Exercise - Introduction to Version Control

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

Ans.

```
ttn@kunark:~$ git --version
git version 2.17.1
ttn@kunark:~$
```

Sudo apt-get install git is used to install git

2. Initialize a Git Repository

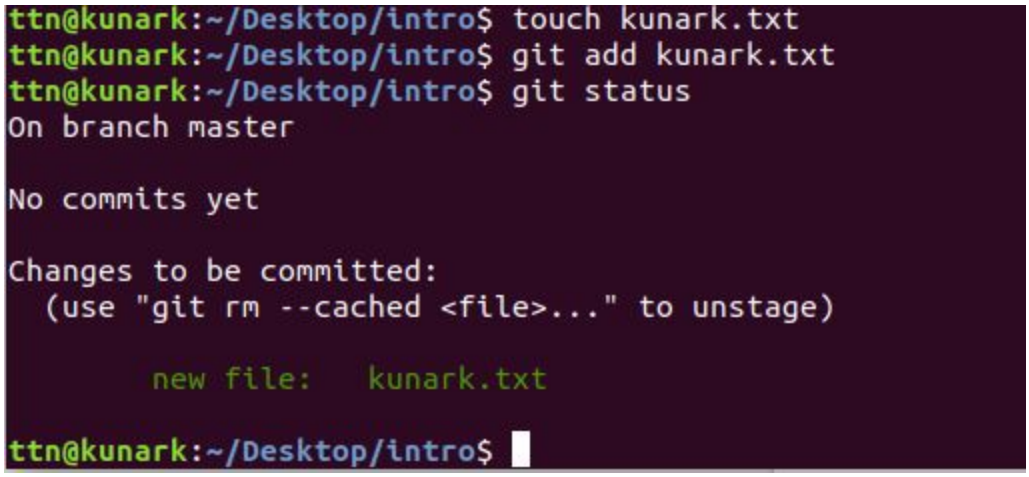
Ans.

```
ttn@kunark:~/Desktop$ mkdir intro
ttn@kunark:~/Desktop$ cd intro
ttn@kunark:~/Desktop/intro$ git init
Initialized empty Git repository in /home/ttn/Desktop/intro/.git/
ttn@kunark:~/Desktop/intro$
```

Git init for git initialize in local directory

3. Add files to the repository

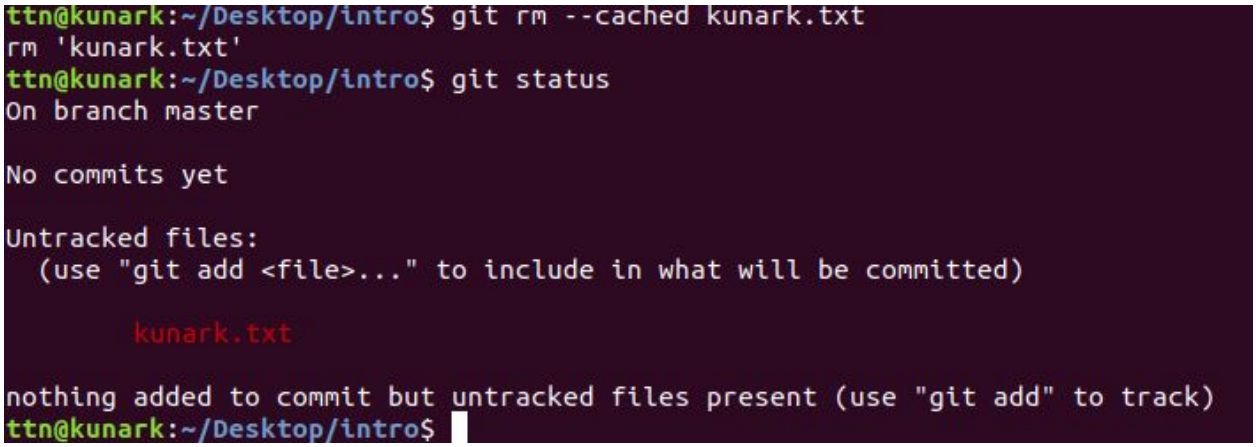
Ans.



Touch to create file
Git add <file name> to stage the the file or adding

4. Unstage 1 file

Ans.



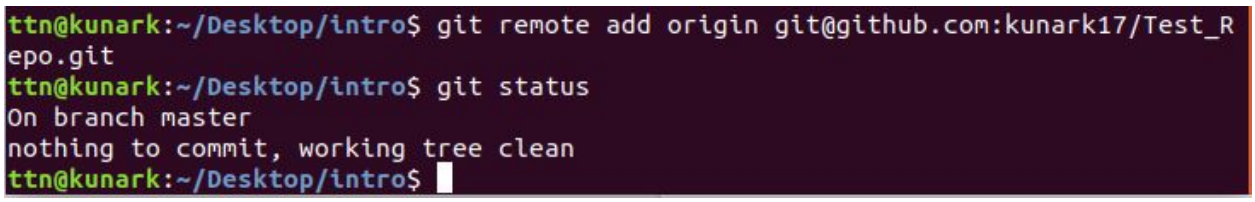
Git rm to remove or unstage the staged file --cached means the files saved in cached that are staged files

5. Commit the file

Ans.

6. Add a remote

Ans.



7. Undo changes to a particular file

Ans.

```
ttn@kunark:~/Desktop/intro$ git commit -m "changes in text file"
[master 20e80ee] changes in text file
 1 file changed, 1 insertion(+)
ttn@kunark:~/Desktop/intro$ git status
On branch master
nothing to commit, working tree clean
ttn@kunark:~/Desktop/intro$ git revert HEAD~0
[master 002eabe] Revert "changes in text file"
 1 file changed, 1 deletion(-)
ttn@kunark:~/Desktop/intro$ git status
On branch master
nothing to commit, working tree clean
ttn@kunark:~/Desktop/intro$
```

Firstly file is commited then undo is done by revert command and then file is deleted HEAD~0 specify the 1st commit

8. Push changes to Github

Ans.

```
ttn@kunark:~/Desktop/intro$ git push origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 214 bytes | 214.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:kunark17/test_repo.git
 * [new branch]      master -> master
ttn@kunark:~/Desktop/intro$
```

9. Clone the repository

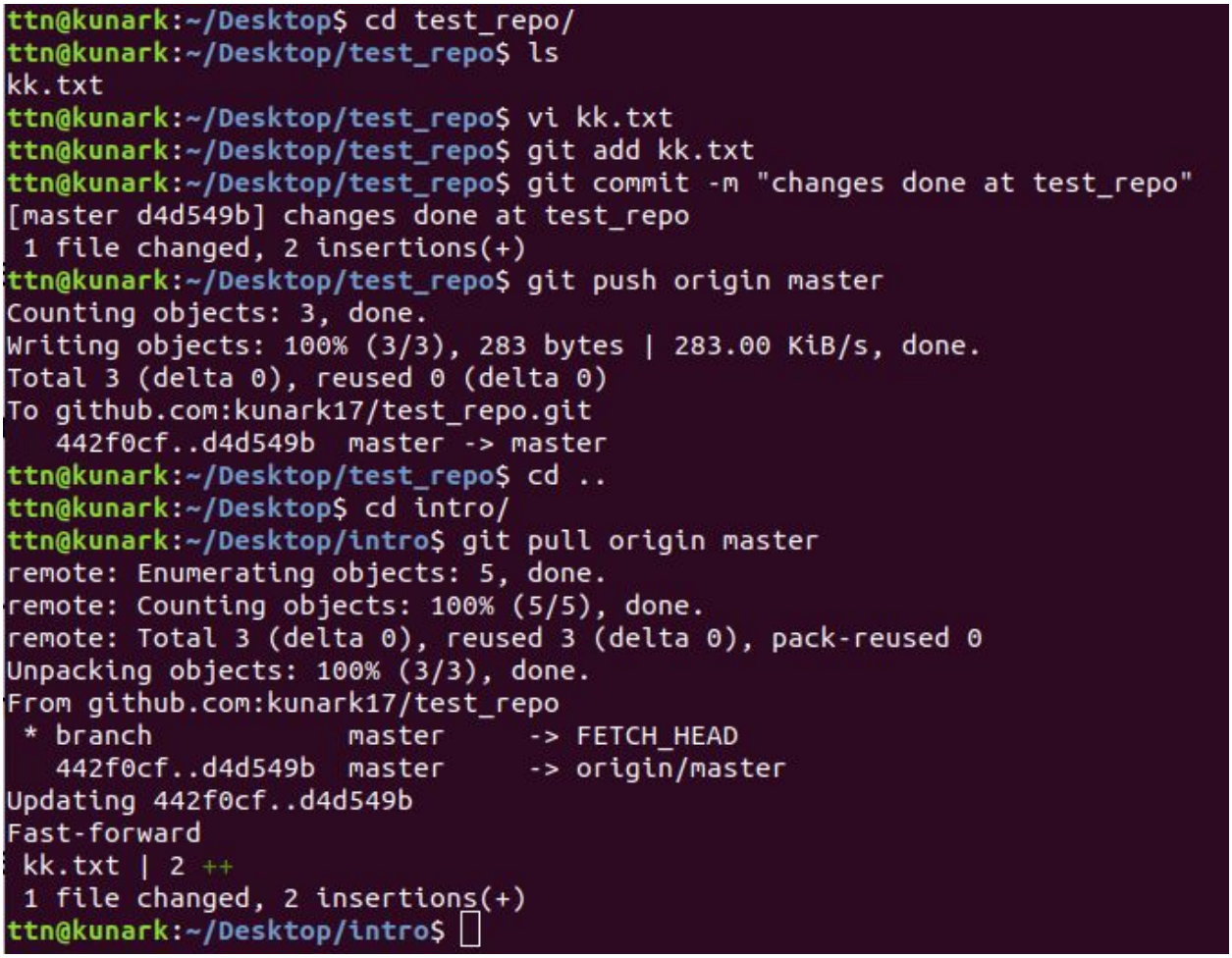
Ans.

```
ttn@kunark:~/Desktop/intro$ cd ..
ttn@kunark:~/Desktop$ git clone git@github.com:kunark17/test_repo.git
Cloning into 'test_repo'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
ttn@kunark:~/Desktop$ ls
3848765-wallpaper-images-download.jpg  intro  learning  test_repo
ttn@kunark:~/Desktop$
```

Git clone <url of repo>

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

Ans.



1.Changes made in test_repo, added, commited

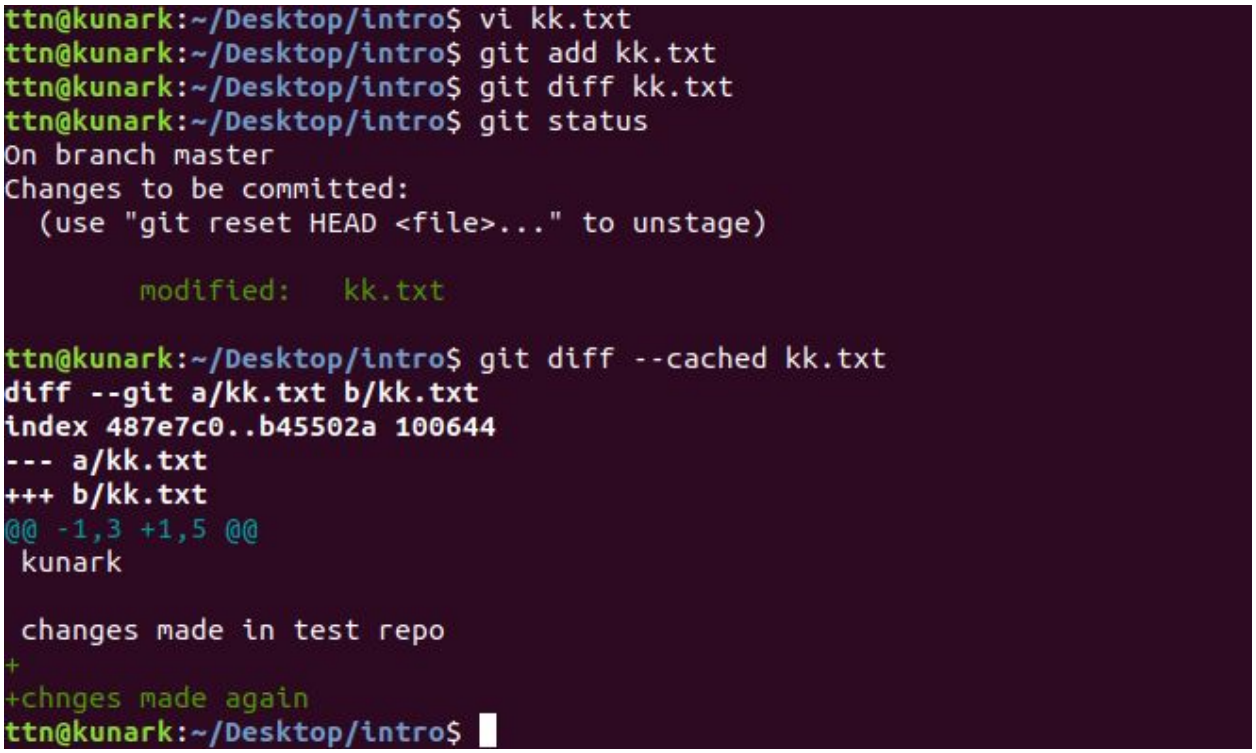
2.changes pushed

3.changed directory to intro

4.git pulled the files

11. Check differences between a file and its staged version

Ans.



1. Changes made in kk.txt file in intro
2. Git add kk.txt for staging the file
3. Git diff --cached <filename> => git diff to check the diffrence and --cached to check diffrence with the cached/staged vrsion of the filename

12. Ignore a few files to be checked in

Ans.

```
ttn@kunark:~/Desktop/intro$ vi .gitignore
ttn@kunark:~/Desktop/intro$ git add file1.txt file2.txt
The following paths are ignored by one of your .gitignore files:
file1.txt
file2.txt
Use -f if you really want to add them.
ttn@kunark:~/Desktop/intro$
```

vi .gitignore creates the gitignore file in vi
Then add the file name you want to ignore in format /file1.txt
When try to add the ignored file it shows the following above msg

13. Create a new branch.

Ans.

```
ttn@kunark:~/Desktop/intro$ git branch newBranch
ttn@kunark:~/Desktop/intro$ git branch
* master
  newBranch
ttn@kunark:~/Desktop/intro$
```

Git branch is used to just create a branch
Other way being
Git checkout -b <banchname>
This method will create the branch by -b and will move you to tht branch because of checkout

14. Diverge them with commits

Ans.

```
ttn@kunark:~/Desktop/intro$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   kk.txt

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

        .gitignore
        .gitignore.swp
        gitignore

ttn@kunark:~/Desktop/intro$ git commit -m "commit in master"
[master 6145d2b] commit in master
 1 file changed, 2 insertions(+)
ttn@kunark:~/Desktop/intro$ git checkout newBranch
Switched to branch 'newBranch'
ttn@kunark:~/Desktop/intro$ vi kk.txt
ttn@kunark:~/Desktop/intro$ git add kk.txt
ttn@kunark:~/Desktop/intro$ git commit -m "commit in new Branch"
[newBranch 1f866c6] commit in new Branch
 1 file changed, 2 insertions(+)
ttn@kunark:~/Desktop/intro$ git status
On branch newBranch
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .gitignore
        .gitignore.swp
        gitignore

nothing added to commit but untracked files present (use "git add" to track)
ttn@kunark:~/Desktop/intro$
```

New branch is made and changes are commit
The files were also changed in master branch causing them to diverge

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

Ans.

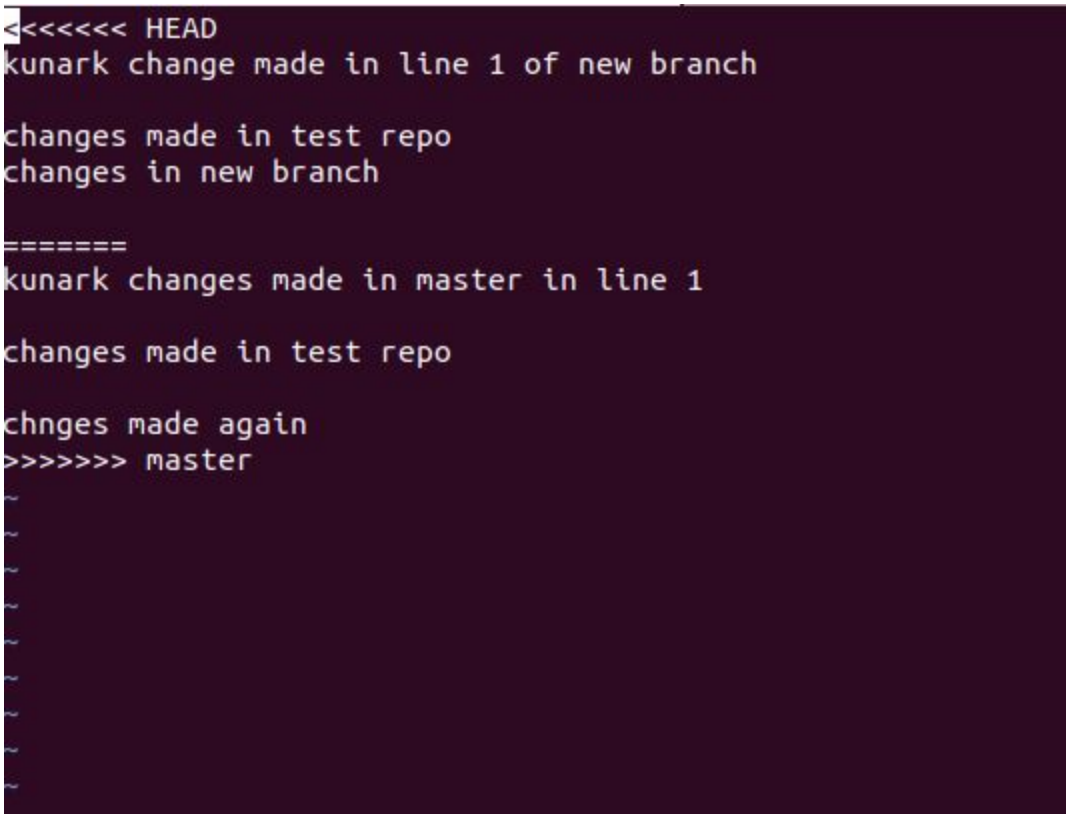
```
ttn@kunark:~/Desktop/intro$ git checkout
ttn@kunark:~/Desktop/intro$ git branch
  master
* newBranch
ttn@kunark:~/Desktop/intro$ vi kk.txt
ttn@kunark:~/Desktop/intro$ git add kk.txt
ttn@kunark:~/Desktop/intro$ git commmit -m "changes made on line 1 in new branch"
git: 'commmit' is not a git command. See 'git --help'.

The most similar command is
        commit
ttn@kunark:~/Desktop/intro$ git commit -m "changes made on line 1 in new branch"
[newBranch d8c4643] changes made on line 1 in new branch
 1 file changed, 1 insertion(+), 1 deletion(-)
ttn@kunark:~/Desktop/intro$ git che
checkout      cherry        cherry-pick
ttn@kunark:~/Desktop/intro$ git checkout master
Switched to branch 'master'
ttn@kunark:~/Desktop/intro$ vi kk.txt
ttn@kunark:~/Desktop/intro$ git add kk.txt
ttn@kunark:~/Desktop/intro$ git commit -m "changes made in line 1 in master branch"
[master 6e809e8] changes made in line 1 in master branch
 1 file changed, 1 insertion(+), 1 deletion(-)
ttn@kunark:~/Desktop/intro$
```

16. Try merging and resolve merge conflicts

Ans.

```
ttn@kunark:~/Desktop/intro$ git branch
* master
  newBranch
ttn@kunark:~/Desktop/intro$ git checkout newBranch
Switched to branch 'newBranch'
ttn@kunark:~/Desktop/intro$ git merge master
Auto-merging kk.txt
CONFLICT (content): Merge conflict in kk.txt
Automatic merge failed; fix conflicts and then commit the result.
ttn@kunark:~/Desktop/intro$ vi kk.txt
ttn@kunark:~/Desktop/intro$
```

Trying to merge but conflict occur in kk.txt then the file is opened in vi



this was the file with changes shown

```
kunark change made in line 1 of new branch

changes made in test repo
changes in new branch

kunark changes made in master in line 1

changes made in test repo

chnges made again
~
~
~
~
~
~
```

After removing the unwanted lines file is saved by :wq!

```
ttn@kunark:~/Desktop/intro$ git add .
ttn@kunark:~/Desktop/intro$ git commit -m "resolved merge conflict"
[newBranch 780f398] resolved merge conflict
ttn@kunark:~/Desktop/intro$ git status
On branch newBranch
nothing to commit, working tree clean
ttn@kunark:~/Desktop/intro$
```

Then file is added and commit is made as “resolved merge conflict”

```
ttn@kunark:~/Desktop/intro$ git merge master
Already up to date.
ttn@kunark:~/Desktop/intro$
```

Then doing git merge won`t show any conflict

17. Stash the changes and pop them

Ans.

1. Here first branch is checked
2. Then a change is made in kk.txt
3. When trying to checkout it shows warning to either commit or stash
4. git stash -p is used to stash the uncommit changes

For Pop

```
ttn@kunark:~/Desktop/intro$ git checkout master
Switched to branch 'master'
ttn@kunark:~/Desktop/intro$ git stash pop
Auto-merging kk.txt
CONFLICT (content): Merge conflict in kk.txt
ttn@kunark:~/Desktop/intro$ vi kk.txt
```

1. After checkout to master I've done git stash pop to add changes
2. Then merge conflict error is shown so file is edited in vi

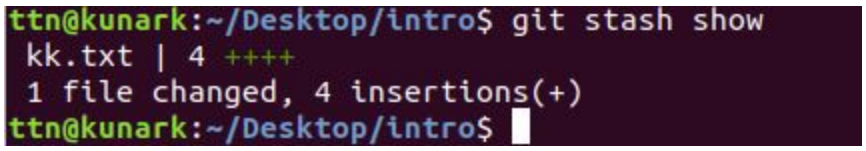
```
Dchnge in new branch



kunark change made in line 1 of new branch

changes made in test repo
changes in new branch

kunark changes made in master in line 1

changes made in test repo

chnges made again
~
~
```

Here appropriate changes are made

If we try pop again

```
ttn@kunark:~/Desktop/intro$ git stash pop
kk.txt: needs merge
unable to refresh index
ttn@kunark:~/Desktop/intro$
```

Stash show



Stash show is uesd to show the changes stashed

18. 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
```

Ans.

Before

```
ttn@kunark:~/Desktop/intro$ git status
On branch master
nothing to commit, working tree clean
ttn@kunark:~/Desktop/intro$
```

In bashrc code is pasted and saved

```
# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

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)\[\03
3[00m\]\$ '
else
PS1='\u@\h:\W $(parse_git_branch)\$ '
fi
unset color_prompt force_color_prompt
```

Source is used to install the bashrc

```
ttn@kunark:~/Desktop/intro$ source ~/.bashrc
ttn@kunark:intro (master)$ git status
On branch master
nothing to commit, working tree clean
ttn@kunark:intro (master)$ git checkout newBranch
Switched to branch 'newBranch'
ttn@kunark:intro (newBranch)$
```