Course name : Intro to GIT

GIT - Global Information Tracker

1.What is Git ?

- Git is a Version control system
- It stores reference points to snapshots of your code.
- Designed to handle minor to major projects with high speed and efficiency.
- Keep track of all the changes that we make in our files of our project.

IDE(Integrated development Enviroinment)
Cloud 9 (c9) service.

Git commands :

1. **mkdir** (Create a new directory)
2. **Git init** command :
 Git init (Initialize empty git repository inside the directory)
 Creates a new git repository.It can be used to convert an existing unversioned project to a git repository or initialize a new empty repository
3. **ls~a** : Shows all files inside the directory
4. **(master)** : Indicates that the git is initialized.

Three states of git:
1.Working Directory
  → Area where all our files and directories and changes are living all the time .
2.Staging Area
  → Files and directories that we explicitly add to the staging area
3.Git repository
  → Where all our snapshots are stored.

Initially the file will be created in Working directory.

<u>Commands:</u>

git add <filename> → Add file in staging area
git status → Displays the state of the working directory and the staging area.
rm → remove individual files or a collection of files.
mv → Rename or move files within the git repository without deleting its history.
Touch → Command used in the UNIX/Linux operating system which is used to create,change and modify the timestamps of files.

git commit -m "Message here" → adding commits to keep track of our progress and changes as we work. Git considers each commit change point or "save point".
It is a point in the project you can go back to if you find a bug, or want to make a change.
When we commit, we should always include a message.
git log → To view the history of commits for a repository, you can use the log command.
Create hidden file → touch .filename
Notes :

Practice 1 :
Instructions:

- Create a new directory for your project
- Change directories into your project folder
- Initialize a Git repository to begin tracking your project
- Create some random files for the project (e.g., index.html and style.css)
- Check the status of your repository
- Add the files to the staging area
- Check the status of your repository again
- Commit the files to your git repository

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"

Adding multiple files of certain type

git add *.html → Adds all html files in the staging area

Adding all files in directory (including hidden)

git add -A → (adding all files in staging area)
Adds all files and folders from the directory that you are in.
This is a good command for adding everything in your project,all at one time
git commit -m → Add all files in repository

Remove files
After adding the files from staging area,

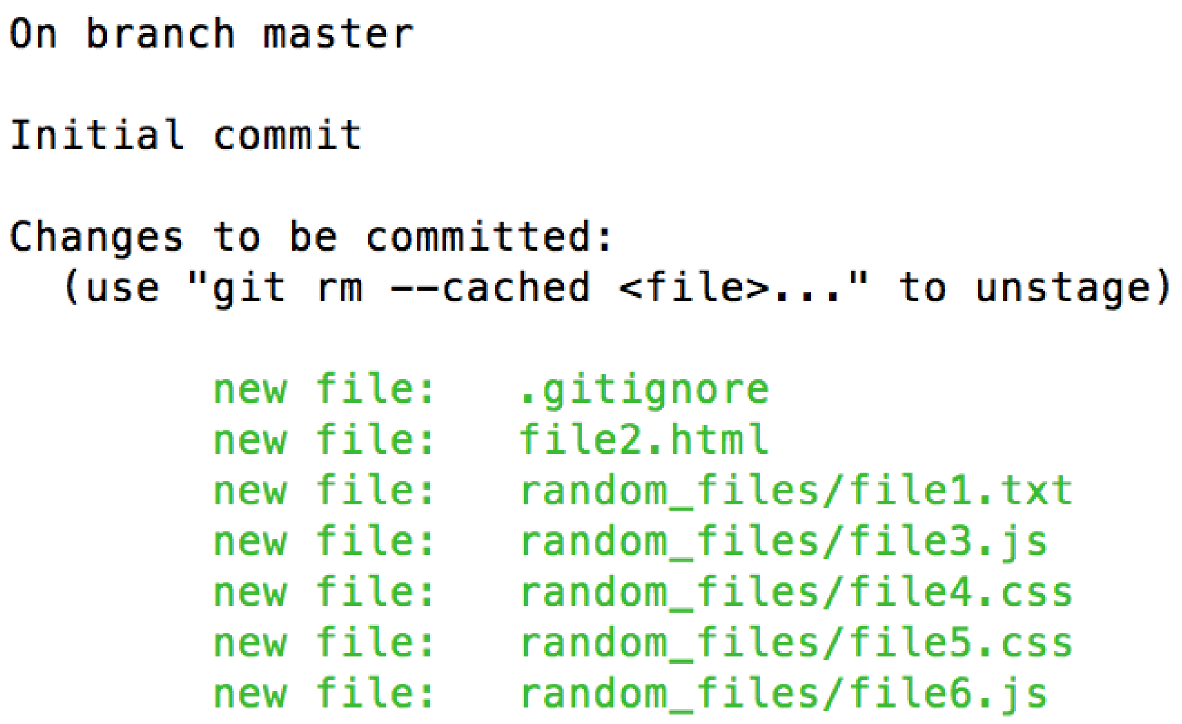
git reset HEAD <file> → remove the file from the staging area ,and it will be stored in the working directory.

Ignoring the files :
While inserting the normal files in hidden files, it

Practice 2 :

Instructions:

- Create a new folder for this project, run all commands from this folder (name it **git_section_3**)
- Change directories into **git_section_3**
- Initialize a Git repository to begin tracking your project
- Create 3 new files using the touch command (name them **file1.txt**, **file2.html**, and **file3.js**)
- Create 1 new folder named **random_files**
- Move the text file (**.txt**) and the javascript file (**.js**) into the **random_files** directory
- Check the status of your repository (you will only see the random_files directory listed, not the files inside it)
- Add all newly created/untracked files and folders to the staging area
- Check the status of your repository
- Remove **file3.js** from the staging area
- Create 3 new files in the **random_files** directory (name them **file4.css**, **file5.css**, and **file6.js**)
- Check the status of your repository
- Add all files with the file type of **.css** to the staging area (hint: you need to be inside of the **random_files** directory)
- Check the status of your repository
- Add all files with the file type of **.js** to the staging area
- Check the status of your repository
- Create a new directory named **secret_stuff** (hint: make sure you **cd** back into **git_section_3** first)
- Create two files inside of **secret_stuff** named **file1.yml** and **file2.js**
- Create a **.gitignore** file so we can ignore the **secret_stuff** directory and all of its contents (hint: **.gitignore** should be inside of **git_section_3**)
- Add the **secret_stuff** folder to the **.gitignore** file
- Check the status of your repository
- Add the **.gitignore** file to the staging area
- If your staging area looks like the image below then you have completed this exercise successfully. You may now commit your changes

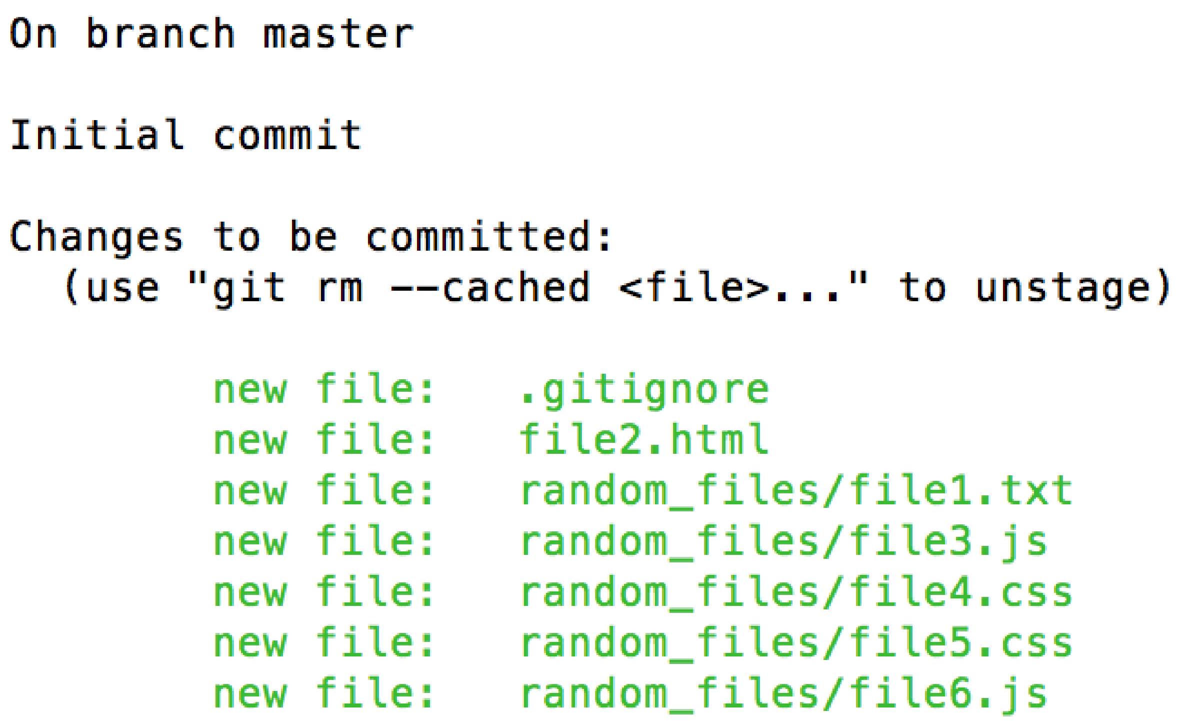


```
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   .gitignore
        new file:   file2.html
        new file:   random_files/file1.txt
        new file:   random_files/file3.js
        new file:   random_files/file4.css
        new file:   random_files/file5.css
        new file:   random_files/file6.js
```

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
- Add all files with the file type of **.css** to the staging area (hint: you need to be inside of the **random_files** directory if you aren't already) cd random_files ; git add *.css ; cd ..
- Check your git status
- Add all files with the file type of **.js** to the staging area cd random_files && git add *.js && cd ..
- Check your git status
- Create a new directory named **secret_stuff** mkdir secret_stuff
- Create two files inside of **secret_stuff** named **file1.yml** and **file2.js**cd secret_stuff && touch file1.yml && touch file2.js && cd ..
- Create a **.gitignore** file so we can ignore the **secret_stuff** directory and all of its contents (hint: **.gitignore** should be inside of **git_section_3**) touch .gitignore
- Add the **secret_stuff** folder to the **.gitignore** file (you can use the following command or do this in your text editor) echo "secret_stuff" >> .gitignore
- Check your git status
- Add the **.gitignore** file to the staging area git add .gitignore
- If your staging area looks like the image below then you have completed this exercise successfully. You may now commit your changes git commit -m "Complete section 3 exercise"



```
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   .gitignore
        new file:   file2.html
        new file:   random_files/file1.txt
        new file:   random_files/file3.js
        new file:   random_files/file4.css
        new file:   random_files/file5.css
        new file:   random_files/file6.js
```

Git branches

<u>Listing all branches</u>
Git branch → List all current branches

<u>Adding a branch</u>
git branch <branch name> → used to create a new branch
git checkout -b feature → Create a new branch named <u>feature</u>

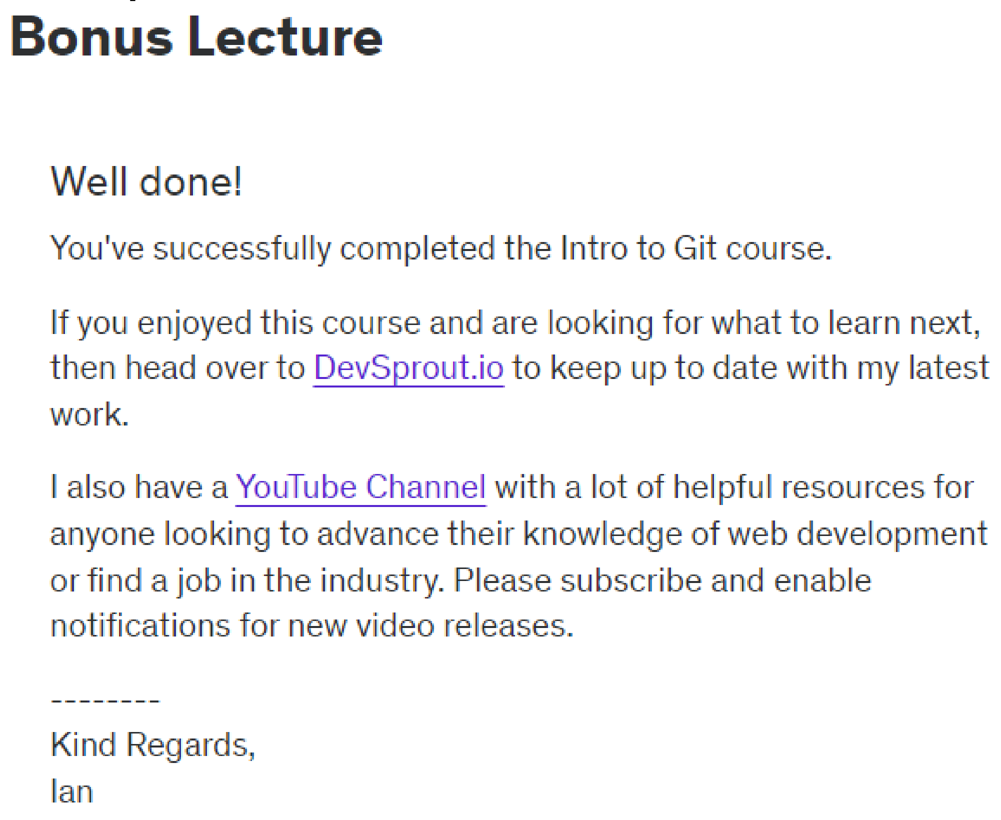
<u>Changing a branch</u>

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
<u>Merging branches together</u>
Checkout your master branch and merge the <u>feature</u> branch into master
git checkout master
git merge feature

Removing a branch

```
Git branch -d <branch name>
```

Course completion



Bonus Lecture

Well done!

You've successfully completed the Intro to Git course.

If you enjoyed this course and are looking for what to learn next, then head over to DevSprout.io to keep up to date with my latest work.

I also have a YouTube Channel with a lot of helpful resources for anyone looking to advance their knowledge of web development or find a job in the industry. Please subscribe and enable notifications for new video releases.

Kind Regards,

Ian