**SWE 525 GIT Version Control Mid Term Exam 08/21/2016**

**ITU Student Name: Supriya Nittur Kumar**

**ITU Student ID: 89706**

1. Perform the following steps on any repository of your choice and push it to remote. Explain your work and paste your github repo address. Make me collaborator. Draw flow chart if applicable.
2. Create a repository

Answer: created a repository “midtermgit” on github and used git clone <https://github.com/Supriyankumar/midtermgit.git> which now has a local copy on my computer as midtermgit folder.

Last login: Sun Aug 21 09:27:01 on console

SUPRIYAs-MacBook-Pro:~ suppi$ cd desktop

SUPRIYAs-MacBook-Pro:desktop suppi$ cd SWE 525

-bash: cd: SWE: No such file or directory

SUPRIYAs-MacBook-Pro:desktop suppi$ SWE525

-bash: SWE525: command not found

SUPRIYAs-MacBook-Pro:desktop suppi$ cd SWE525

SUPRIYAs-MacBook-Pro:SWE525 suppi$ git clone https://github.com/Supriyankumar/midtermgit.git

Cloning into 'midtermgit'...

remote: Counting objects: 3, done.

remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0

Unpacking objects: 100% (3/3), done.

SUPRIYAs-MacBook-Pro:SWE525 suppi$ cd midtermgit

SUPRIYAs-MacBook-Pro:midtermgit suppi$ ls

README.md

1. Make some changes -- add some files, change content of some files

Answer:

SUPRIYAs-MacBook-Pro:midtermgit suppi$ ls

README.md

SWE 525 GIT Version Control Mid Term Exam 08.docx

SUPRIYAs-MacBook-Pro:midtermgit suppi$ echo "#This is a new test file" > file1

SUPRIYAs-MacBook-Pro:midtermgit suppi$ ls

README.md

SWE 525 GIT Version Control Mid Term Exam 08.docx

file1

SWE 525 GIT Version Control Mid Term Exam 08.docx

file1

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git add file1

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git commit -m 'add new file1'

[master 84c148b] add new file1

1 file changed, 1 insertion(+)

create mode 100644 file1

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git status

# On branch master

# Your branch is ahead of 'origin/master' by 1 commit.

#

# Untracked files:

# (use "git add <file>..." to include in what will be committed)

#

# SWE 525 GIT Version Control Mid Term Exam 08.docx

nothing added to commit but untracked files present (use "git add" to track)

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git add file1

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git commit -m 'new line added'

[master cffb938] new line added

1 file changed, 1 insertion(+)

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git status

# On branch master

# Your branch is ahead of 'origin/master' by 2 commits.

#

# Untracked files:

# (use "git add <file>..." to include in what will be committed)

#

# SWE 525 GIT Version Control Mid Term Exam 08.docx

nothing added to commit but untracked files present (use "git add" to track)

1. Commit changes

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git commit -m 'add new file1'

[master 84c148b] add new file1

1 file changed, 1 insertion(+)

create mode 100644 file1

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git status

# On branch master

# Your branch is ahead of 'origin/master' by 1 commit.

#

# Untracked files:

# (use "git add <file>..." to include in what will be committed)

#

# SWE 525 GIT Version Control Mid Term Exam 08.docx

nothing added to commit but untracked files present (use "git add" to track)

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git add file1

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git commit -m 'new line added'

[master cffb938] new line added

1 file changed, 1 insertion(+)

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git status

# On branch master

# Your branch is ahead of 'origin/master' by 2 commits.

#

# Untracked files:

# (use "git add <file>..." to include in what will be committed)

#

# SWE 525 GIT Version Control Mid Term Exam 08.docx

nothing added to commit but untracked files present (use "git add" to track)

1. repeat steps 1-3 for 2 more times. try to remove and ignore some files . Use git diff before each commit.

Answer:

SUPRIYAs-MacBook-Pro:midtermgit suppi$ touch file2.txt

SUPRIYAs-MacBook-Pro:midtermgit suppi$ ls

README.md

SWE 525 GIT Version Control Mid Term Exam 08.docx

file1

file2.txt

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git add file2.txt

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git status

# On branch master

# Your branch is ahead of 'origin/master' by 2 commits.

#

# Changes to be committed:

# (use "git reset HEAD <file>..." to unstage)

#

# new file: file2.txt

#

# Untracked files:

# (use "git add <file>..." to include in what will be committed)

#

# SWE 525 GIT Version Control Mid Term Exam 08.docx

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git add SWE\_525\_GIT\_Version\_Control\_Mid\_Term\_Exam\_08.docx

fatal: pathspec 'SWE\_525\_GIT\_Version\_Control\_Mid\_Term\_Exam\_08.docx' did not match any files

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git add Answersgitmidterm.docx

SUPRIYAs-MacBook-Pro:midtermgit suppi$ ls

Answersgitmidterm.docx

README.md

SWE 525 GIT Version Control Mid Term Exam 08.docx

file1

file2.txt

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git add file2

fatal: pathspec 'file2' did not match any files

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git add file2.txt

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git diff file2.txt

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git add file2.txt

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git diff file2.txt

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git status

# On branch master

# Your branch is ahead of 'origin/master' by 2 commits.

#

# Changes to be committed:

# (use "git reset HEAD <file>..." to unstage)

#

# new file: Answersgitmidterm.docx

# new file: file2.txt

#

# Untracked files:

# (use "git add <file>..." to include in what will be committed)

#

# SWE 525 GIT Version Control Mid Term Exam 08.docx

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git add file2.txt

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git diff file2.txt

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git status

# On branch master

# Your branch is ahead of 'origin/master' by 2 commits.

#

# Changes to be committed:

# (use "git reset HEAD <file>..." to unstage)

#

# new file: Answersgitmidterm.docx

# new file: file2.txt

#

# Untracked files:

# (use "git add <file>..." to include in what will be committed)

#

# SWE 525 GIT Version Control Mid Term Exam 08.docx

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git commit -m 'new changes made'

[master b902a62] new changes made

2 files changed, 8 insertions(+)

create mode 100644 Answersgitmidterm.docx

create mode 100644 file2.txt

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git diff file2.txt

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git status

# On branch master

# Your branch is ahead of 'origin/master' by 3 commits.

#

# Untracked files:

# (use "git add <file>..." to include in what will be committed)

#

# SWE 525 GIT Version Control Mid Term Exam 08.docx

nothing added to commit but untracked files present (use "git add" to track)

SUPRIYAs-MacBook-Pro:midtermgit suppi$

SUPRIYAs-MacBook-Pro:midtermgit suppi$ touch .gitignore

SUPRIYAs-MacBook-Pro:midtermgit suppi$ ls

Answersgitmidterm.docx

README.md

SWE 525 GIT Version Control Mid Term Exam 08.docx

file1

file2.txt

SUPRIYAs-MacBook-Pro:midtermgit suppi$ ls -a

.

..

.git

.gitignore

Answersgitmidterm.docx

README.md

SWE 525 GIT Version Control Mid Term Exam 08.docx

file1

file2.txt

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git rm --cached .gitignore

fatal: pathspec '.gitignore' did not match any files

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git add .gitignore

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git rm --cached .gitignore

rm '.gitignore'

SUPRIYAs-MacBook-Pro:midtermgit suppi$

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git status

# On branch master

# Changes not staged for commit:

# (use "git add <file>..." to update what will be committed)

# (use "git checkout -- <file>..." to discard changes in working directory)

#

# modified: Answersgitmidterm.docx

#

# Untracked files:

# (use "git add <file>..." to include in what will be committed)

#

# .gitignore

# SWE 525 GIT Version Control Mid Term Exam 08.docx

no changes added to commit (use "git add" and/or "git commit -a")

SUPRIYAs-MacBook-Pro:midtermgit suppi$

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git add Answersgitmidterm.docx

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git commit -m 'added more answers for remove and ignore'

[master 22f4ef4] added more answers for remove and ignore

1 file changed, 0 insertions(+), 0 deletions(-)

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git diff Answersgitmidterm

fatal: ambiguous argument 'Answersgitmidterm': unknown revision or path not in the working tree.

Use '--' to separate paths from revisions

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git diff Answersgitmidterm.docx

SUPRIYAs-MacBook-Pro:midtermgit suppi$

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git add file2.txt

SUPRIYAs-MacBook-Pro:midtermgit suppi$ echo '# test line' >> file2.txt

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git status

# On branch master

# Changes not staged for commit:

# (use "git add <file>..." to update what will be committed)

# (use "git checkout -- <file>..." to discard changes in working directory)

#

# modified: Answersgitmidterm.docx

# modified: file2.txt

#

# Untracked files:

# (use "git add <file>..." to include in what will be committed)

#

# .gitignore

# SWE 525 GIT Version Control Mid Term Exam 08.docx

no changes added to commit (use "git add" and/or "git commit -a")

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git diff

diff --git a/Answersgitmidterm.docx b/Answersgitmidterm.docx

index 549aa61..450de80 100644

Binary files a/Answersgitmidterm.docx and b/Answersgitmidterm.docx differ

diff --git a/file2.txt b/file2.txt

index 0f3bb87..7112a74 100644

--- a/file2.txt

+++ b/file2.txt

@@ -5,4 +5,4 @@ Here are the new added lines into this file

This is the third change in the file

-This is the fifth change in the file.

\ No newline at end of file

+This is the fifth change in the file.# test line

SUPRIYAs-MacBook-Pro:midtermgit suppi$

1. create changes and revert those changes using git when those changes are not added to staging area?

Answers: SUPRIYAs-MacBook-Pro:midtermgit suppi$ git commit --amend

added more answers for remove and ignore

# Please enter the commit message for your changes. Lines starting

# with '#' will be ignored, and an empty message aborts the commit.

# On branch master

# Changes to be committed:

# (use "git reset HEAD^1 <file>..." to unstage)

#

# modified: Answersgitmidterm.docx

#

# Changes not staged for commit:

# (use "git add <file>..." to update what will be committed)

# (use "git checkout -- <file>..." to discard changes in working directory)

#

# modified: Answersgitmidterm.docx

# modified: file2.txt

#

# Untracked files:

# (use "git add <file>..." to include in what will be committed)

#

# .gitignore

# SWE 525 GIT Version Control Mid Term Exam 08.docx

~

".git/COMMIT\_EDITMSG" 22L, 706C

1. create changes and revert those changes after adding changes to staging area?

Answer:

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git add \*

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git status

# On branch master

# Your branch and 'origin/master' have diverged,

# and have 1 and 1 different commit each, respectively.

#

# Changes to be committed:

# (use "git reset HEAD <file>..." to unstage)

#

# modified: Answersgitmidterm.docx

# new file: SWE 525 GIT Version Control Mid Term Exam 08.docx

# modified: file2.txt

#

# Untracked files:

# (use "git add <file>..." to include in what will be committed)

#

# .gitignore

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git reset HEAD file2.txt

Unstaged changes after reset:

M file2.txt

SUPRIYAs-MacBook-Pro:midtermgit suppi$

1. create change and revert those changes after committing those changes .

8. Check repository status before adding changes to staging area, after adding changes to staging area and after committing changes .

9. Display all the commits .

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git log

commit 22f4ef476387a6692c53f87cbb0748b882a8b938

Author: Supriya Kumar <supriya.nkumar@gmail.com>

Date: Sun Aug 21 13:20:43 2016 -0700

added more answers for remove and ignore

commit 732dd582d51faf200939265f9c0e4015ce1980ee

Author: Supriya Kumar <supriya.nkumar@gmail.com>

Date: Sun Aug 21 13:02:39 2016 -0700

new answers

commit b902a626648a3134596063e9a4f77000b07b101b

Author: Supriya Kumar <supriya.nkumar@gmail.com>

Date: Sun Aug 21 13:00:36 2016 -0700

new changes made

commit cffb938d25e9ff23f8244b948e00613d934a3590

Author: Supriya Kumar <supriya.nkumar@gmail.com>

Date: Sun Aug 21 12:44:51 2016 -0700

new line added

:

10. Display a specific commit .

Answer: SUPRIYAs-MacBook-Pro:midtermgit suppi$ git log -Snew

commit b902a626648a3134596063e9a4f77000b07b101b

Author: Supriya Kumar <supriya.nkumar@gmail.com>

Date: Sun Aug 21 13:00:36 2016 -0700

new changes made

commit 84c148b817bcddfff0414ee59989f400f7b36008

Author: Supriya Kumar <supriya.nkumar@gmail.com>

Date: Sun Aug 21 12:42:05 2016 -0700

add new file1

SUPRIYAs-MacBook-Pro:midtermgit suppi$

11. create branches called "feature1" and "feature2" from master.

Answers: SUPRIYAs-MacBook-Pro:midtermgit suppi$ git branch

\* master

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git branch feature1

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git branch feature2

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git branch

feature1

feature2

\* master

SUPRIYAs-MacBook-Pro:midtermgit suppi$

12. Delete branch "feature2".

Answer: SUPRIYAs-MacBook-Pro:midtermgit suppi$ git branch -d feature2

Deleted branch feature2 (was a54132a).

SUPRIYAs-MacBook-Pro:midtermgit suppi$

13. Make some commits on "feature1".

14. Make some commits on branch "master".

15. merge branch "feature1" in "master" (make some conflicts by changing same file on same line in both the branches)

16. create branch "feature2" from master .

17. Make some commits on "feature2".

18. Make some commits on branch "master".

19. rebase branch "master" in "feature2" (make some conflicts by changing same file on same line in both the branches) .

20. find the differences between "merge" and "rebase" in above cases.

21. Use git stash

22. create a repository on github for your working repository and push changes on remote repository.

2. List out the key difference between a centralized version control system and distributed version control system.

Answer:

**Centralized Version Control System**:

Provides a server software component which stores and manages the different versions of the files and let developer copy(checkout) a certain version onto their individual computer.

There is a single point of failure in centralized version control system. The server machine is a single point of failure here.

It gets harder to work in parallel on different features.

**Distributed Version Control system**:

Each user has a complete copy of a repository on his individual computer. The user can copy an existing repository.

There is a central server for keeping a repository but each cloned repository is a full copy of this repository. There is no problem of a single system failure since a copy of the repository is saved locally and uploaded only what is required.

Distributed version control system makes it easier to work in parallel on different features.

List down any two centralized version control system and 2 distributed version control system.

Two Centralized Version control system: Perforce, Subversion

Two Distributed Version Control system: git, mercurial

What are the advantages of git VCS over other VCS.

The main difference between Git and any other VCS is the way Git thinks about its data. Git thinks of its data more like set of snapshots of a miniature filesystem.

Git maintains a set of configuration values within each repository. Git manages and inspects configuration and setup information on a per-site, per-user, and per-repository basis

What are the different states of a file in the Git VCS.

Committed, modified and staged. Committed means, that the data is safely stored in your local database. Modified means that you have changed the file but have not committed it to your database yet. Staged means that you have marked a modified file in its current version to go into your next commit snapshot.

* 3. Answer following short questions:
* Question1: How to delete a Git branch both locally and remotely?
* Answer:

1. To remove a local branch from your local system.

Answer: git branch -d branchName

1. To remove a remote branch from the server.

git push origin --delete branchName

* Question 2: How do you undo the last commit?
* Answer: using git reset -- soft HEAD ~ 1
* Question 3: How to Edit an incorrect commit message in Git?
* Answer: git commit --amend –m “Message here”
* Question 4: What are the differences between 'git pull' and 'git fetch'?
* Answer: git pull runs git fetch with the given parameters and calls git merge to merge the retrieved branch heads into the current branch.
* Git fetch only pull the data and should be externally followed by git merge.
* Question 5: How do you rename the local branch?
* Answer: git branch –m <old\_name> <new\_name>
* Question 6: How do I remove local files (Not in Repo) from my current Git
* Answer: git clean –f -n
* Question 7: How to Checkout remote Git branch?
* Answer: git checkout -b test origin/test
* Question 8: How do you create a remote Git branch?
* Answer: git branch <branch\_name>
* git push <remote\_name> <branch\_name>
* Question 9: How to Change the URL for a remote Git repository?

Answer: git remote set-url origin git://new.url.here

* Question 10: How to Change the author of a commit in Git?
* git commit --amend --author "New Author Name <email@address.com>"
* Question 11: Explain different states of a file.
* Committed means, that the data is safely stored in your local database. Modified means that you have changed the file but have not committed it to your database yet.
* Staged means that you have marked a modified file in its current version to go into your next commit snapshot.
* What is difference between Untracked, Tracked, Unstaged, Commited, modified, unmodified and Uncommited files
* Question 12: How a remote branch can be tracked by Git branch
* Question 13: How to stash only one file out of multiple files that have changed
* Question 14: I didn't ask you to do this, but suppose I asked you to make a commit on the 'other' branch. If I wanted you to push this branch, what command would you use?
* Question 15: Why do we call the place we put stashes a "stack"? What git stash operation can we do that is usually impossible with stacks?

1. Run the following commands on a given repo. Copy your git log below. Also upload your repo to remote repository on github and paste your link to the github repo. Also Draw a git Flow Diagram for following set of commands. You can draw it manually with hand and then upload the photo to EMS with Question number as file name.
2. git commit –m “A”
3. git commit –m “B”
4. git branch stable
5. git checkout –b experiment
6. git commit –m “C”
7. git checkout master
8. git commit –m “D”
9. gitbranchgoodidea
10. gitcheckout experiment
11. git branch whereami
12. git commit –m “E”
13. git checkout goodidea
14. git checkout master
15. git commit –m “F”
16. git checkout whereami
17. gitcommit–m“G”

git checkout master

Answer:

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git commit -m "A"

# On branch master

# Changes not staged for commit:

# (use "git add <file>..." to update what will be committed)

# (use "git checkout -- <file>..." to discard changes in working directory)

#

# modified: Answersgitmidterm.docx

#

no changes added to commit (use "git add" and/or "git commit -a")

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git commit -m "B"

# On branch master

# Changes not staged for commit:

# (use "git add <file>..." to update what will be committed)

# (use "git checkout -- <file>..." to discard changes in working directory)

#

# modified: Answersgitmidterm.docx

#

no changes added to commit (use "git add" and/or "git commit -a")

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git branch stable

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git checkout -b experiment

M Answersgitmidterm.docx

Switched to a new branch 'experiment'

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git commit -m "C"

# On branch experiment

# Changes not staged for commit:

# (use "git add <file>..." to update what will be committed)

# (use "git checkout -- <file>..." to discard changes in working directory)

#

# modified: Answersgitmidterm.docx

#

no changes added to commit (use "git add" and/or "git commit -a")

SUPRIYAs-MacBook-Pro:midtermgit suppi$

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git checkout master

M Answersgitmidterm.docx

Switched to branch 'master'

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git commit -m "D"

# On branch master

# Changes not staged for commit:

# (use "git add <file>..." to update what will be committed)

# (use "git checkout -- <file>..." to discard changes in working directory)

#

# modified: Answersgitmidterm.docx

#

no changes added to commit (use "git add" and/or "git commit -a")

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git branch goodidea

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git checkout experiment

M Answersgitmidterm.docx

Switched to branch 'experiment'

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git branch whereami

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git commit -m "E"

# On branch experiment

# Changes not staged for commit:

# (use "git add <file>..." to update what will be committed)

# (use "git checkout -- <file>..." to discard changes in working directory)

#

# modified: Answersgitmidterm.docx

#

no changes added to commit (use "git add" and/or "git commit -a")

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git commit -m "E"

# On branch experiment

# Changes not staged for commit:

# (use "git add <file>..." to update what will be committed)

# (use "git checkout -- <file>..." to discard changes in working directory)

#

# modified: Answersgitmidterm.docx

#

no changes added to commit (use "git add" and/or "git commit -a")

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git checkout goodidea

M Answersgitmidterm.docx

Switched to branch 'goodidea'

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git checkout master

M Answersgitmidterm.docx

Switched to branch 'master'

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git commit -m "F"

# On branch master

# Changes not staged for commit:

# (use "git add <file>..." to update what will be committed)

# (use "git checkout -- <file>..." to discard changes in working directory)

#

# modified: Answersgitmidterm.docx

#

no changes added to commit (use "git add" and/or "git commit -a")

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git commit -m "F"

# On branch master

# Changes not staged for commit:

# (use "git add <file>..." to update what will be committed)

# (use "git checkout -- <file>..." to discard changes in working directory)

#

# modified: Answersgitmidterm.docx

#

no changes added to commit (use "git add" and/or "git commit -a")

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git checkout whereami

M Answersgitmidterm.docx

Switched to branch 'whereami'

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git commit -m "G"

# On branch whereami

# Changes not staged for commit:

# (use "git add <file>..." to update what will be committed)

# (use "git checkout -- <file>..." to discard changes in working directory)

#

# modified: Answersgitmidterm.docx

#

no changes added to commit (use "git add" and/or "git commit -a")

SUPRIYAs-MacBook-Pro:midtermgit suppi$ git checkout master

M Answersgitmidterm.docx

Switched to branch 'master'

SUPRIYAs-MacBook-Pro:midtermgit suppi$

9.Explain what is merge? Explain merge strategies? What is merge conflict? Explain how different ways of resolving merge conflict? Give an example of merge conflict please annotate and document your git command flow below. i.e. Create a branch/es, create a conflict and resolve it using git command. Show your work.

**Answer:** git merge is used to join two or more development histories together.

Merge strategies: git merge -s recursive -X ours : This option directs Git to resolve conflicting changes in favor of the current branch. This is different from the ours strategy, in that nonconflicting changes can still be resolved in favor of either branch. You can use -X theirs as well, to resolve in favor of the other branch instead.

git merge -s ours: This leaves the content on your branch unchanged, and when you next merge from the other branch, Git will only consider changes made from this point forward.

ignore-space-change, ignore-all-space, ignore-space-at-eol

These options to the recursive strategy automatically resolve conflicts differing only in certain types of whitespace

“octopus” strategy

Can merge any number of branches, but only if all changes can be resolved automatically.

subtree

This is a modified recursive strategy. When merging trees A and B, if B corresponds to a subtree of A, B is first adjusted to match the tree structure of A, instead of reading the trees at the same level. This adjustment is also done to the common ancestor tree.

Renormalize

This runs a virtual check-out and check-in of all three stages of a file when resolving a three-way merge. This option is meant to be used when merging branches with different clean filters or end-of-line normalization rules.

Merge conflict: When two files cannot be added by git automatically when merge command is run results in a merge conflict. Merge conflict A conflict during a merge operation occurs if two commits from different branches have modified the same content and Git cannot automatically determine how both changes should be combined when merging these branches.

This happens for example if the same line in a file has been replaced by two different commits.If a conflict occurs, Git marks the conflict in the file and the programmer has to resolve the conflict manually.After resolving it, he adds the file to the staging area and commits the change. These steps are required to finish the merge operation.

Resolving merge conflicts:

Decide not to merge. The only clean-ups you need are to reset the index file to the HEAD commit to reverse 2. and to clean up working tree changes made by 2. and 3.; git merge --abort can be used for this.Resolve the conflicts. Git will mark the conflicts in the working tree. Edit the files into shape and git add them to the index. Use git commit to seal the deal.You can work through the conflict with a number of tools:Use a mergetool. git mergetool to launch a graphical mergetool which will work you through the merge.Look at the diffs. git diff will show a three-way diff, highlighting changes from both the HEAD and MERGE\_HEAD versions.Look at the diffs from each branch. git log --merge -p <path> will show diffs first for the HEAD version and then the MERGE\_HEAD version.Look at the originals. git show :1:filename shows the common ancestor, git show :2:filename shows the HEAD version, and git show :3:filename shows the MERGE\_HEAD version.

Example:

Create a new repository:

$ cd git-recipes

$ git init 09-01

$ cd 09-01

and then follow these steps:

1. Create the file numbers.txt with the contents 1 2 3. The numbers.txt file created in the master branch

1

2

3

2. Commit numbers.txt file with $ git snapshot Numbers: 1, 2, 3

3. Create the branch named en with $ git branch en

4. Create the branch named fr with $ git branch fr

5. Create a new commit in the en branch

a. Switch to the en branch with $ git checkout en

b. Change the contents of numbers.txt. Replace 2 with two

The numbers.txt file commited in the en branch

1

two

3

c. Commit the change with $ git snapshot Numbers: two

6. Create a new commit in the fr branch

a. Switch to the fr branch with $ git checkout fr

b. Change the contents of numbers.txt. Replace 2 with deux

The numbers.txt file commited in the fr branch

1

deux

3

c. Commit the change with $ git snapshot Numbers: deux

Finish the recipe with $ git checkout en.

Solution:

$ cd git-recipes

$ git clone-with-branches 09-01 09-02

$ cd 09-02

$ git checkout en

Your current branch now is en. Merge the fr branch into en with $ git merge fr. This time the automatic merge

fails with the following message:

Auto-merging numbers.txt

CONFLICT (content): Merge conflict in numbers.txt

Automatic merge failed; fix conflicts and then commit the result.

As you can see the merging was paused. You have to fix conflicts and then commit the result. The output of

$ git status -s is:

UU numbers.txt

The conflicted files are labeled with UU, which according to $ git status --help stands for updated but

unmerged. The contents of numbers.txt right after $ git merge fr

1

<<<<<<< HEAD

two

=======

deux

>>>>>>> fr

3

Right now, you have to edit the file and prepare the contents that you regard as the proper solution for the

conflict. You can use any editor you like, and you can insert any contents you like. Open the file numbers.txt and

change it accordingly.

The contents of numbers.txt file with manually edited contents

1

two - deux

3

When the file is saved you can verify that its status did not change. The command $ git status -s returns the

same output as before: UU numbers.txt.

Once you have manually resolved the conflict you can change the status of the file from UU into M\_. This is done

with the $ git add numbers.txt command. Committing the change with the $ git commit --no-edit command,

you will finish the recipe.

1. Write a series of git commands (similar to what was given in the previous question) that would produce the following git graph. You may omit edit and git add commands for brevity. Make sure that branches and HEAD end up on the right commits by the end

