PHANIDEEP S

SOFTWARE METHODS AND TOOLS ASSIGNMENT #9

GIT

**WORKFLOW OF GIT**

CLI (Command line interface)

Branch Local repository

Remote repository

**git init – creates a local master repository git clone – creates a remote branch**

**git add – Adds a file to index**

**git commit – Commits the data in the file**

**git pull – Pulls the data from remote repository**

**merge**

**git push – Pushes the data into remote repository**

**git checkout – Checkout from branch**

**WORKING MECHANISM OF GIT**

Shared Repository Local Repository of A

Version 1 by

A

git push

Version 1

git fetch

git commit

Index

Or staging

git add

Version 2 by

B

git pull

Version 2

area Workspace

Version 3 by

A

git merge

Version 4 by

C

…………

Git push: This command pushes the data from local repository to Shared repository in server.

Git add: This command adds the file from workspace to Index/Staging area in the local repository. Git commit: This command commits the data in the Index to a commit version 1.

Git fetch: This command fetches the data from shared repository to local repository. Git merge: This command merges the data from local repository to current workspace.

Git pull: This command pulls the data from Shared repository to workspace directly. This command is a combination of Git fetch and Git merge.

**Diagram to illustrate the working process of software project life cycle:**

Master 1 repository

Features

Release Deploy

2 3 4 5 8 6 7

Develop Testing

Hotfix

**Steps of illustration:**

1. The Features assumed for project are in features branch. These are pushed into Master repository.

2. The Features in the master repository are pulled directly into develop branch.

3. All features are merged into develop branch and if a new feature is been updated by any team member, a pull is done to develop branch.

4. The developed features are now moved to a new release branch.

5. The release is tested with test cases and a tested release is obtained.

6. The tested release is deployed into deploy branch.

7. If the customer faces any bugs/issues, maintenance can be done and a hotfix is prepared.

8. The hotfix is induced into release branch. This is been tested and deployed again to the customer.

Initially created a remote repository called RemoteRepo.git using following command

**git init –bare**, by inside RemoteRepo.git directory.

Now I have created a local repository for user1 called Localrepo1 and added Tetris games files to it using following commands

**git init** (inside Localrepo1 directory)

Added Tetris project files manually inside the directory

Then adding those files to git using following commands

**git add \*** , later this command user1 commited his changes and pushed it to repository.

**git commit –m** “First Commit of Tetris Game Code Repository”.

**git remote add origin file:///C/Users/Phanideep/Documents/GitHub/RemoteRepo.git**

now pushing it to master branch of remote repo

**git push origin master**

All right now couple of developers were added to the team they are user2 and user3.

They will clone the remore repository using following commands respectively(please go out of other user directory while executing below commands)

User2 Clone:

**git clone file:///C/Users/Phanideep/Documents/GitHub/RemoteRepo.git Localrepo2**

User3 Clone:

**git clone file:///C/Users/Phanideep/Documents/GitHub/RemoteRepo.git Localrepo3**

User2 made changes to Tetris.java and pushed to origin master

**git status -v**

**git add BoardPanel.java**

**git commit -m** "Fixed an problem in score of tetris game"

**git push origin master**

Now, user3 started working on his code he doesn't know master has been changed and without taking update he also fixed the same issue in BoardPanel.java

**git status -v** user3 has changed BoardPanel.java file.

now user3 stages the file to commit and push to origin master

**git add BoardPanel.java**

**git commit -m** "User3: fixed Board panel issue”

Now, at this point we will an error saying we need to take update from master before pushing code to it, because master in remote is ahead of one commit than user3.

To solve this we will fetch master with remote and merge will merge the changes to local

**git fetch origin**

**git merge origin/master**

Above two command solves the problem.

Now, user3 again can user push command to push his changes

**git push origin master**

Now user1 will create remote development branch, as there is only one copy of master branch So user1 will pull the new code and creates a development branch locally

**git pull origin**

**git checkout -b development**

Now to create a remote tracking branch i used following command

**git push origin development**

To see above changes use **git remote show origin**

Now other developers start using the development branch, they can track it remotely.

**git fetch origin**

**git checkout -b dev origin/development**

Now user2 made some changes in development branch

git status -v to see modified files

Now user2 will merge with his master branch

**git checkout** **master**

**git merge dev**

That's how user2 got user1 code got merged into local master.

now user2 will push local master to remote using

**git fetch origin**

**git merge origin/master**

**git push origin master.**