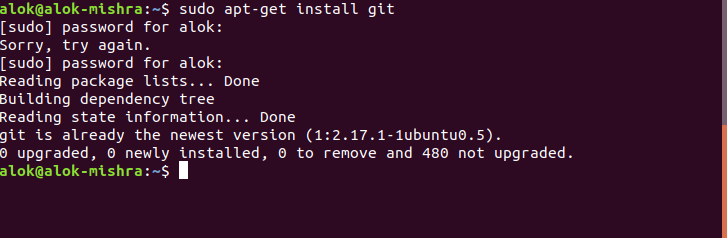
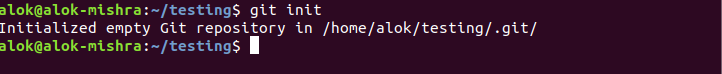
1. Git Setup <https://confluence.atlassian.com/bitbucket/set-up-git-744723531.html>

**Sudo apt-get install git**



1. Initialize a Git Repository

**git init (inside the directory)**

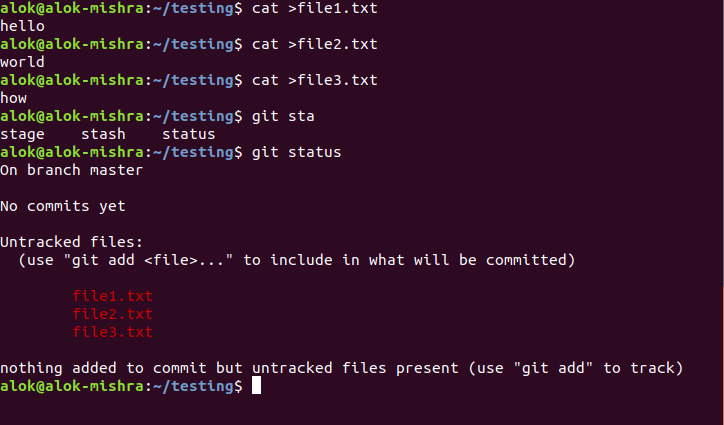


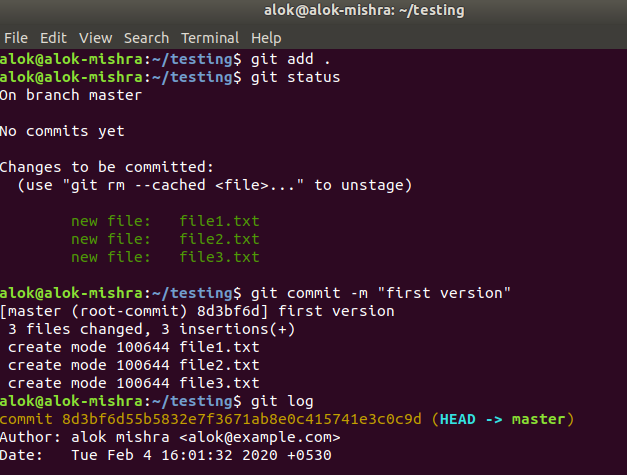
1. Add files to the repository

**cat > file1.txt**

**cat > file2.txt**

**cat > file3.txt**



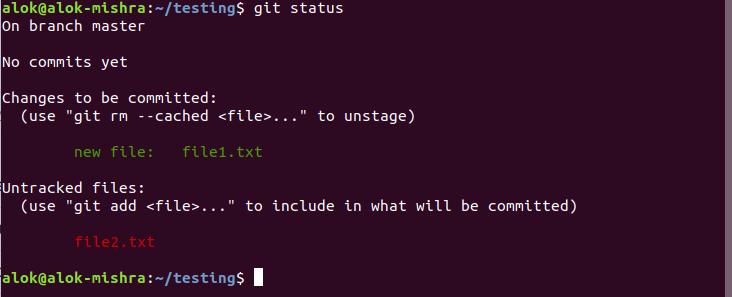


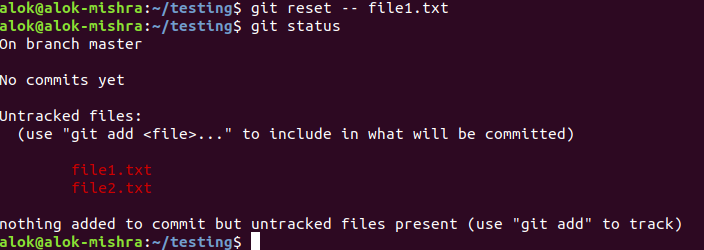
1. Unstage 1 file

Check status

**git status**

**git reset -- file1.txt**



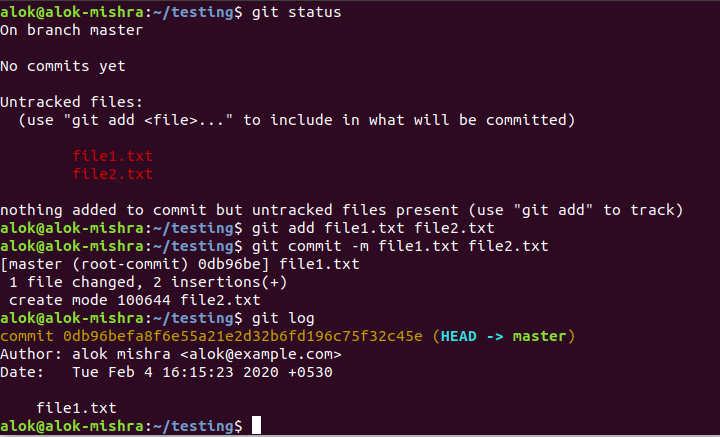


￼

1. Commit the file

**git add file1.txt file2.txt**

**git commit -m “first commit”**

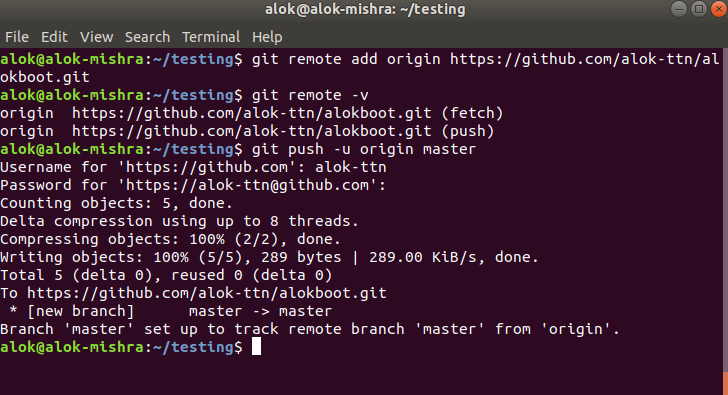


1. Add a remote

**git remote add origin <url> <path>**

**git remote -v**

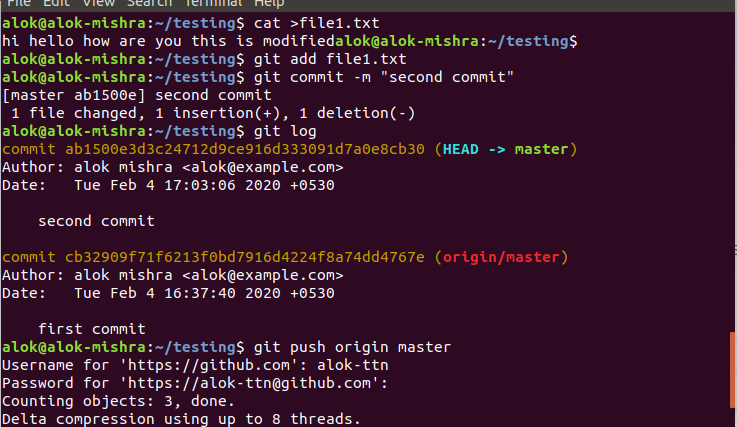
**git push -u origin master**

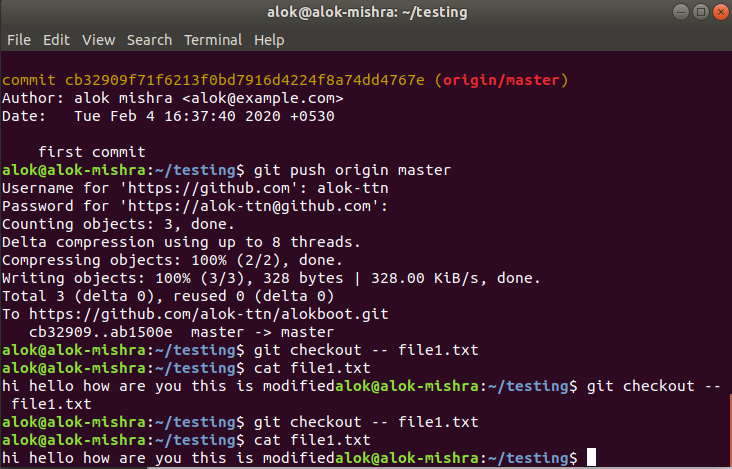


1. Undo changes to a particular file

After making changes use

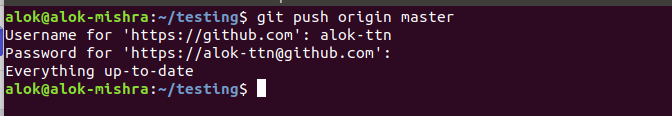
**git checkout -- file1.txt**





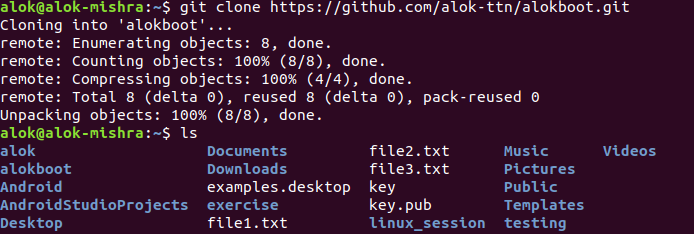
1. Push changes to Github

**git push origin master**

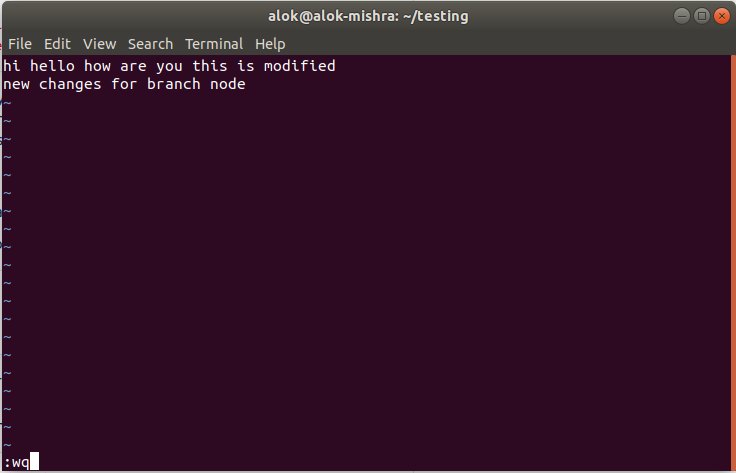


1. Clone the repository

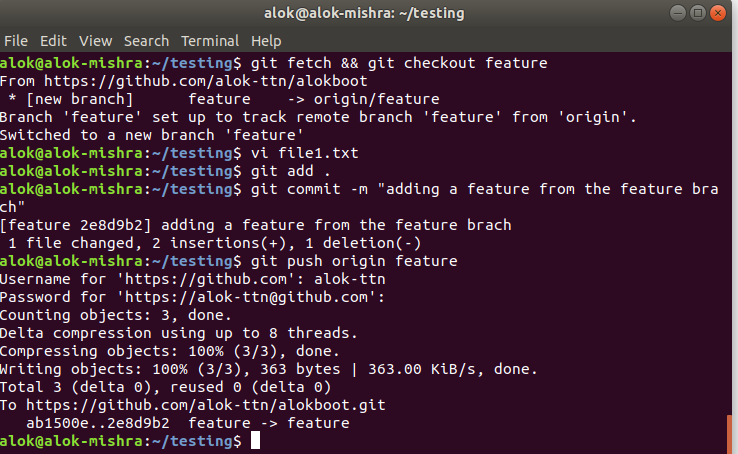
**git clone** [**https://github.com/alok-ttn/alokboot.git**](https://github.com/alok-ttn/alokboot.git)



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

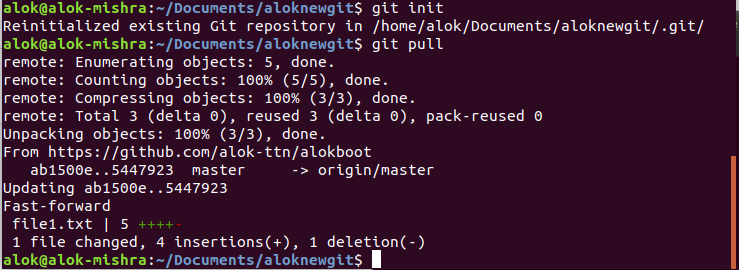


**git fetch && git checkout feature**

**git push origin feature**

**git init**

**git pull**

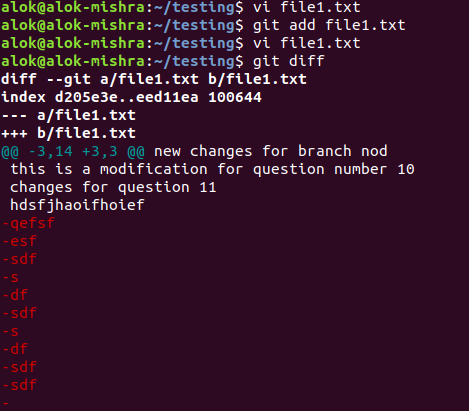


1. Check differences between a file and its staged version

**git add file1.txt**

**vi file1.txt**

**git diff**



1. Ignore a few files to be checked in

**#create a .gitignore file using text editor**

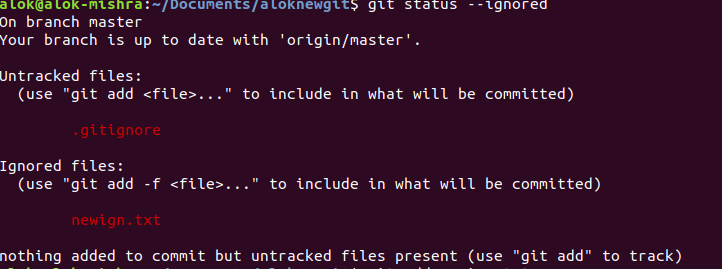
**vi .gitignore**

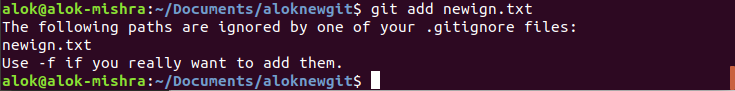
**#configure**

**git config --global core.excludesfile ~/.gitignore\_global**

**#check whether file is being ignored**

**git status --ignored**

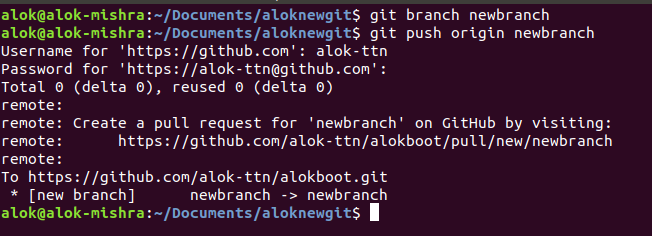




1. Create a new branch.

**git branch newbranch**

**git push origin newbranch**



1. Diverge them with commits

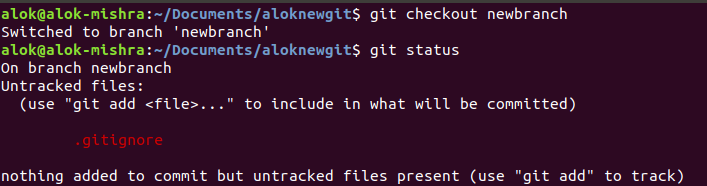
**#switch to second branch and commit edited files**

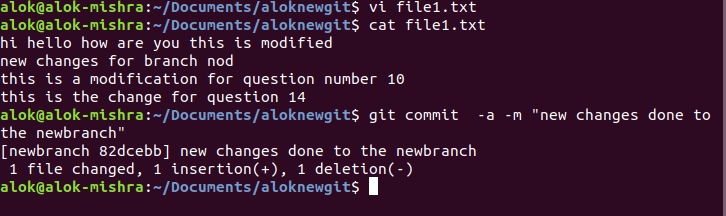
**git checkout newbranch**

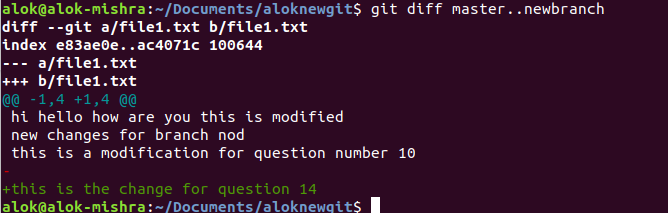
**vi file1.txt**

**git commit -a -m “new changes done to the newbranch”**

**git diff master..newbranch**







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

**#add new changes using master branch**

**git checkout master**

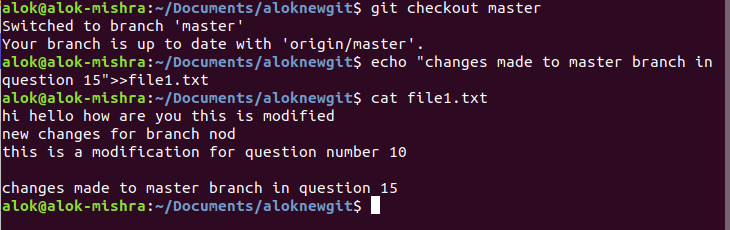
**echo “changes made to master branch in question 15”>>file1.txt**

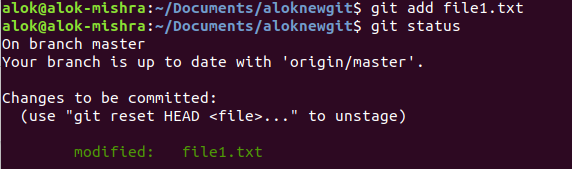
**git add file1.txt**

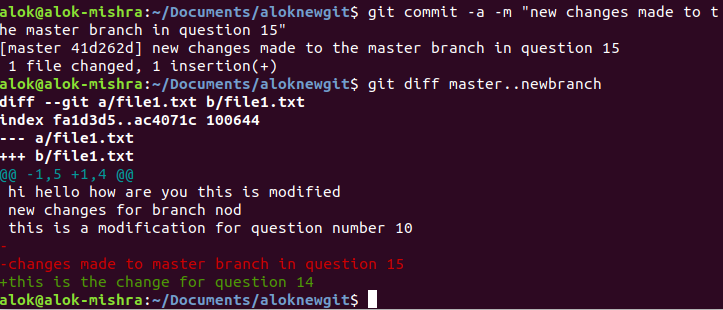
**git status**

**git commit -a -m “changes made to master branch in question 15”**

**git diff master..newbranch**



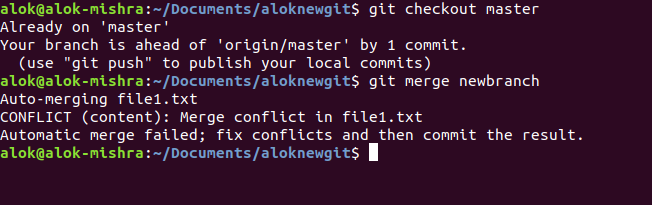


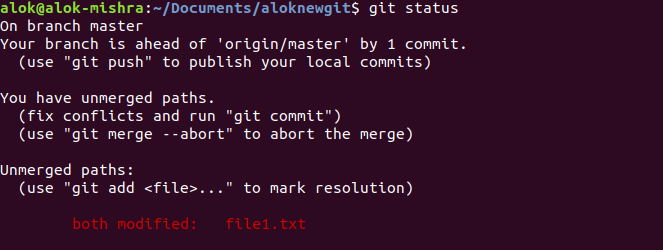


1. Try merging and resolve merge conflicts

**git checkout master**

**git merge newbranch**

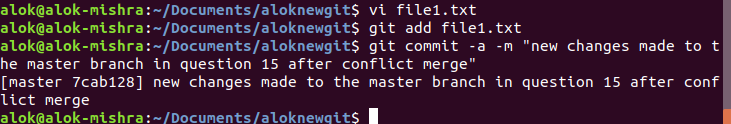




**git add file1.txt**

**git commit -a -m “new changes made to the master branch in question 15 after conflict merge”**

**git merge newbranch**





1. Stash the changes and pop them

Echo “new changes made in question 16 stash” >> file1.txt

git add file1.txt

#stashing the uncommitted changes

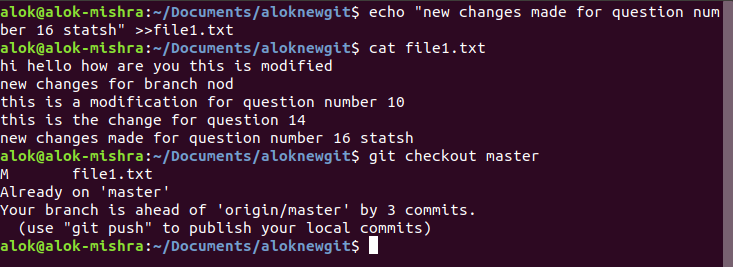
git stash

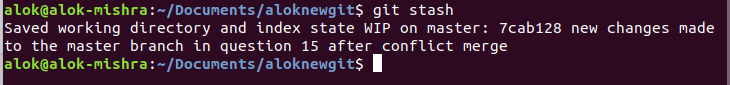
**#**check current working file

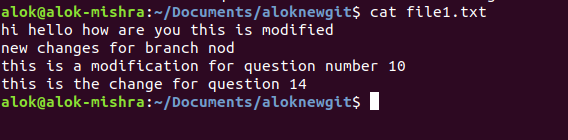
#re-applying old stashed changes

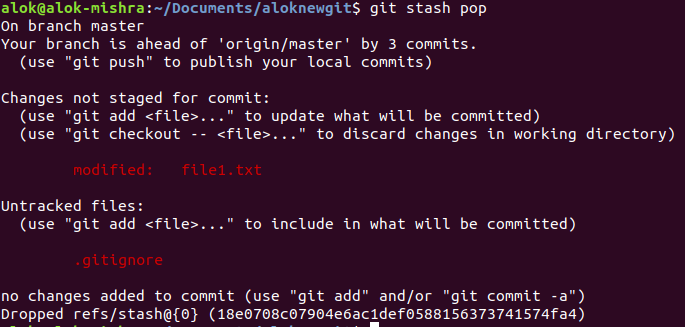
git stash pop

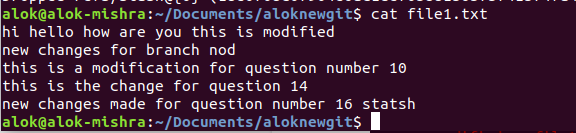
#stashed changes re-applied











1. 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

**vi ~/.bashrc**

