. Stagging
4. Local storage
5. Repository

BASICS STEPS:

1)create repository in github or cloud 9

2)In command prompt we have to create a folder

mkdir intro-into-git

3)**git init** -initialize git.

4)To create a file in the folder.

nanoIndex.html or touch index.html

5)we have to stage that file- **git add index.html.**

6)**git commit -m"added index.thlm"-**commit it

7) To check status **git status.**

8)To see the history **git log.**

9) to make an hidden file **touch .hidden.txt**

10) To stage multiple file with same extension **git add *.html**

11) To add all files including hidden files git add -A. In the command prompt git add .

Solutions:

- Create a new directory for your project `mkdir git_section_2`
- Change directories into your project folder `cd git_section_2`
- Initialize a Git repository to begin tracking your project `git init`
- Create some random files for the project (e.g., `touch index.html && touch style.css`)
- Check your `git status`
- Add the files to the staging area `git add <file-name>` (repeat for each file)
- Check the status of your repository again
- Commit the files to your git repository `git commit -m "Commit message here"`

Congratulations! You've successfully started tracking your project with Git. In the next section we'll learn how to add, remove, and ignore specific files and folders in your git repository

Command to remove file from staging area

Git reset HEAD <file>

COMMANDS FOR BRANCH:

Git branch **list branch**

Git branch feature-1 **add new branch**

touch index.html **create a new file**

git add index.html **stage**

git commit -m"added feature -1" **commit**

git log

Commit available "added feature -1"

git checkout master **change branch**

git status

git log

Git merge feature-1 **merge branch**

git commit "feature-1 is added "

Git status

git log

git branch -d feature-1

Section 3 - Practice Exercise Solutions

Solution:

- Create a new folder for this project, run all commands in this exercise from this folder `mkdir git_section_3`
- Change directories into git_section_3 `cd git_section_3`
- Initialize a Git repository to begin tracking your project `git init`
- Create 3 new files using the touch command (name them **file1.txt**, **file2.html**, and **file3.js**) `touch file1.txt file2.html file3.js`
- Create 1 new folder named **random_files** `mkdir random_files`
- Move the text file (.txt) and the javascript file (.js) into the random_files directory `mv file1.txt random_files && mv file3.js random_files`
- Check the status of your repository (you will only see the random_files directory listed, not the files inside it) `git status`
- Add all newly created/untracked files and folders to the staging area with `git add .` OR `git add -A`
- Check your `git status` again
- Remove **file3.js** from the staging area `git rm --cached random_files/file3.js`
- Create 3 new files in the **random_files** directory (name them **file4.css**, **file5.css**, and **file6.js**) `cd random_files ; touch file4.css file5.css file6.js ; cd ..`
- Check your `git status`

Section 4 - Practice Exercise Solutions

Solutions:

- Make a new directory named **git_section_4** mkdir git_section_4
- Change into this directory cd git_section_4
- Initialize a git repository git init
- Create some files, add them to the staging area, and commit them to git
    - touch file1.txt && touch file2.html && touch file3.css
    - git status
    - git add -A
    - git status
    - git commit -m "Add 3 new files"
- List all current branches git branch
- Create a new branch named **feature** git checkout -b feature
- List your branches and make sure you're in the **feature** branch, if not, change into it
    - git branch
    - if not in feature, git checkout feature
- Create some files and make some changes to one of the existing files
    - touch file4.js && touch file5.html
    - echo "hello world" >> file1.txt
- Commit these changes to the **feature** branch
    - git status
    - git add -A
    - git status
    - git commit -m "Add feature files and add hello world to file1.txt"
- Checkout your **master** branch and merge the **feature** branch into **master**
    - git checkout master
    - git merge feature
- Remove the **feature** branch

    - git branch -d feature
- List all branches to ensure feature was removed `git branch`

Section 5 - Exercise Solutions

Solutions:

- Use your terminal to open up the **git_section_4** directory from the last set of practice exercises.
- You should have 2 commits in this git repository, if you don't have 2 then go ahead and create some changes and commit them so you will have 2 to work with
- Check your repository commit history git log
- Checkout the first commit git checkout <SHA-1_hash_key_here>
- Make some changes to **file1.txt** echo "hello world" >> file1.txt
- Checkout a new branch to save these changes into git checkout -b feature2
- Check the status of the repository git status
- Add **file1.txt** to the staging area git add file1.txt
- Check the status of the repository again git status
- Commit the changes git commit -m "Update file1.txt"
- Switch back to the **master** branch git checkout master
- Check your repo commit history (notice how the changes from your new branch do not exist in master) git log

Completed the "Intro to Git" (1st tutorial) Continuing "Git Tutorial for Beginners" (2nd Tutorial)