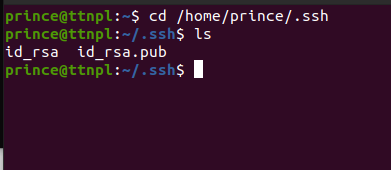
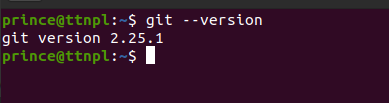
Q1.Git Setup

Soln

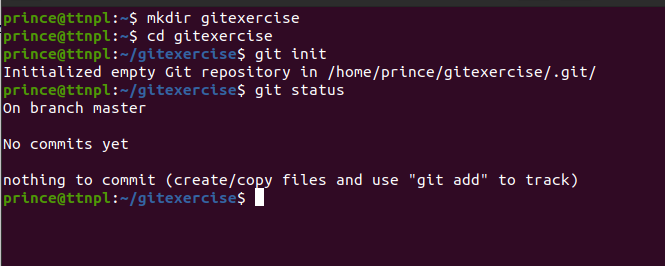
1. Generated SSH Key



1. 

Q2. Initialize a Git Repository.

Soln



$mkdir gitexercise

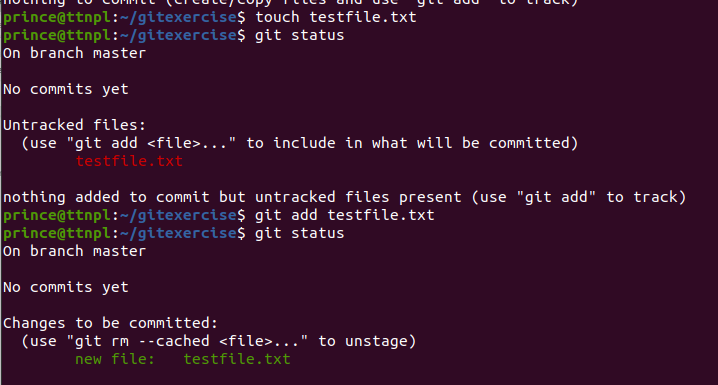
$cd gitexercise

$git init

$git status

Q3.Add files to the repository.

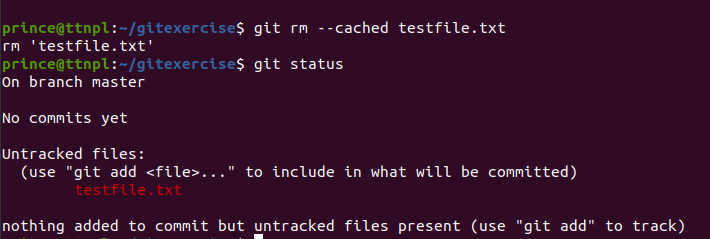
Soln



$git add testfile.txt

Q4.Unstage 1 file.

Soln

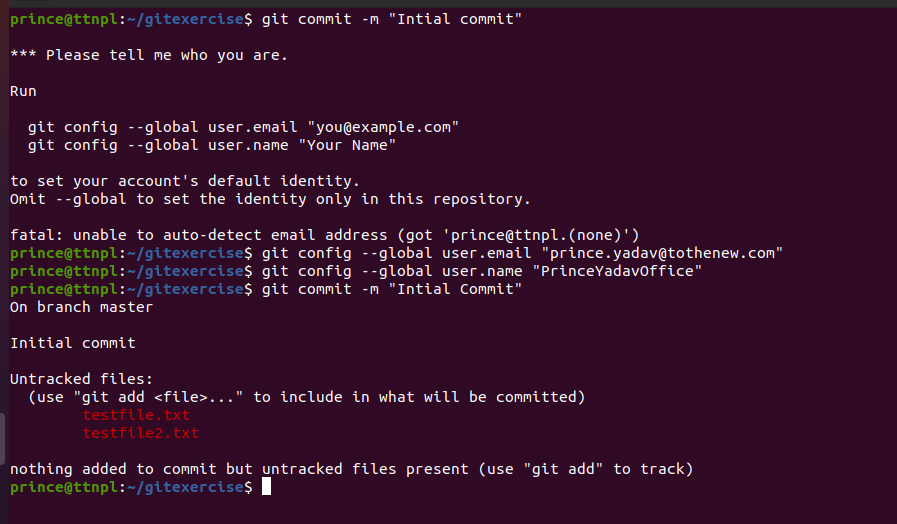


$git rm --cached testfile.txt

$git status

Q5.Commit the file.

Soln.



Q6.Add a remote.

Soln



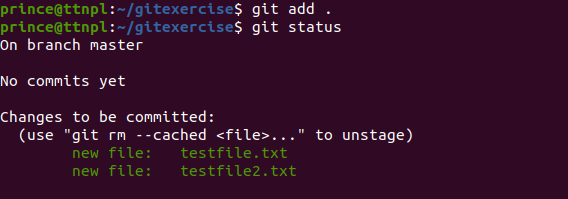
$git remote add origin <https://github.com/PrinceYadavOffice/myassignmentsrepo.git>

Q7.Undo changes to a particular file.

Soln

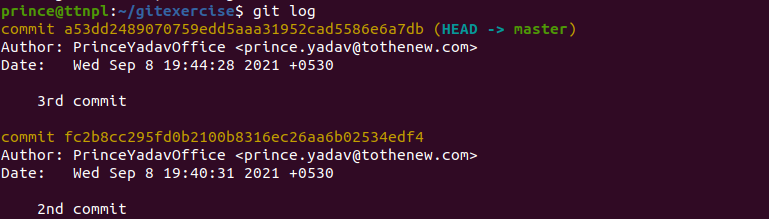


$cat testfile.txt

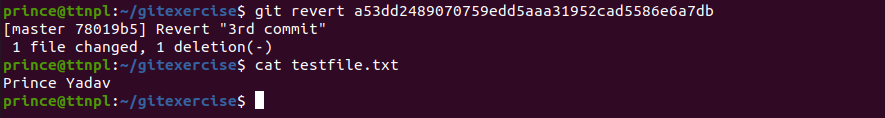


$git add.

$git status



$git log

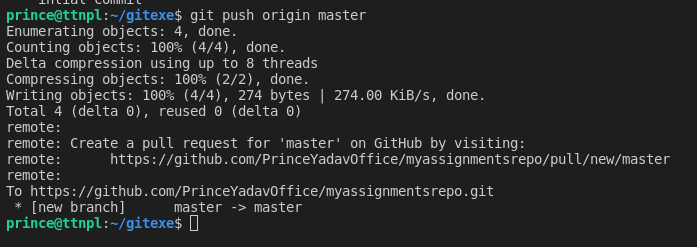


$git revert a53dd2489070759edd5aaa31952cad5586e6a7db

$cat testfile.txt

Q8.Push Changes to github.

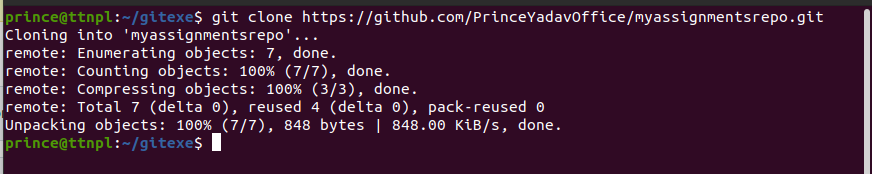
Soln



$git push origin master

Q9. Clone the repository.

Soln



$git clone <https://github.com/PrinceYadavOffice/myassignmentsrepo.git>

Q10.Add changes to one of the copies and pull the changes in the other.

Soln

Que 11: Check differences between a file and its staged version

Soln:

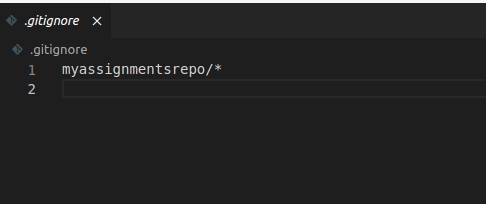
A file exists locally in a directory, but isn’t a part of the Git repository. It’s change history is not maintained and it will not be pushed to the repository when the $ git push command is executed.

While,

Staged version of a file will be committed when we run $ git commit. On the next commit, staged changes become part of Git history. $ git status will not list them as changes since they’re part of the last commit now.

Q12.Ignore a few files to be checked in.

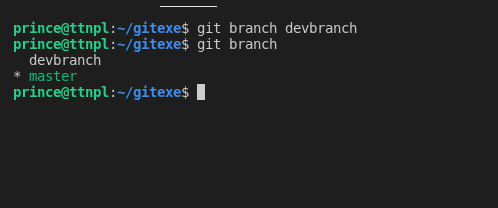
Soln.



By using **.gitignore** we can ignore files which we do not want to upload on remote repo.

Q13.Create new branch.

Soln.

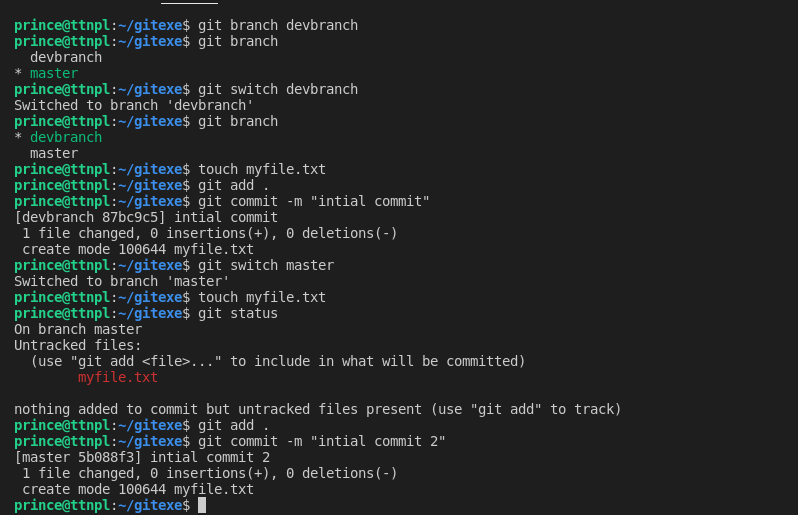


$git branch devbranch

$git branch

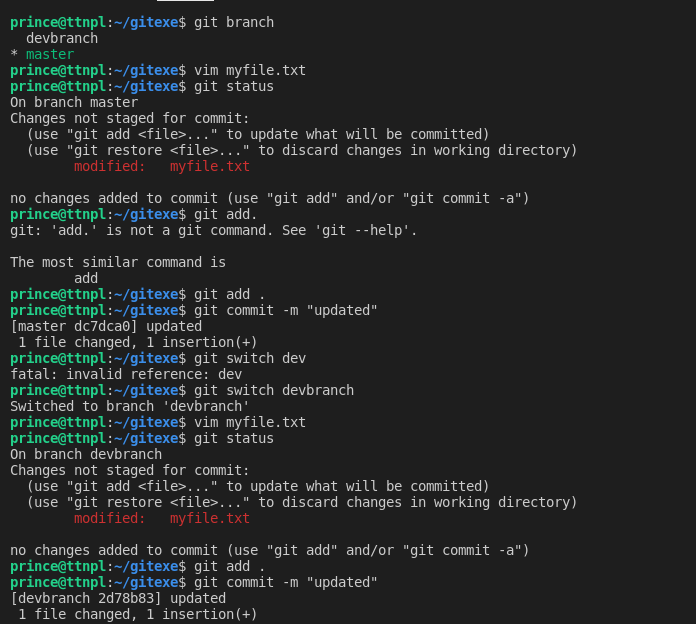
Q14.Diverge them with commits.

Soln



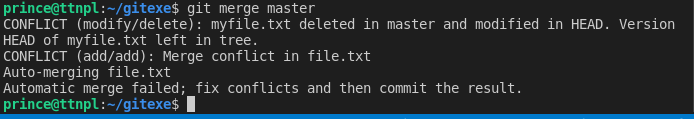
Q15.Edit the same file at the same line on both branches and commit.

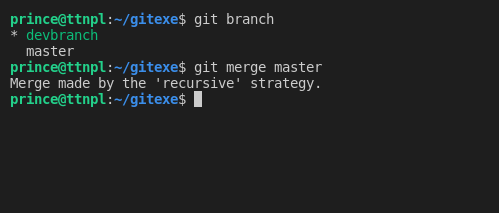
Soln



Q16.Try merging and resolve merge conflicts.

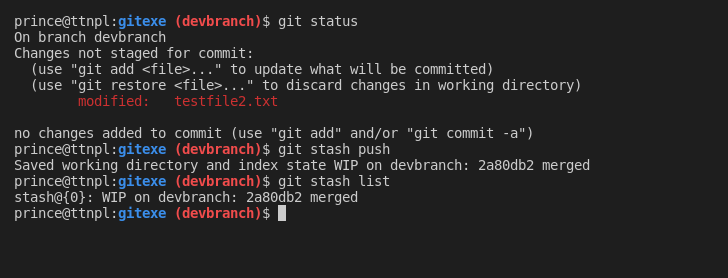
Soln





Q17.Stash the changes and pop them.

Soln.



Q18.Add the following code to your .bashrc file : color\_prompt="yes"  
parse\_git\_branch() {  
git branch 2> /dev/null | sed -e '/^[^\*]/d' -e 's/\* \(.\*\)/(\1)/'  
}  
if [ "$color\_prompt" = yes ]; then  
PS1='\u@\h\[\033[00m\]:\[\033[01;34m\]\W\[\033[01;31m\] $(parse\_git\_branch)\[\033[00m\]\$ '  
else  
PS1='\u@\h:\W $(parse\_git\_branch)\$ '  
fi  
unset color\_prompt force\_color\_prompt

Soln.

