Exercise-2

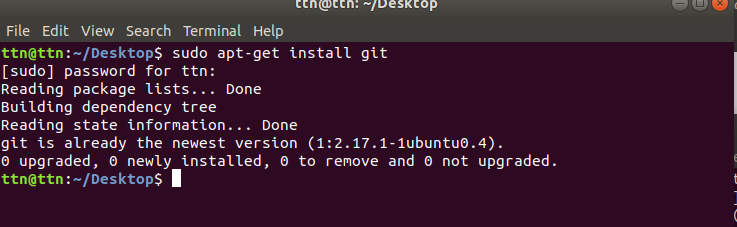
Introduction to Version Control & Git

Date:05-02-2019

1. Git Setup<https://confluence.atlassian.com/bitbucket/set-up-git-744723531.html>.

Ans.- To get Git setup ,we can install it on linux via command “sudo apt-get install git”.

Since here Git is already installed ,it update to it’s newer version.



1. [Initialize a Git Repository.](https://confluence.atlassian.com/bitbucket/set-up-git-744723531.html)

Ans.- To initialize the Git repository “git init” command is used.here i first created a directory Exercise-2.now inside that directory Git repository initialized.





1. Add files to the repository.

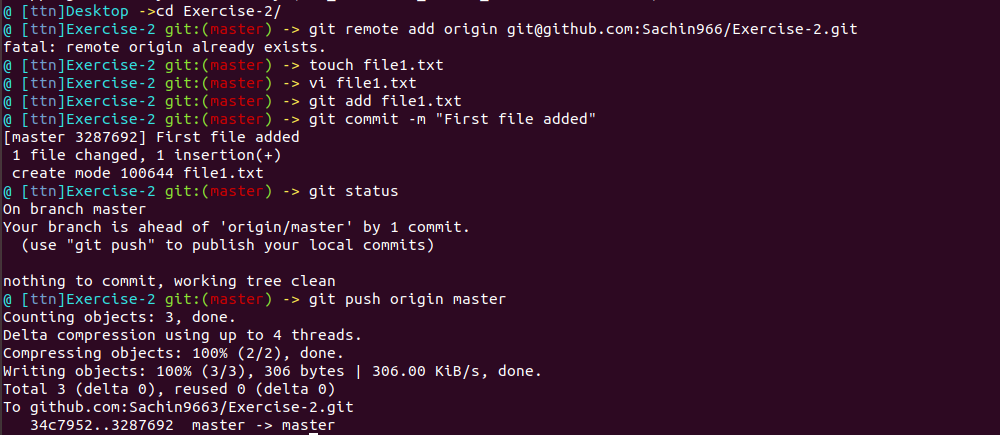
Ans:-To add file to repository,

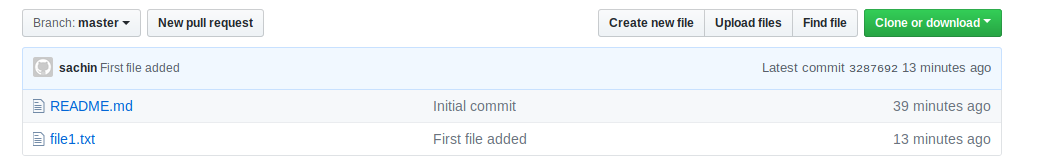
first create file using touch command.

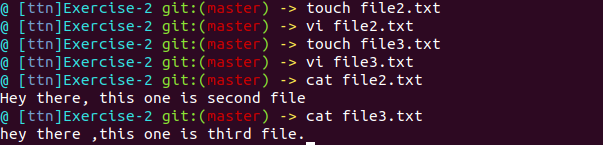
Edit file if need.

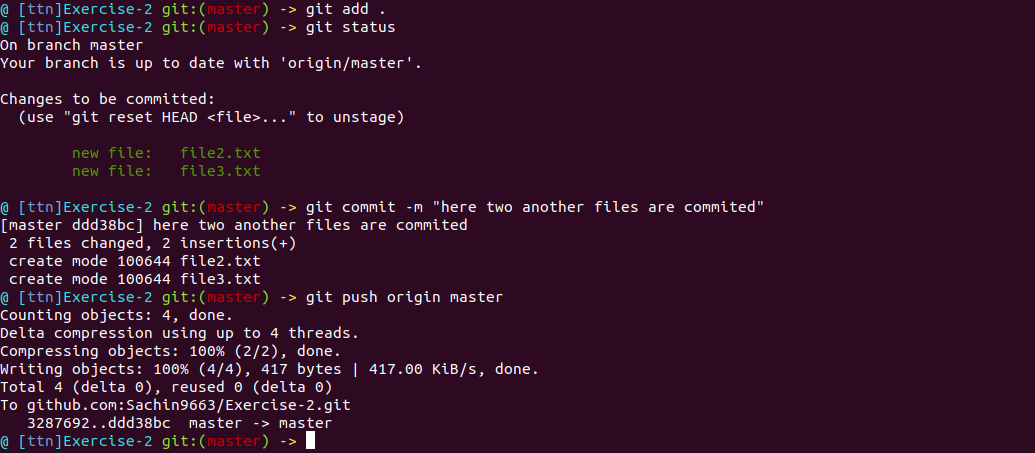
Use command git add file\_name or to add all file git add .

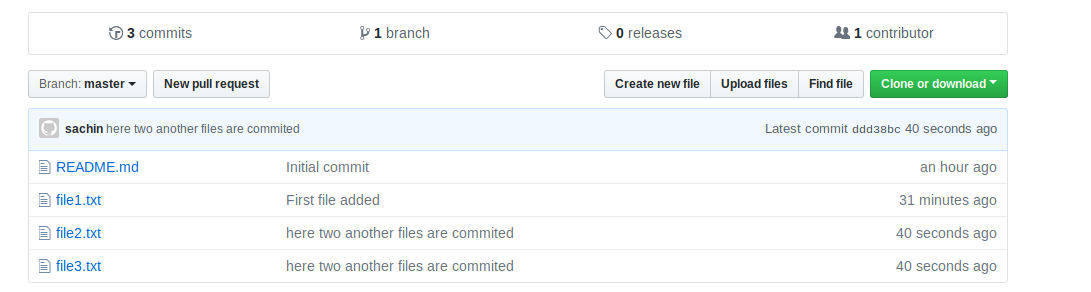
Use command git commit -m “Commit message”

Git push origin master(branch name);



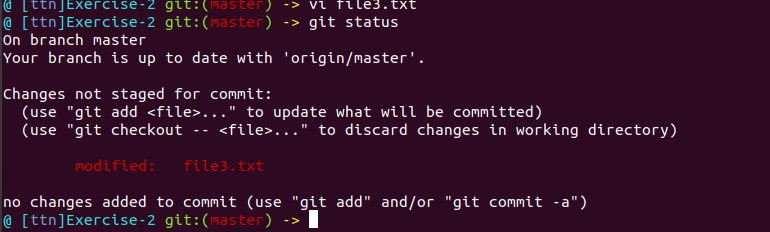




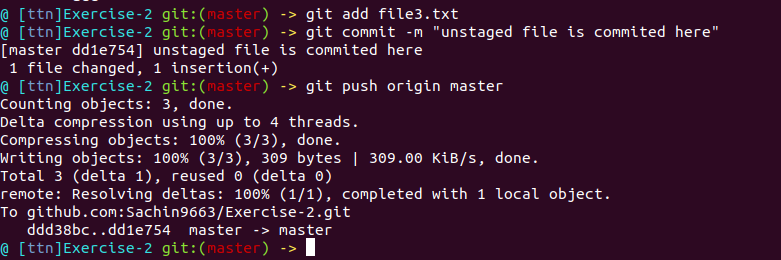


1. Unstage 1 file.

Ans.-



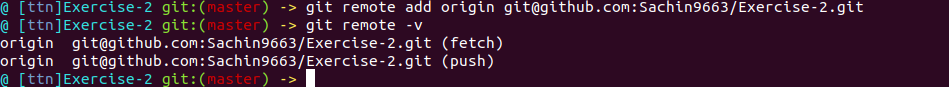
1. Commit the file.

Ans:-

1. Add a remote.

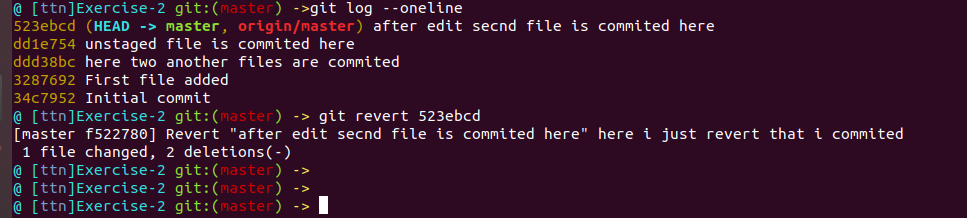
Ans:-In terminal to add the url for the remote repository,following command is used-

“Git remote add origin remote\_repository\_url”----To set the new remote

“Git remote -v”-----To verify new remote url 

1. Undo changes to a particular file.

Ans: git revert function is used to revert the changes in file.



1. Push changes to Github.

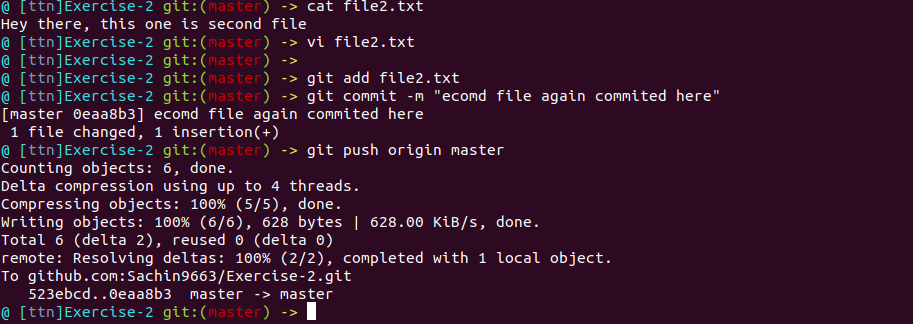
To push the changes to github following command is use:

Git add file\_name

Git commit -m ”Commit to the file”

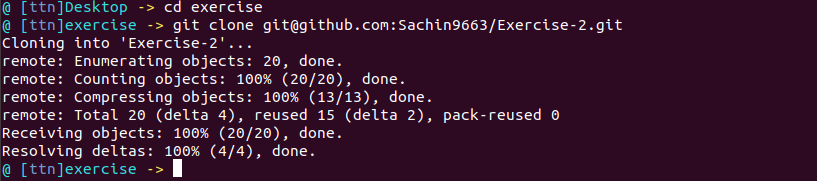
Git push origin master

After run these command ,puch will get successful.

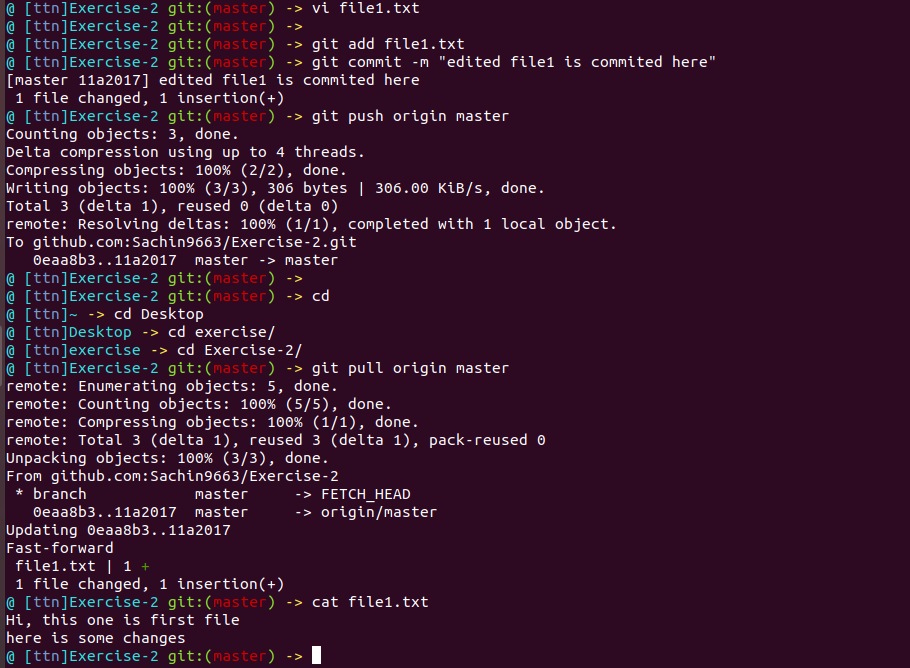


1. Clone the repository .

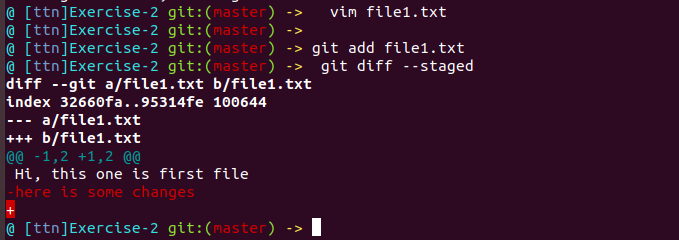
Ans :- to create the clone of repository first change the current working directory to the location where want the cloned directory to be made. And then to clone use command “git clone repository“



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



1. Check differences between a file and its staged version.



1. Ignore a few files to be checked in.

A file named “.gitignore” is used to specify the name of files that need not to be tracked by

Git. if we want to be make any file untracked

Touch file\_name

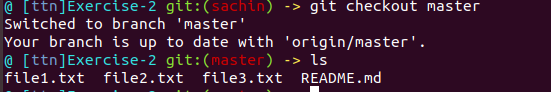
Vi .gitignore

1. Create a new branch.

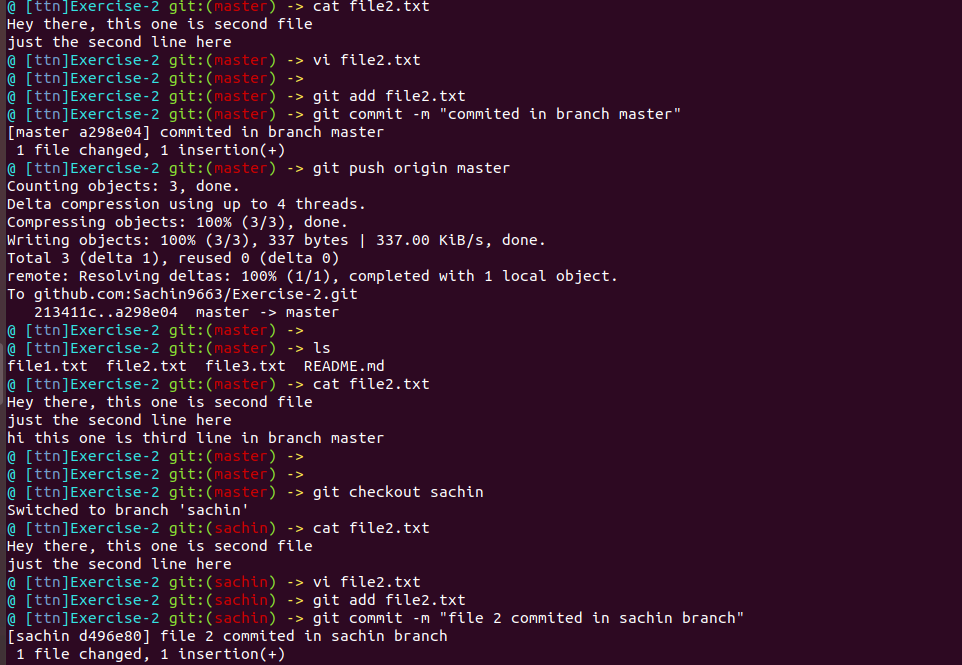
Ans:- 

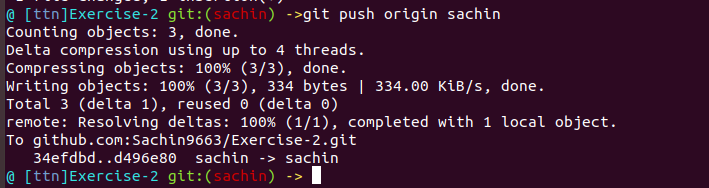
1. Diverge them with commits

Ans:-



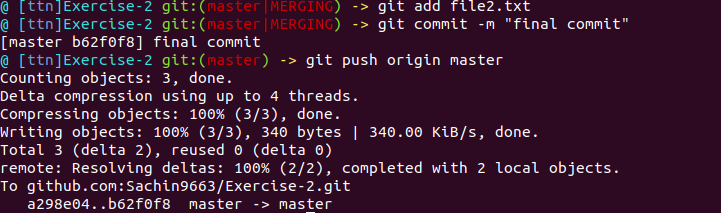
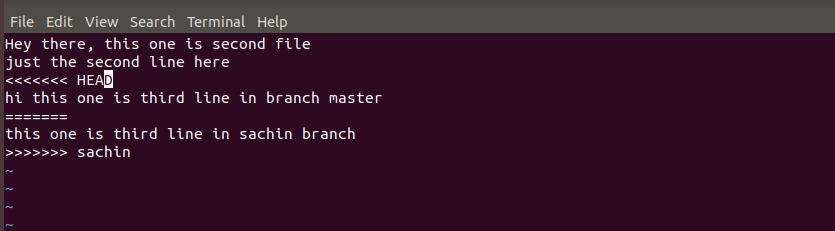
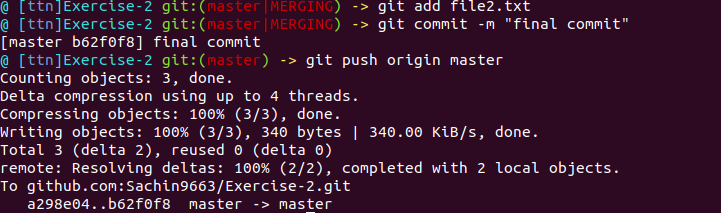
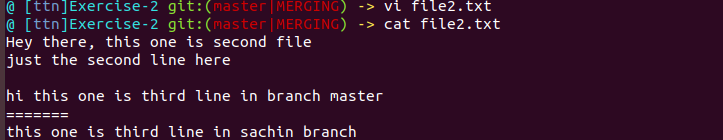
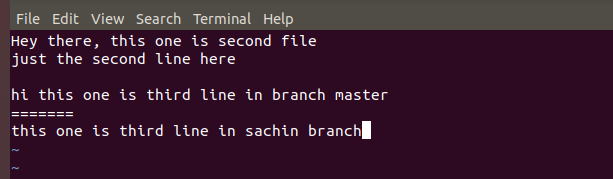
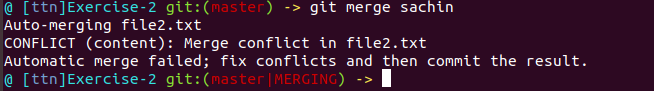
1. Edit the same file at the same line on both branches and commit.



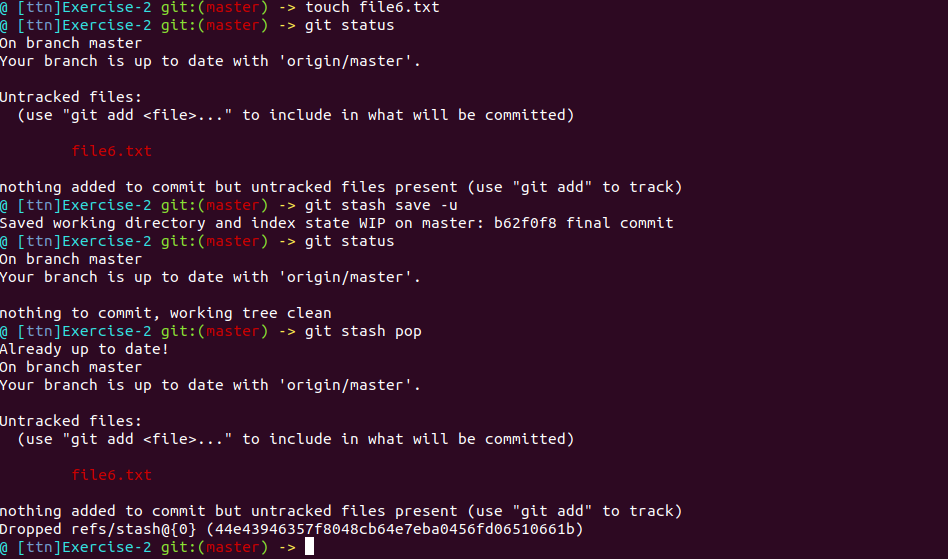


1. Try merging and resolve merge conflicts.

Ans in second picture heads are removed to resolve conflicts.



1. Stash the changes and pop them.



1. Add the following code to your .bashrc file :

Ans :-

#Git branch to be shown in bracket while in console

parse\_git\_branch() {

git branch 2> /dev/null | sed -e '/^[^\*]/d' -e 's/\* \(.\*\)/(\1)/'

}

if [ "$color\_prompt" = yes ]; then

PS1='${debian\_chroot:+($debian\_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;34m\]\w\[\033[01;31m\]$(parse\_git\_branch)\[\033[00m\]\$ '

else

PS1='${debian\_chroot:+($debian\_chroot)}\u@\h:\w$(parse\_git\_branch)\$ '

fi

unset color\_prompt force\_color\_prompt

RED="\e[0;31m"

GREEN="\e[0;92m"

BLACK="\e[m"

YELLOW="\e[0;93m"

export PS1='\[\e[0;96m\]@ [\[\e[0;94m\]\u\[\e[0;96m]\]\W\[\e[m\]\

$(echo $(\_\_git\_ps1 "\['$GREEN'\] git:(\['$RED'\]%s\['$GREEN'\])")) \['$YELLOW'\]->\['$BLACK'\] '



