MODUL E -2 GIT ASSIGNMENT -1

Based on what you have learnt in the class, do the following steps:

* Create a new folder
* Put the following files in the folder

○ Code.txt

○ Log.txt

○ Output.txt

* Stage the Code.txt and Output.txt files
* Commit them
* And finally push them to github

As a solution share the github cmnds

**SOLUTION**

**mkdir fire**

**45 cd fire/**

**46 sudo nano code.txt**

**47 sudo nano log.txt**

**48 sudo output.txt**

**49 sudo nano output.txt**

**50 ls --**

**51 git add code.txt output.txt**

**52 git commit -m "this r asnmnts"**

**53 git remote -v**

**54 git remote add asgnmnts https://github.com/shoebahmedd/dryrun.git**

**55 git push asgnmnts master**

MODUL E -2 GIT ASSIGNMENT -2

Do the following tasks:

● Create a git working directory with feature1.txt and feature2.txt in the master branch

● Create 3 branches develop, feature1 and feature2

● In develop branch create develop.txt, do not stage or commit it

● Stash this file, and checkout to feature1 branch

● Create new.txt file in feature1 branch, stage and commit this file

● Checkout to develop, unstash this file and commit Please submit all the git commands used to do the above steps

**SOLUTION**

**mkdir working**

**60 pwd**

**61 git init**

**62 sudo nano feature1.txt**

**63 sudo nano feature2.txt**

**64 git branch develop**

**65 sudo git clone https://github.com/shoebahmedd/dryrun.git**

**66 ls --l**

**67 cd working/**

**68 ls --l**

**69 sudo nano feature1.txt**

**70 sudo nano feature2.txt**

**71 ls --l**

**72 sudo git clone https://github.com/shoebahmedd/dryrun.git**

**73 ls --l**

**74 sudo git add .**

**75 sudo git commit -m "this is 2nd dry test"**

**76 git branch**

**77 git branch develop**

**78 sudo git branch develop**

**79 sudo git branch feature1**

**80 sudo git branch feature2**

**81 git branch**

**82 git checkout develop**

**83 sudo git checkout develop**

**84 git branch**

**85 sudo nano develop.txt**

**86 sudo git stash**

**87 sudo git add develop.txt**

**88 ls --l**

**89 sudo git stash**

**90 ls --l**

**91 sudo git checkout feature1**

**92 sudo nano new.txt**

**93 sudo git add .**

**94 ls -l**

**95 sudo commit -m "dryrun"**

**96 sudo commit -m "this is 2nd dry test"**

**97 sudo nano new.txt**

**98 sudo git add new.txt**

**99 sudo git commit -m "cool"**

**100 ls --l**

**101 git checkout develop**

**102 sudo git checkout develop**

**103 sudo git stash pop**

**104 sudo nano develop.txt**

**105 sudo git commit -m "poped"**

MODUL E -2 GIT ASSIGNMENT -3

* Create a git working directory, with the following branches

○ Develop

○ F1

○ f2

* In the master branch, commit main.txt file
* Put develop.txt in develop branch, f1.txt and f2.txt in f1 and f2 respectively
* Push all these branches to github
* On local delete f2 branch
* Delete the same branch on Github as well

SOLUTION

1 sudo apt-get update

2 git init

3 mkdir assign3

4 cd assign3

5 git init

6 git branch develop

7 sudo nano main.txt

8 git add .

9 git commit -m "master added"

10 git branch develop

11 git branch f1

12 git branch f2

13 git branch

14 git checkout develop

15 sudo develop.txt

16 sudo nano develop.txt

17 git add .

18 git commit -m "added dev"

19 git checkout f1

20 sudo nano f1.txt

21 git add .

22 git commit -m "added f1"

23 git checkout f2

24 sudo nano f2.txt

25 git add .

26 git commit -m "added f2"

27 git remote add origin "https://github.com/shoebahmedd/assign3.git"

28 git push origin master

29 git push origin develop

30 git push origin f1

31 git push origin f2

32 git branch

33 git branch -D f2

34 git chcekout master

35 git checkout master

36 git branch -D f2

37 git branch

MODUL E -2 GIT ASSIGNMENT -4

You have been asked to:

* Put master.txt on master branch, stage and commit
* Create 3 branches: public1, public2 and private
* Put public1.txt on public 1 branch, stage and commit
* Merge public 1 on master branch
* Merge public 2 on master branch
* Edit master.txt on private branch, stage and commit
* Now update branch public 1 and public 2 with new master code in private
* Also update new master code on master
* Finally update all the code on the private branch

SOLUTION

sudo mkdir assgin4

51 cd assgin4

52 ls

54 sudo git init

55 sudo nano master.txt

sudo git add .

59 sudo git commit -m "added master"

60 sudo git branch public1

61 sudo git branch public2

62 sudo git branch private

63 sudo git checkout public1

64 sudo nano public.txt

65 git add .

66 sudo git add .

67 sudo git commit -m "added public1"

68 sudo git checkout master

69 sudo git merge public1

70 sudo git merge public2

71 sudo git checkout private

72 ls

73 sudo nano master.txt

74 git add .

75 sudo git add .

76 sudo git commit -m "modified"

77 sudo git checkout public1

78 cat master.txt

79 sudo git merge private

80 git add .

81 sudo git add .

82 sudo git commit -m "merged"

83 cat master.txt

84 sudo git checkout public2

85 cat master.txt

86 sudo git merge private

87 cat master.

87 cat master.txt

88 sudo git add .

89 sudo git commit -m "added"

90 sudo git checkout master

91 sudo git merge private

92 cat master.txt

93 git branch

94 sudo git checkout private

95 ls

96 sudo git merge public1

97 ls

98 sudo git merge public2

MODUL E -2 GIT ASSIGNMENT -5

* Create a gitflow workflow architecture on git
* Create all the required branches
* Starting from the feature branch, push the branch to the master, following the architecture
* Push a urgent.txt on master using hotfix

mkdir assign5

2 cd assign

3 cd assign5

4 git init

5 nano main.txt

6 git agg .

7 git add .

8 git commit -m "added"

9 git branch f1

10 git branch f2

11 git checkout f1

12 nano 1.txt

13 git add .

14 git commit -m "added"

15 git checkout f2

16 nano 2.txt

17 git add .

18 git commit -m "added"

19 git checkout master

20 git branch inter

21 git checkout inter

22 git merge f1

23 git merge f2

24 git checkout master

25 git merge inter

26 ls

27 git branch hotfix

28 git checkout hotfix

29 ls

30 nano urgent.txt

31 git add .

32 git commit -m "added"

33 git checkout master

34 git merge hotfix

35 ls

CASE STUDY - RESOLVING MERGE CONFLICTS

**Problem Statement:**

You work for Zendrix Software & Co. You have been assigned the task of updating the Master branch of their Git repository with all the features from the feature branches.

Following is the GitHub account, <https://github.com/devops-intellipaat/merge-conflict.git>

Consider,

* Feature1 branch to be a public branch
* Feature2 branch to be a private branch





The company relies on a monolithic architecture, and for now all the code resides in one file

“main.c”.

The respective features have been added in the feature branches for main.c.

Meanwhile, a security patch was made to the master branch, and now feature1 and

feature2 branches are behind from master by 1 commit.

**Following tasks have to be done:**

1. Update Feature1 and Feature2 branch with the Security Patch
2. Apply changes of Feature1 and Feature2 branch on master
3. Finally push all the branches to GitHub

SOLUTION

1 sudo apt-get update

2 clear

3 git init

4 git clone "https://github.com/shoebahmedd/merge-conflict.git"

5 ls

6 cd merge-conflict

7 ls

8 cat main.c

9 git checkout feature1

10 ls

11 cat main.c

12 git merge master

13 git mergetool

14 git add .

15 git commit -m "added"

16 cat main.c

17 git checkout feature2

18 ls

19 cat main.c

20 git rebase master

21 git mergetool

22 git rebase --continue

23 git add .

24 git commit -m "added"

25 git status

26 git pull

27 git add .

28 git commit -m "merged"

29 cat main.c

30 git checkout master

31 git merge feature1

32 git merge feature2

33 git mergetool

34 git add .

35 git commit -m "added"

36 cat main.c

37 git push origin master

38 git push origin feature1

39 git push origin feature2

CAPSTONE PROJECT

Integration of Devops tools with Jenkins

You have been Hired as a Devops Engineer in xyz software company. They want to implement CI/CD pipeline in their company. You have been asked to implement this lifecycle as fast as possible. As this is a product-based company, their product is available on this GitHub link. https://github.com/hshar/website.git Following are the specifications of the Continuous integration:

1. Git Workflow has to be implemented

2. Code Build should automatically be triggered once commit is made to master branch or develop branch. If commit is made to master branch, test and push to prod. If commit is made to develop branch, just test the product, do not push to prod.

3. Create a pipeline for the above tasks.

4. Create a container with Ubuntu and apache installed in it and use that container to build the code and the code should be on ‘/var/www/html’.

SOLUTION

1. GIT workflow has to be implemented
2. Launch two instances
3. Install Jenkins ,Docker,Java
4. Then add public key and private key copy to files to connect to instances ( i.e connection between Master and agent)
5. Create the Git Workflow
6. Clone the reprositories
7. Add to the Working reprository
8. Create a branch develop
9. Create docker file on both branch
10. Code should be automatically triggered
11. Enable webhook and payload url with Jenkins url
12. Go to Jenkins dashboard create new project Master
13. Again one more item master-build
14. Under build add exec steps (i.e execute shell)
15. Connect master and agent
16. Now under agent instance push(I.e check trigger)
17. Go to master build add cmd in exec shell to check on website for working fine
18. Same create another project develop and develop build
19. Create an pipeline for above task
20. Install respected plugins
21. Pipeline view can be shown
22. Above task has been completed

EXECUTION COMMANDS

Master:

1 sudo apt-get update

2 sudo apt-get install openjdk-8-jdk -y

3 sudo apt-get install docker.io -y

4 wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -

5 sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > \

/etc/apt/sources.list.d/jenkins.list'

6 sudo apt-get update

7 sudo apt-get install jenkins

8 sudo cat /var/lib/jenkins/secrets/initialAdminPassword

9 cd .ssh

10 ssh-keygen

11 ls

12 cat id\_rsa.pub

13 sudo nano authorized\_keys

14 sudo nano id\_rsa

15 sudo cat id\_rsa

16 history

Slave:

cd .ssh

ls

sudo nano authorized\_keys

/\* paste the key from master\*/

sudo su

cd

pwd

/root# cd .ssh

ls

sudo nano authorized\_keys

/\* paste the keys from master \*/

\*\*\*\*\*\*ON Slave SERVER\*\*\*\*\*\*\*

exit root user

/,ssh$ cd

ls

#jenkins

cd jenkins

mkdir test

cd

sudo git clone " "

ls

cd website/

ls

sudo mv ./image/ /home/ubuntu/jenkins/test/

sudo mv ./index.html /home/ubuntu/jenkins/test/

cd

cd jenkins/test

ls

sudo git add .

sudo git commit -m " "

git branch develop

git repo:

https://github.com/hshar/website.git

