GIT TASKS

BY ANJU M DOMINIC

TASK 1 : HOW TO DO GIT INSTALLATION AND GITHUB ACCOUNT CREATION

- Activity:
  - Being a developer you are going to work on Git version control system. Configure git in your development system which is a Ubuntu 18.04 system
- Steps:
  - Install Git: *sudo apt-get install git*
  - Get the version of git using command below: *git --version*
  - Create an account in GitHub

- Definition of Done:
  - Git installed in your development system

TASK 2 : HOW TO CONFIGURE GIT WITH YOUR USERNAME TO TRACK THE CHANGES YOU MAKE

- Activity:
  - Being a developer you have done some code changes and want the changes to be mapped under your name

- Steps:
  - Update git config file with your username and email id. Edit the git config file

    ```
    git config --global –edit
    ```

  Save the file with ctrl + x key combination and then enter

- Definition of Done:
  - Git configured and all commits done in the system will be under your name

TASK 3 : HOW TO GET THE SOURCE CODE FROM AN EXISTING GIT REPOSITORY

- Activity:
  - Being a new developer in your project you need to get the source code for the project. Clone a GitHub repository https://github.com/AnjuMeleth/HelloWorld.git where we have the source code

- Steps:
  - Clone the repository https://github.com/AnjuMeleth/HelloWorld.git

git clone https://github.com/AnjuMeleth/HelloWorld.git

- Definition of Done:
  - Cloned repository is viewable in the system

TASK 4 : HOW TO CREATE A GIT REPOSITORY FOR THE PROJECT WHICH IS GOING TO START

- Activity:
  - Being a developer you are assigned a new project that needs to be created from the scratch and there is no existing source code and a repository. You are using Git version control system to create a central repository and track the source code files.
- Steps:
  - Create a directory hello in your current path and go inside the directory

    mkdir hello

    cd hello
  - Initialize git repository : *git init*
  - View the repository using the command : *ls –al*
  - Create a file "hello.html" with below contents in the hello directory
  - *vi hello.html*

    <h1>Hello World </h1>

    Save the file using esc + : +w +q + !
  - Create a repository in GitHub called HelloSample
  - Add the GitHub details to the .git folder : *git remote add origin <GitHub URL>*
- Definition of Done:
  - .git folder could be viewed in the initialized directory

TASK 5 : HOW TO PUSH YOUR SOURCE CODE TO A CENTRAL REPOSITORY ON GITHUB

- Activity:
  - From the previous task you might have already created a file "hello.html" in the hello folder. Assume that you are done with the coding and want to be moved to a central repository

- Steps:
  1. Add the "hello.html" file to the staging area : *git add hello.html*
  2. View the objects folder in the .git folder. Run the command to view the object details in the hello directory path

     find .git/objects -type f

     git cat-file -p <SHA>

     You can also verify the hash object by the command

     git hash-object -w helloworld.html
  5. Commit the file : *git commit –m "Added hello.html"*

     find .git/refs

     git log --pretty=oneline master
  6. Push the file to GitHub repository HelloSample : *git push -u origin master*

- Definition of Done:
  - Source code pushed to a GitHub repository

TASK 6 : FORK THE REPOSITORY

- Activity:
  - In your project the source code is kept in https://github.com/AnjuMeleth/WorkBook_Content.git repository . Fork the repository
- Steps:
  - Fork the https://github.com/AnjuMeleth/WorkBook_Content.git repository.
  - Then clone the repository to your local.
- Definition of Done:
  - Git Hub Repository having all the source code that we are going to use in our course is cloned to your local

TASK 7 : HOW TO DELETE A COMMITTED FILE (GIT RM)

- Activity:
  - In your source code, after analysis you found that you are not using product.css file and you want to delete the file from your repository.
- Definition of Done:
  - Tracked file is deleted
  - A commit in the commit history that states that the file is deleted

#	Steps	Commands
1	Go to directory DeleteFile	cd DeleteFile
2	List down the files in the path	ls
3	Delete the file product.html	git rm product.css
4	Check the status	git status
5	Commit the Files	git commit -m "deleted unwanted file"
6	Push the commit	git push

TASK 8 : HOW TO DELETE A COMMITTED FILE (RM)

- Activity:
  - In your source code, after analysis you found that you are not using category.css file and you want to delete the file from your repository.
- Definition of Done:
  - Tracked file is deleted
  - A commit in the commit history that states that the file is deleted

#	Steps	Commands
1	Go to directory DeleteFile	cd DeleteFile
2	List down the files in the path	ls
3	Delete the file product.html	rm category.css
4	Check the status	git status
5	Commit the Files	git commit -m "deleted unwanted file using rm"
6	Push the commit	git push

TASK 9 : HOW TO RENAME A COMMITTED FILE(GIT MV)

- Activity:
  - You have created a file and committed it. Due to coding standards you have to rename the file. Kindly fix the issue?
- Definition of Done:
  - Renamed the committed file
  - A commit in the commit history that states that the file is renamed

#	Steps	Commands
1	Go to directory RenameFile	cd RenameFile
2	List the files in the path	ls
3	Rename the file	git mv productdiscount.html ProductDiscount.html
3	Get the status of the file	git status
5	Commit the file	git commit -m "Renamed the file"
6	Push the commit	git push

TASK 10 : HOW TO RENAME A COMMITTED FILE(MV)

- Activity:
  - You have created a file and committed it. Due to coding standards you have to rename the file. Kindly fix the issue?
- Definition of Done:
  - Renamed the committed file
  - A commit in the commit history that states that the file is renamed

#	Steps	Commands
1	Go to directory RenameFile	cd RenameFile
2	List the files in the path	ls
3	Rename the file	mv ProductDiscount.html ProdDiscount.html
4	Get the status of the file	git status
5	Commit the file	git commit -m "Renamed the file using mv command"
6	Push the commit	git push

TASK 11 : HOW TO REMOVE FILE FROM THE STAGED STATE

- Activity:
  - You have accidentally added a file in the staging area and you want to remove it before committing. What should you do?

- Steps:
  - Go to directory "RemoveFileFromStagingArea" : *cd RemoveFileFromStagingArea*
  - Edit the files by including pricing as well
  - Add them to staging area : *git add .*
  - You realized mango.html need not to be added in this commit
  - Remove it from the staging area by the following command

    git reset HEAD mango.html
  - Commit the vegetable files then fruit file

- Definition of Done:
  - Mango.html file is removed from the staging area

TASK 12 : HOW TO MODIFY A STAGED FILE

- Activity:
  - You have modified a file contact.html and added it to staging area. But later found that you have to add some more details. Kindly fix the issue?

- Definition of Done:
  - All the changes in the contact.html should be added to staging area

#	Steps	Commands
1	Go to folder ModifyingStagedFile	cd ModifyingStagedFile
2	Modify the file, contact.html (add Email)	vi contact.html
3	Add the file to staging area	git add contact.html
4	Get the status of the file	git status
5	Modify the file again(add Mobile)	vi contact.html
6	Get the status of the file	git status
7	Add the file to staging area	git add contact.html
8	Get the status of the file	git status
9	Commit the file	git commit -m "Modified contact.html"
10	Push the commit	git push

TASK 13 : HOW TO DELETE A COMMIT FROM THE COMMIT HISTORY (MIXED RESET)

- Activity:
  - You have made a commit by mistake and want to delete the commit from the commit history but you need to keep the modifications that went into the commit. Kindly fix the issue?
- Definition of Done:
  - Last one committed will be deleted from the history

#	Steps	Commands
1	Go to directory "Reset"	cd Reset
2	List down the files in the path	ls
3	Modify the file category.html to add category groceries	vi category.html
3	Get the status of the files	git status
4	Add category file to staging area	git add category.html
5	Commit the file	git commit -m "Modified category.html"
6	List the commits	git log --oneline
7	Delete the last commit as we still have more categories like fruits to add	git reset HEAD~1
8	List the commits	git log --oneline
9	Get the status of the files	git status

TASK 14: RESET(SOFT)

- Activity:
  - You have made a commit by mistake and want to delete the commit from the history. Kindly fix the issue?

- Definition of Done:
  - Last 1 commit is deleted from history.

#	Steps	Commands
1	Modify the file category.html and category.css to update the category page	vi category.html vi category.css
2	Get the status of the files	git status
3	Add category file to staging area	git add category.html
4	Commit the file	git commit -m "Modified category page"
5	List the commits and push the commit	git log --oneline git push
6	Delete the last commit as you forgot to add the category.css file	git reset --soft HEAD~1
7	List the commits	git log --oneline
8	Get the status of the files	git status
9	Create a new commit with all files and push it	git push

TASK 15: HOW TO RESET A COMMIT IN THE CENTRAL REPOSITORY

- Activity:
    - After doing the previous task on soft reset and then try to push it to GitHub. You get error message as there is conflict in the local history and history in the GitHub. How to solve this problem

- Definition of Done:
    - New history in the GitHub.

#	Steps	Commands
1.	Create a new commit with all files and push it	git push --force
2	Get the history in the GitHub	

TASK 16 : RESET(HARD)

- Activity:
  - All the changes you have made to the category page was no longer needed and needs to be removed permanently with it's modifications. Kindly fix the issue?

- Definition of Done:
  - Last 1 commit deleted along with it's modifications

#	Steps	Commands
2	List the commits	git log --oneline
3	Delete the last commit	git reset --hard HEAD~1
4	List the commits	git log --oneline
5	View the file	cat category.html

TASK 17 : HOW TO MODIFY THE LATEST COMMIT FOR AN ERROR COMMIT MESSAGE

- Activity:
  - Accidentally you have committed with an error message . Kindly fix the issue?

- Definition of Done:
  - Commit message of the last commit is updated

#	Steps	Commands
1	Create a new commit	
2	List the commits	git log --oneline
3	Amend the last commit	git commit --amend
4	List the commits	git log --oneline

TASK 18 : HOW TO REVERT AN OLD COMMIT

- Activity:
  - You have ended up in a commit which is not relevant for you any more but wants to keep it in the history as for logging purpose . You want to revert it . Kindly fix the issue?

- Definition of Done:
  - Selected commit is reverted but old commit still remains in the history

#	Steps	Commands
1	List the commits	git log --oneline
2	Select a commit and show the files that went into the commit	git show <commit Id>
3	Revert the selected commit	git revert <commit id>
4	List the commits	git log --oneline
5	Select the new commit and show the file changes	git show <new commit Id>

TASK 19 : HOW TO STOP FILES GETTING ADDED TO GIT EVEN BY MISTAKE ESPECIALLY CONFIDENTIAL FILES(PROJECT SPECIFIC)

- Activity:
  - You have a log file in your application which could get added accidentally to your code repository (git add .). Kindly fix the issue?

- Definition of Done:
  - Newly created log file is not even shown as untracked file

#	Steps	Commands
1	Go to directory gitignore	cd gitignore
2	List down the files in the path	ls
7	Assume a new log file is created by your application	touch log0407.txt
8	Check the status	git status
9	Create a .gitignore file in the root path	vi .gitignore
10	Enter the file details to be ignored	log*.txt
11	Check the status	git status
12	Add .gitignore to staging area	git add .gitignore
13	Check the status	git status
14	Commit the file	git commit -m "Added .gitignore"
15	Push the commit	git push

TASK 20 : HOW TO STOP FILES GETTING ADDED TO GIT EVEN BY MISTAKE ESPECIALLY CONFIDENTIAL FILES(GLOBAL)

- Activity:
  - You have a log file in your application which could get added accidentally to your code repository. Kindly fix the issue?
- Definition of Done:
  - Selected files are ignored across all of your repositories

#	Steps	Commands
1	Create a global .gitignore files	touch ~/.gitignore
2	Create a exclude property set up	git config --global core.excludesFile ~/.gitignore
3	Include common patterns like .DS_Store	

TASK 21: HOW TO IGNORE A FILE THAT IS ALREADY BEING TRACKED BY GIT

- Activity: Git is tracking a file in a source code path. You are making changes on the file and if the changes get added to the production branch accidentally it's very risky. How to solve it?

- Definition of Done:
  - Modifications on a file removed from the git system is never tracked

#	Steps	Commands
1	Untrack the file from git	git rm --cached log0107.txt
2	Modify the file log0107.txt and get the status of the file	

TASK 22: **BASIC BRANCHING AND MERGING**

- Activity:
  - You are being asked to work on an issue "iss53" for your website project . While half way through it you received a request for a hotfix. After fixing the hotfix you need to return back to your iss53 issue. How to manage this using git branches ?

#	Steps	Commands
1	Go To directory.	cd Merge
2	Create a branch iss53	git branch iss53
3	Checkout the branch	git checkout iss53
4	Work on issue 53	touch issue53.html
5	Add the file to staging area	git add issue53.html
6	Commit the file	git commit -m "Added issue53.html"

Task 22 : Basic Branching and Merging cont..

- Definition of Done
  - Fast forward merge of hotfix branch

7	Checkout master branch	git checkout master
8	Create a branch hotfix	git branch hotfix
9	Checkout the hotfix branch	git checkout hotfix
10	Fix the defect	touch hotfix.html
11	Add the file to staging area	git add hotfix.html
12	Commit the file	git commit -m “Fixed hotfix"
13	Checkout the master branch	git checkout master
14	Merge the hotfix branch	git merge hotfix
15	List the existing branches	git branch
16	Delete the hotfix branch	git branch -d hotfix
17	Checkout the iss53 branch and modify the file. Then commit the changes	git checkout iss53 git push

TASK 23 : RECURSIVE MERGE

- Activity:
  - (Continued activity from Task 22).You have a master branch and a iss53 branch. You now need to merge the issue branch. How would you merge?

- Definition of Done
  - Recursive merge of iss53 branch

#	Steps	Commands
1	Go to the folder Merge	cd Merge
2	List the branches.	git branch
3	Checkout master (Make sure you are on the master branch)	git checkout master
4	Merge iss53 branch	git merge iss53 -m "Recursive merge for iss53"
5	Push the commit	git push
6	Delete the iss53 branch	git branch -d iss53
7	List the branches	git branch

TASK 24 : HOW TO FIX MERGE CONFLICTS

- Activity:
  - You have a master branch and two feature branch. Same file is getting modified in both branches. This causes conflicts. What should you do?

- Definition of Done:
  - Merge conflicts arises

#	Steps	Commands
1	Go to folder MergeConflicts	cd MergeConflicts
2	Create and checkout branch redesign	git checkout -b redesign
	Create and checkout branch date	git checkout -b date
3	Modify home.html in redesign branch(links to contact page)	vi home.html
3	Add and commit the file	git add home.html git commit -m "Home page redesigned"
4	Modify home.html in date branch(added date section)	vi home.html
5	Add and commit the file	git add home.html git commit -m "Added date to home page"

Task 24 : Merge Conflicts Cont…

6	Checkout master	git checkout master
7	Merge redesign branch	git merge redesign
8	Merge date branch	git merge date
9	View and fix the conflicts	vi home.html
10	Add and commit the home.html file. Push the commit	git add home.html git commit -m "Fixed merge conflicts"
11	Delete the branches	git branch -d redesign git branch -d date

TASK 25 : HOW TO CREATE A PARALLEL LINE OF WORK FOR YOUR PROJECT (GLOBAL BRANCH

- Activity:
  - You have a master branch where all your code is kept. You want to run a parallel line of work so that multiple work could be done at a time.

#	Steps	Commands
1	Go to folder CreateABranch	cd CreateABranch
2	List the files	ls
3	Create a welcome branch	git branch welcome
4	List down the branches	git branch
5	Checkout the welcome branch	git checkout welcome
6	List down the branches	git branch
7	Touch file in the welcome branch	touch welcome.html
8	Add the file to the staging area	git add welcome.html
9	Commit the file	git commit -m "Added welcome.html"
10	List the commits in the welcome branch	git log --oneline

TASK 25: CREATING A GLOBAL BRANCH CONT…

- Definition of Done:
  - Global branch created

11	Checkout the master branch	git checkout master
12	List the commits in the master branch	git log --oneline
13	Push the commit along with the branch to Git Hub	git push origin welcome
14	Delete the local branch welcome	git branch -D welcome

TASK 26 : HOW TO DO GLOBAL BRANCH MERGE

- Activity:
  - You have a master branch and a home branch globally. How to merge the global home branch to master branch?
- Steps:
  - Track the global branch using the command : *git checkout --track origin/welcome*
  - Pull down the changes : *git pull*
  - Checkout the master branch : *git checkout master*
  - Merge the home branch : *git merge welcome*
  - Push the files : *git push*
  - Delete the global home branch : *git push origin --delete welcome*

- Definition of Done:
  - All the modifications in the welcome branch is moved to master branch and it's deleted globally

TASK 27 : CENTRALISED WORKFLOW

- Activity:
  - You have a team of 2 members working on a product called GRAPES and wanted to adopt centralized workflow for your project development. What are the steps you should follow?

- Steps:
  - Create a new repository called GRAPES
  - Add your team member AnjuMeleth as a collaborators . Go to Settings -> Manage Access -> Invite a collaborator

- Definition of Done:
  - Collaborator is added

TASK 28 : FEATURE WORKFLOW

- Activity:
  - You are a team of 10 members and 5 members are going to work for Feature A and other 5 in Feature B. Your Team have decided to adopt Feature Workflow. What are the steps you should follow?

- Steps:
  - Go to folder FeatureWorkFlow.
  - Create two branches Feature A and Feature B
  - Create a new file in each branch and commit it.
  - Push the branches with the files.

- Definition of Done:
  - Feature branches created

TASK 29: GITFLOW WORKFLOW

- Activity:
  - You are working on an e-commerce application and you want to have a dedicated branch for each of the releases you do . What are the steps you should follow?

- Steps:
  - Create a new folder Gitflow
  - Create a branch release1.
  - Create a new file and commit it.
  - Push the branch with files.

- Definition of Done:
  - Release branches created

TASK 30: HOW TO FORK A REPOSITORY

- Activity:
  - You are working on an e-commerce application. You want to have a dedicated repository exclusively for you so that all your modifications should be reviewed first before merging to central repository. What are the steps you should follow?

- Steps:
  - Go to the GitHub repository https://github.com/AnjuMeleth/static.git
  - Fork the repository
- Definition of Done:
  - Forked repository could be seen in under your GitHub Account.

TASK 31 :FORK WORKFLOW (DEMO)

- Activity:
  - You have a forked repository and you found that some modifications need to be done on the code. How could you get the new code to your parent repository?

- Definition of Done:
  - Pull request needs to be created and accepted

#	Steps	Commands
1	Go to the GitHub repository after logging as a new user https://github.com/AnjuMeleth/static.git	
2	Fork the repository	
3	Modify the index.html in your forked repository	vi index.html
4	Create a pull request. Go to New Pull Request	
5	Administrator of the https://github.com/AnjuMeleth/static.git accepts the pull request and it's approved	
6	Merge Pull request and confirm merge	

TASK 32: REBASING

- Activity:
  - You have a master branch and a branch called preorder. There occurred multiple commits on the master after the branching. You now need to bring all the commits to the preorder branch and do the integration test in the preorder branch itself. What are the steps you should follow?

- Definition of Done:
  - All commits are brought down to the preorder branch

#	Steps	Commands
1	Go to folder Rebasing	cd Rebasing/
2	Create and checkout branch preorder	git checkout -b preorder
3	Create a preorder.html and commit the file	touch preorder.html git add preorder.html git commit -m "Added preorder.html file"
4	Checkout the master branch	git checkout master
5	Create another commit on master branch	touch sales.html git add sales.html git commit -m "Added sales files"

TASK 32: REBASING CONT…

6	Checkout preorder branch	git checkout preorder
7	Get the commit log	git log --oneline
8	Rebase master	git rebase master
9	Get the commit log	git log --oneline
10	Checkout the master branch	git checkout master
7	Merge the preorder branch	git merge preorder
8	Push the commit	git push

TASK 33 : STASHING

- Activity:
  - You have a master branch and a feature branch called wholesalers .You are currently working with a page for wholesalers but a high priority issue have come to fix an issue in the master. What should you do?
- Definition of Done:
  - Changes in the wholesalers branch is stashed and could be retrieved again

#	Steps	Commands
1	Go to folder stashing	cd stashing
2	Create a branch wholesalers and modify a file product.html	git checkout -b wholesalers vi product.html
3	Stash the changes and checkout master branch	git stash git checkout master
4	Rename home.html file to index.html Add and commit the file	git mv home.html index.html git add index.html git commit -m "Renamed home.html to index.html"
5	Checkout the wholesalers branch	git checkout wholesalers

TASK 33: STASHING CONT …

6	Get the status	git status
7	List the stashes	git stash list
8	Pop the stashed changes and continue with the work.	git stash pop
9	Add and Commit the files	git add product.html git commit -m "Modified product.html to add wholesaler details"
10	Checkout the master branch	git checkout master
11	Merge the branch	git merge wholesalers
12	Push the commit	git push
13	Delete the branch wholesalers	git branch -d wholesalers

TASK 34 : APPLYING STASH TO MULTIPLE BRANCHES

- Activity:
  - You have a master branch and a feature branch called retailers .You are currently working with a page for retailers but a high priority issue have come to add copyright details to index.html . These changes need to be applied to retailers branch also. Kindly fix the issue?

- Steps:
  - Create a branch retailers and create a file retailers.html. Stash the changes
  - Checkout master and create another branch copyright . Add copyright details to index.html
  - Stash the changes and go to retailers branch
  - Apply stash to retailers branch : *git stash apply <stash ID>*
  - Add the files and commit the files
  - Push the files

- Definition of Done:
  - Stash applied across branches

TASK 35: CHERRYPICK

- Activity:
    - You have a master branch , partner branch and supplier branch. You have added a login page exclusive for partner branch. You need to have the same feature for supplier also. What should you do?
- Definition of Done:
    - We can cherry-pick a commit and add it in a new branch

#	Steps	Commands
1	Go to cherrypick folder	cd cherrypick
2	Create partner branch and supplier branch	git branch partner git branch supplier
3	Create a commit in partner branch for login feature	touch login.html git add login.html git commit -m "Added login feature"
4	Get the commit log	git log --oneline
5	Checkout supplier branch	git checkout supplier
6	Get the commit log	git log --oneline
7	Cherry-pick the login commit from the partner branch	git cherry-pick <commit id>

TASK 36 : TAGGING IN GIT

- Activity:
  - You have a lot of commits and you want to release a particular version . What should you do?
- Definition of Done:
  - Release tags are viewable in Git

#	Steps	Commands
1	Checkout the master branch	git checkout master
2	List down the commits	git log --oneline
3	Tag a commit as v1	git tag -a v1 <commit ID> -m "Release 1"
	List down the commits	git log --oneline
4	Push the tags	git push --tags

TASK 37 : HOW TO MAKE SURE THAT YOU HAVE ALL CHANGES IN THE MASTER IN YOUR LOCAL

- Activity:
  - There are a lot of commits in the GitHub repository(most probably by your team mates) which is not there in your local .
  - Kindly fix the issue.
- Definition of Done:
  - New changes in GitHub is fetched and merged with local

#	Steps	Commands
1	Create a commit in GitHub	
2	Pull the commits to your local	git pull

TASK 38: FETCH THE CHANGES FROM GITHUB

- Activity:
  - There are a lot of commits in the GitHub repository(most probably by your team mates) which is not there in your local .
  - Kindly fix the issue.
- Definition of Done:
  - Fetched details from GitHub could be reviewed

#	Steps	Commands
1	Create a commit in GitHub	
2	Fetch the commits to your local	git fetch
3	View the fetched details	git checkout -b fetchtest origin/master
4	Checkout the master	git checkout master
5	Merge the commit	git merge fetchtest
6	Delete the branch fetchtest	git branch -d fetchtest

TASK 39 : HOW TO RESET A COMMIT IN THE CENTRAL REPOSITORY (TASK 15 REVISITED)

- Activity:
  - Please do task 15 using git fetch and git pull.

TASK 40: HOW TO DIFFERENTIATE BETWEEN CHANGES BETWEEN TWO COMMITS

- Activity:
  - There are a lot of commits in the GitHub repository. You want to find differences between two commits
- Definition of Done:
  - Differences are listed out

#	Steps	Commands
1	Get the difference between two commits	git diff <commit id 1> <commit id 2>

Q & A

QUADRALOGICS

THANK YOU

QUADRALOGICS