

## Q1. Git Setup <https://confluence.atlassian.com/bitbucket/set-up-git-744723531.html>

```
pushkar@Pushkar-Singh: ~/exercise
pushkar@Pushkar-Singh: ~/Desktop/checking-git$ sudo apt-get install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
git is already the newest version (1:2.7.4-0ubuntu1.3).
The following package was automatically installed and is no longer required:
  ubuntu-core-launcher
use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 19 not upgraded.
pushkar@Pushkar-Singh: ~/Desktop/checking-git$ git config --global user.name "pushkarsinghTTN"
pushkar@Pushkar-Singh: ~/Desktop/checking-git$ git config --global user.email "pushkarsingh@tothenew.com"
pushkar@Pushkar-Singh: ~/exercise$ cd /home/pushkar/exercise
git version 2.7.4
pushkar@Pushkar-Singh: ~/exercise$ █
```

git version 2.7.4

1. Create a file named "output" in the directory "/tmp". Copy the contents of the "output" file created above. Since the permission won't allow you to save the changes. Configure such that test user can edit it.

1. Add group owner of the "output" file as the secondary group of testuser and check/change the "output" file permission if it is editable by group. Once done revert the changes.
2. Make the file editable to the world so that test user can access it. Revert the changes after verification.
3. Change the ownership to edit the file.

11. Create alias with your name so that it creates a file as "/tmp/aliasTesting".

12. Edit ~/.bashrc file such that when you change to "test" user it should clear the screen and print "Welcome".

13. Install "zip" package.

14. Compress "output" and "password\_backup" files into a tar ball. List the files present inside the tar created.

15. scp this file to test user

16. Unzip this tar ball by logging into the remote server

17. Download any image from web and move to desktop

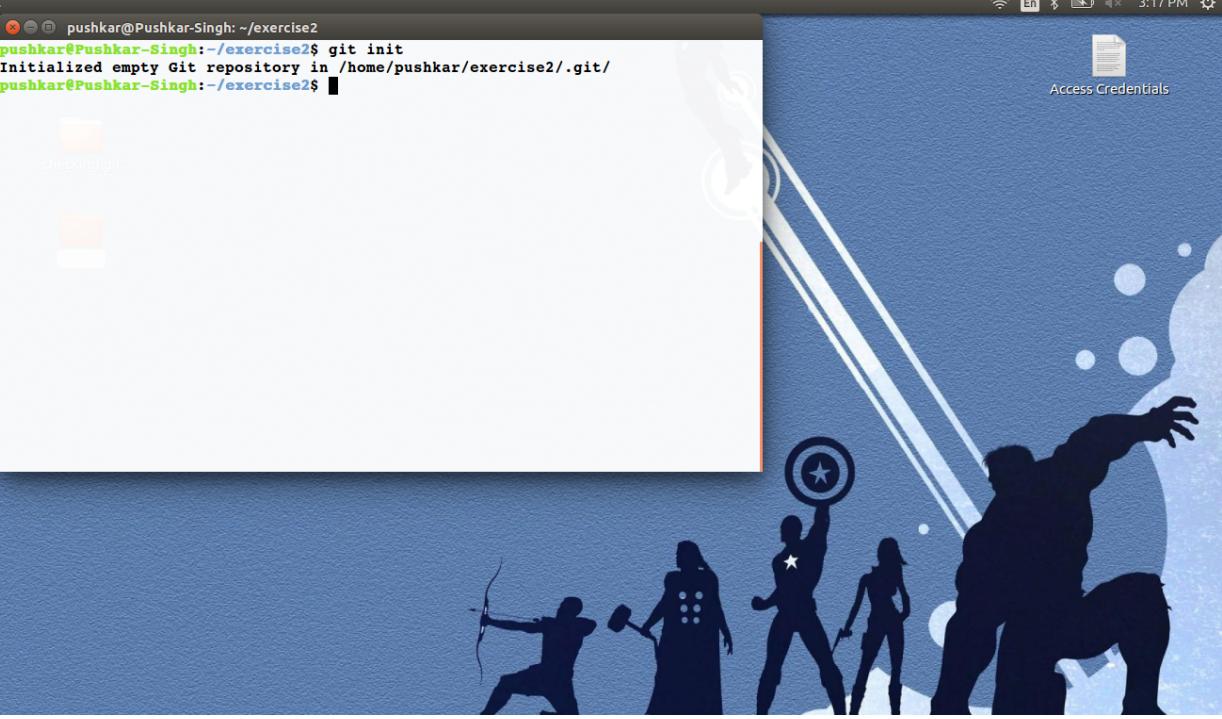
18. How to get help of commands usages.

19. Create a symlink of /etc/services into /tmp/ports-info

20. You are appointed as a Software/DevOps Engineer in ABC media services. On your first day you need to troubleshoot a problem. There is a command "xyz" somewhere installed in that linux system. But as a new joinee you do not have any idea about where is that installed. How can you check that?

## Q2. Initialize a Git Repository

```
Terminal
pushkar@Pushkar-Singh: ~/exercise2
pushkar@Pushkar-Singh: ~/exercise2$ git init
Initialized empty Git repository in /home/pushkar/exercise2/.git/
pushkar@Pushkar-Singh: ~/exercise2$ █
```

A blue-themed background image featuring silhouettes of the Avengers characters: Hawkeye, Thor, Captain America, Iron Man, and Black Widow, standing against a bright, glowing background.

### Q3. Add files to the repository.

The screenshot shows a Mac OS X desktop environment. On the left is a Dock with various icons. In the center is a Terminal window titled "pushkar@Pushkar-Singh: ~/exercise2". The terminal output is:

```
pushkar@Pushkar-Singh:~/exercise2$ cat >sample
This is the sample file.
^Z
[1]+  Stopped                  cat > sample
pushkar@Pushkar-Singh:~/exercise2$ cat sample
This is the sample file.
pushkar@Pushkar-Singh:~/exercise2$
```

To the right of the terminal is a Bitbucket browser window titled "Create a New Repository". The URL in the address bar is "https://www.bitbucket.org/account/repositories/new/presentation/tdy194TPSMWx0kpxz02vpxF9EJBOJCZfOb1x7mDbsM/edit?usp=sharing". The page content includes a numbered list of Git steps:

1. Git Setup <https://confluence.atlassian.com/bitbucket/set-up-git-744723531.html>
2. Initialize a Git Repository
3. Add files to the repository
4. Unstage 1 file
5. Commit the file
6. Add a remote
7. Undo changes to a particular file
8. Push changes to Github
9. Clone the repository
10. Add changes to one of the copies and pull the changes in the other
11. Check differences between a file and its staged version
12. Create a new branch
13. Create a new branch.
14. Diverge them with commits
15. Edit the same file at the same line on both branches and commit
16. Try merging and resolve merge conflicts
17. Slash the changes and pop them
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)/* /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[033[00m]:[033[01;34m]W[033[01;31m] $(parse_git_branch)[033[00m]$ '
```

### Q4. Unstage 1 file

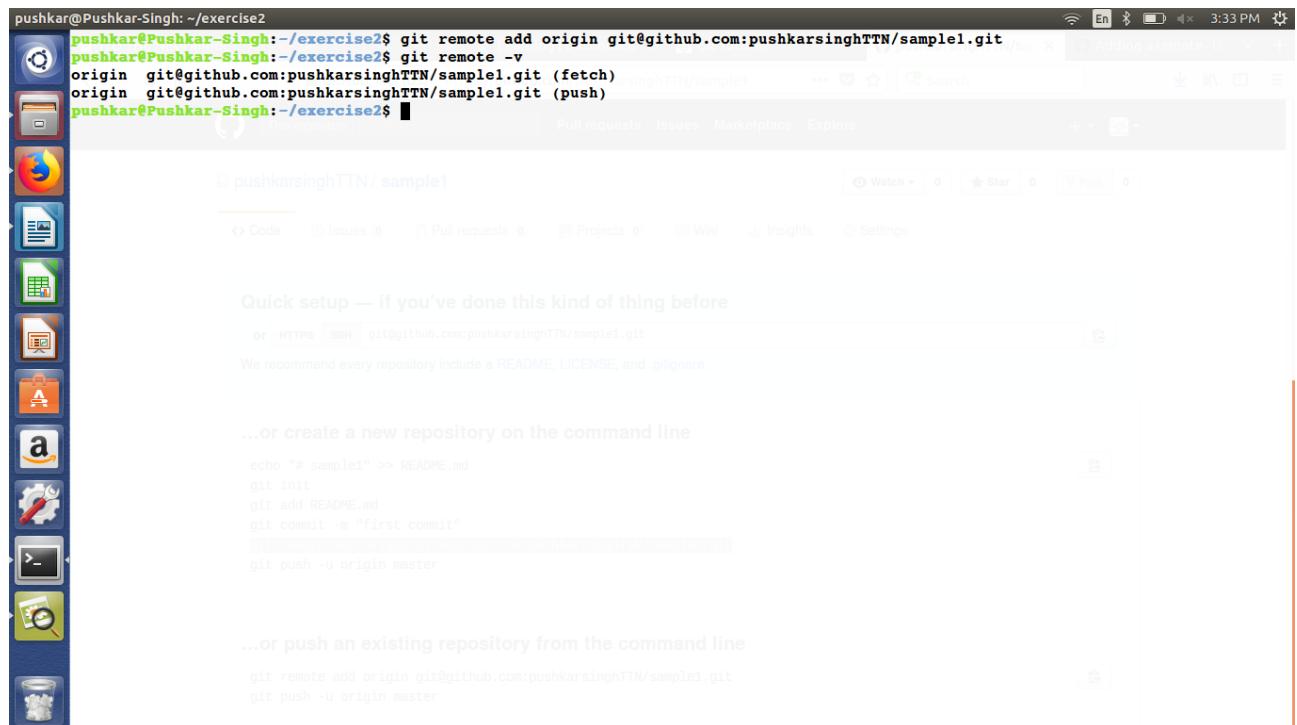
The screenshot shows a Mac OS X desktop environment. On the left is a Dock with various icons. In the center is a Terminal window titled "pushkar@Pushkar-Singh: ~/exercise2". The terminal output is:

```
pushkar@Pushkar-Singh:~/exercise2$  
new file: sample3  
Changes not staged for commit:  
(use "git add/rm <file>..." to update what will be committed)  
(use "git checkout -- <file>..." to discard changes in working directory)  
 deleted: sample  
pushkar@Pushkar-Singh:~/exercise2$ git reset sample1  
Unstaged changes after reset:  
D sample  
M sample2  
M sample3  
pushkar@Pushkar-Singh:~/exercise2$ git status  
On branch master  
Initial commit  
Changes to be committed:  
(use "git rm --cached <file>..." to unstage)  
 new file: sample  
 new file: sample2  
 new file: sample3  
Changes not staged for commit:  
(use "git add/rm <file>..." to update what will be committed)  
(use "git checkout -- <file>..." to discard changes in working directory)  
 deleted: sample  
 modified: sample2  
 modified: sample3  
untracked files:  
(use "git add <file>..." to include in what will be committed)  
 sample1  
pushkar@Pushkar-Singh:~/exercise2$
```

Q5. Commit the file.

```
Sample 16. Try merging and resolve merge conflicts
pushkar@Pushkar-Singh:~/exercise2$ git commit
Aborting commit due to empty commit message.
pushkar@Pushkar-Singh:~/exercise2$ git commit -m "This is commit"
[master (root-commit) 8830b0a] This is commit
 3 files changed, 3 insertions(+)
 create mode 100644 sample
 create mode 100644 sample2
 create mode 100644 sample3
pushkar@Pushkar-Singh:~/exercise2$
```

Q6. Add a remote.

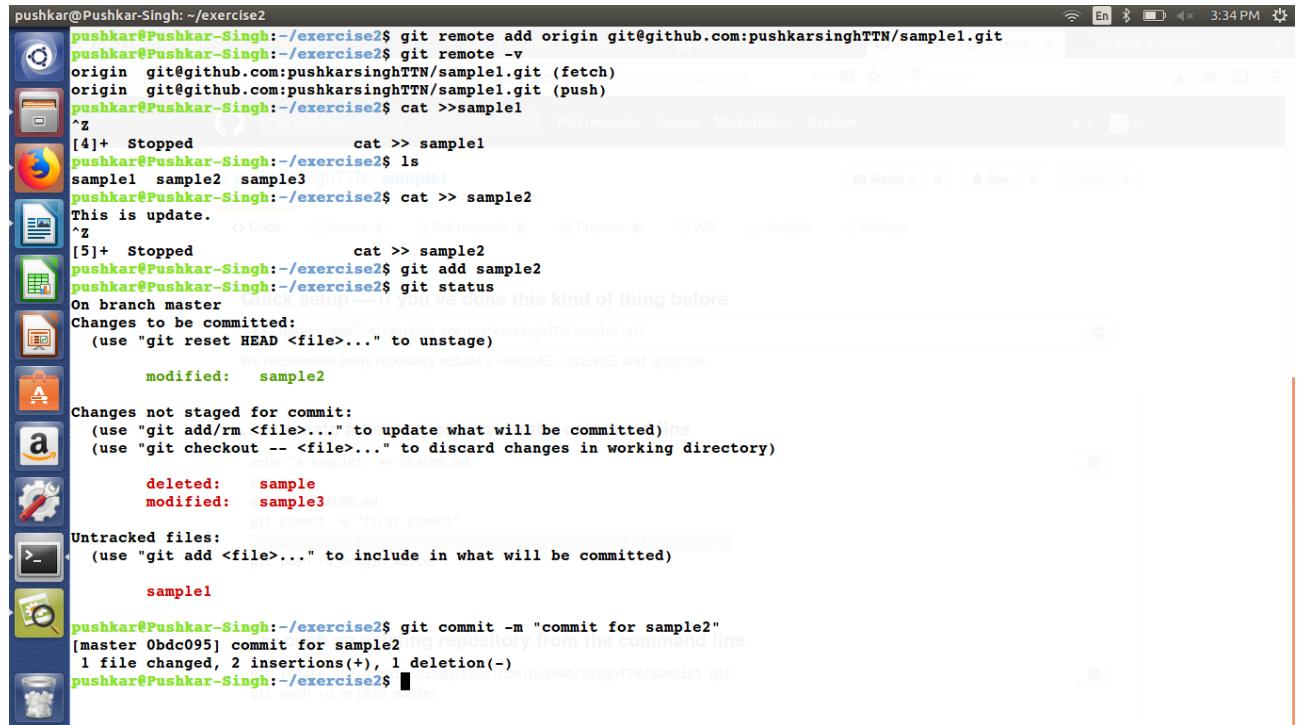


The screenshot shows a terminal window on a Mac OS X desktop. The title bar says "Terminal" and the status bar shows the date and time as 3:33 PM. The terminal content is as follows:

```
pushkar@Pushkar-Singh: ~/exercise2
pushkar@Pushkar-Singh:~/exercise2$ git remote add origin git@github.com:pushkarsinghTTN/sample1.git
pushkar@Pushkar-Singh:~/exercise2$ git remote -v
origin  git@github.com:pushkarsinghTTN/sample1.git (fetch)
origin  git@github.com:pushkarsinghTTN/sample1.git (push)
pushkar@Pushkar-Singh:~/exercise2$
```

Below the terminal, a GitHub repository page for "pushkarsinghTTN / sample1" is visible. The repository has 0 stars and 0 forks. It includes sections for "Code", "Issues", "Pull requests", "Projects", "Wiki", "Insights", and "Settings". A sidebar on the left contains various icons for GitHub features like issues, pull requests, and projects. The main content area shows instructions for quick setup via command line or browser, and examples for creating a new repository or pushing an existing one.

## Q7. Undo changes to a particular file.



```
pushkar@Pushkar-Singh:~/exercise2$ git remote add origin git@github.com:pushkarsinghTTN/sample1.git
pushkar@Pushkar-Singh:~/exercise2$ git remote -v
origin git@github.com:pushkarsinghTTN/sample1.git (fetch)
origin git@github.com:pushkarsinghTTN/sample1.git (push)
pushkar@Pushkar-Singh:~/exercise2$ cat >>sample1
^Z
[4]+  Stopped                  cat >> sample1
pushkar@Pushkar-Singh:~/exercise2$ ls
sample1 sample2 sample3
pushkar@Pushkar-Singh:~/exercise2$ cat >> sample2
This is update.
^Z
[5]+  Stopped                  cat >> sample2
pushkar@Pushkar-Singh:~/exercise2$ git add sample2
pushkar@Pushkar-Singh:~/exercise2$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)
    modified:   sample2
We recommend every repository include a README, LICENSE, and .gitignore.

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)
    deleted:    sample
    modified:   sample3
    modified:   README.md
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    sample1

pushkar@Pushkar-Singh:~/exercise2$ git commit -m "commit for sample2"
[master 0bdc095] commit for sample2
 1 file changed, 2 insertions(+), 1 deletion(-)
pushkar@Pushkar-Singh:~/exercise2$
```

## Q8. Push changes to Github.

```
pushkar@Pushkar-Singh:~/exercise2$ git push -u origin master
Counting objects: 8, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (8/8), 649 bytes | 0 bytes/s, done.
Total 8 (delta 0), reused 0 (delta 0)
To git@github.com:pushkarsinghTTN/sample1.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

## Q9. Clone the repository.

```
pushkar@Pushkar-Singh: ~/Desktop/AssignmentGit
pushkar@Pushkar-Singh:~/exercise2$ cd ..
pushkar@Pushkar-Singh:~$ cd Desktop
pushkar@Pushkar-Singh:~/Desktop$ git clone git@github.com:pushkarsinghTTN/AssignmentGit.git
Cloning into 'AssignmentGit'...
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 4 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
Checking connectivity... done.
pushkar@Pushkar-Singh:~/Desktop$ git remote -v
fatal: Not a git repository (or any parent up to mount point /home)
Stopping at filesystem boundary (GIT_DISCOVERY_ACROSS_FILESYSTEM not set).
pushkar@Pushkar-Singh:~/Desktop$ ls
Access Credentials AssignmentGit checking-git week1
pushkar@Pushkar-Singh:~/Desktop$ cd AssignmentGit/
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ git remote -v
origin git@github.com:pushkarsinghTTN/AssignmentGit.git (fetch)
origin git@github.com:pushkarsinghTTN/AssignmentGit.git (push)
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ ■
    7. Undo changes to a particular file
    8. Push changes to Github
    9. Clone the repository
    10. Add changes to one of the copies and pull the changes in the other
    11. Check differences between a file and its staged version
    12. Ignore a few files to be checked in
    13. Create a new branch
    14. Diverge them with commits
    15. Edit the same file at the same line on both branches and commit
    16. Try merging and resolve merge conflicts
    17. Stash the changes and pop them
    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/* //'
        }
        if [ "$color_prompt" = yes ]; then
        PS1="\u@\h \033[0m]:\033[01;34m]\W\033[01;31m]\$(parse_git_branch)\033[00m\$ "
        else
PS1="\u@\h \033[0m]:\033[01;34m]\W\033[01;31m]\$(parse_git_branch)\033[00m\$ "
```

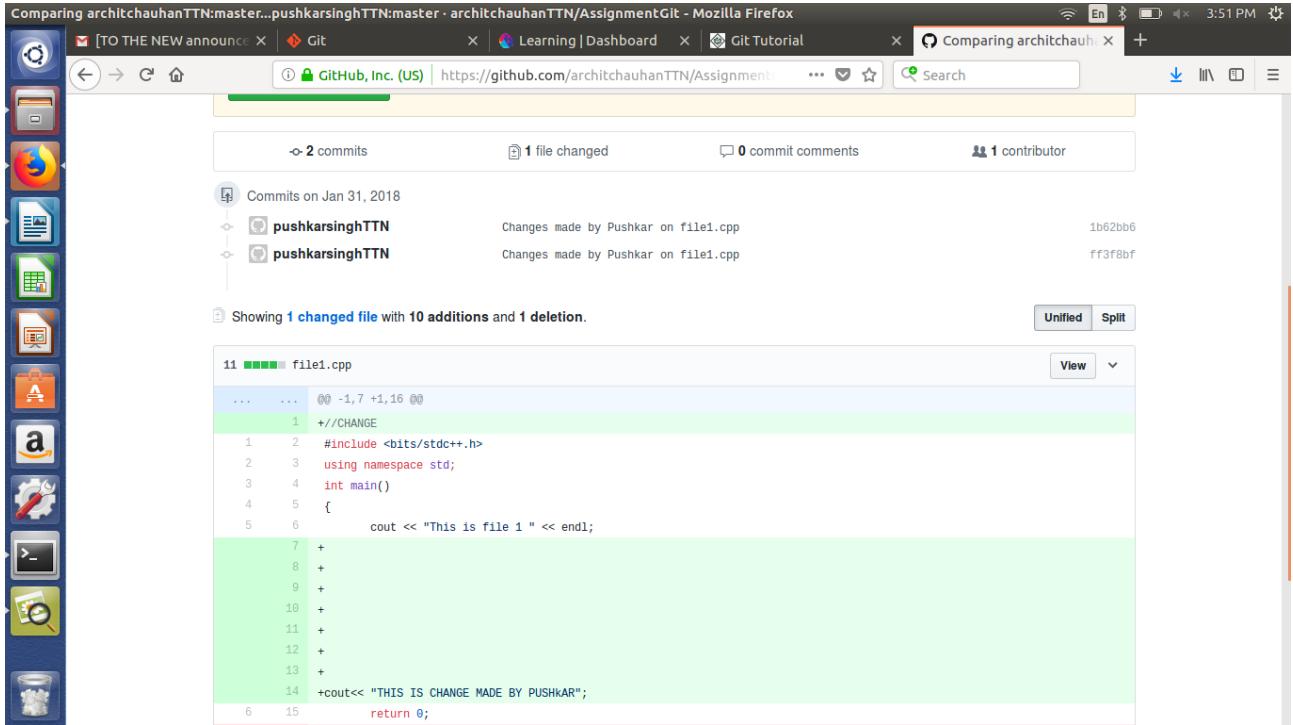
## Q10. Add changes to one of the copies and pull the changes in the other.

```
pushkar@Pushkar-Singh: ~/Desktop/AssignmentGit
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ ls
file1.cpp file2.cpp
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ vi file1.cpp
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ git status
On branch master
Your branch is up-to-date with 'origin/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:   file1.cpp

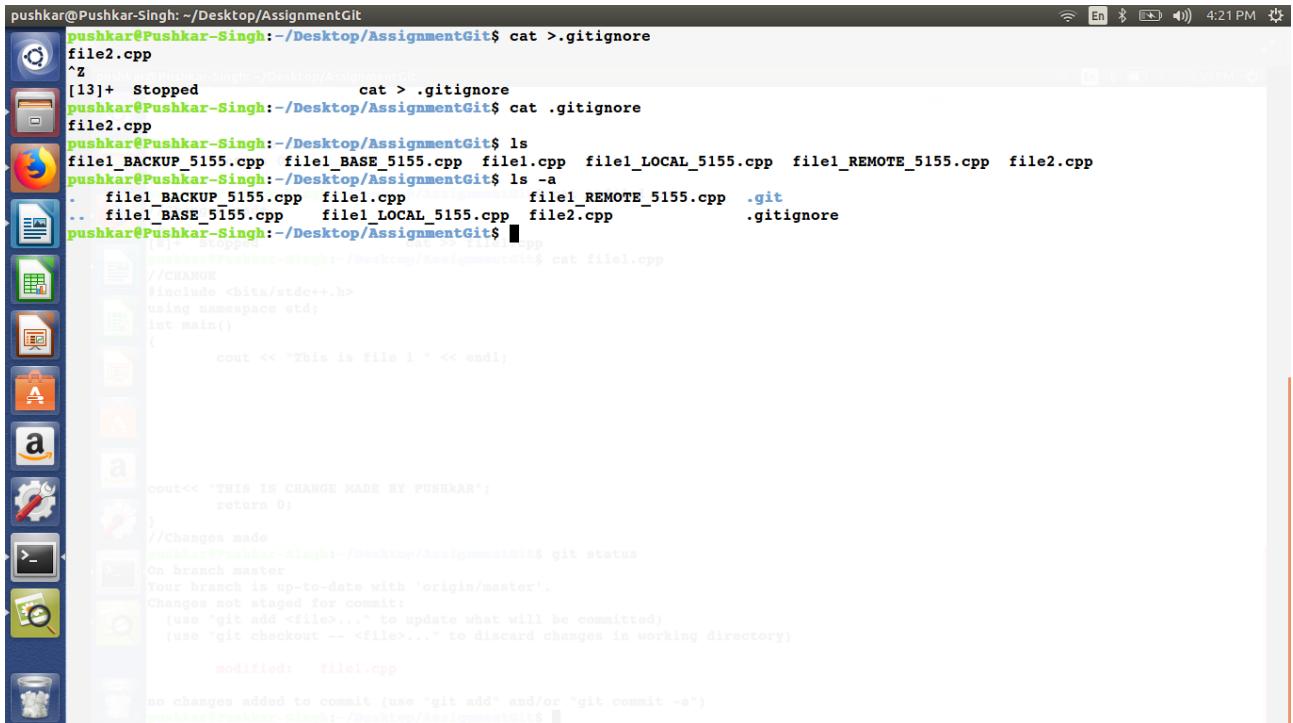
no changes added to commit (use "git add" and/or "git commit -a")
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ git add file1.cpp
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ git commit -m "Changes made by Pushkar on file1.cpp"
[master ff3f8bf] Changes made by Pushkar on file1.cpp
 1 file changed, 1 insertion(+), 1 deletion(-)
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ git push origin master
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 330 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To git@github.com:pushkarsinghTTN/AssignmentGit.git
  1b62bb6..ff3f8bf master -> master
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ ■
    14. Diverge them with commits
    15. Edit the same file at the same line on both branches and commit
    16. Try merging and resolve merge conflicts
    17. Stash the changes and pop them
    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/* //'
        }
        if [ "$color_prompt" = yes ]; then
        PS1="\u@\h \033[0m]:\033[01;34m]\W\033[01;31m]\$(parse_git_branch)\033[00m\$ "
        else
PS1="\u@\h \033[0m]:\033[01;34m]\W\033[01;31m]\$(parse_git_branch)\033[00m\$ "
```

## Q11. Check differences between a file and its staged version



The screenshot shows a Mozilla Firefox browser window with several tabs open. The active tab is 'Comparing architchauhanTTN:master...pushkarsinghTTN:master · architchauhanTTN/AssignmentGit - Mozilla Firefox'. The page displays a GitHub commit history for the 'AssignmentGit' repository. It shows two commits from 'pushkarsinghTTN' on Jan 31, 2018. The first commit (1b62bb6) changes 'file1.cpp', and the second commit (ff3f8bf) also changes 'file1.cpp'. Below the commit history, a diff view for 'file1.cpp' is shown, comparing the staged version (green background) with the working tree version (white background). The diff highlights 10 additions and 1 deletion.

## Q12. Ignore a few files to be checked in.



The screenshot shows a terminal window on a Linux desktop. The user is in the directory '/Desktop/AssignmentGit'. They run 'cat > .gitignore' to create a new ignore file. They then add 'file2.cpp' to the ignore file. Next, they run 'ls' to list files and 'ls -a' to show all files, including '.gitignore'. Finally, they run 'git status' to check the repository status, which shows 'On branch master' and 'Changes not staged for commit'. The terminal window has a standard Linux desktop interface with icons on the left.

## Q13. Create a new branch.



The screenshot shows a terminal window where the user runs 'git checkout -b branch1'. The output indicates that the 'branch1' branch has been switched to and contains the file 'file1.cpp'. The user then runs 'ls' to list files and 'git branch' to see the current branches. The output shows 'branch1' as the active branch and 'master' as the other branch. The terminal window has a standard Linux desktop interface with icons on the left.

#### Q14. Diverge them with commits

```
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ cat >>file2.cpp
//CHANGES MADE TO FILE2 BY PUSHKAR
^Z
[9]+  Stopped                  cat >> file2.cpp
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ git add *
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ git commit -m "this is for new branch"
[branch1 7ec3bc9] this is for new branch
 2 files changed, 2 insertions(+), 1 deletion(-) both branches and commit
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ git push origin branch1
Counting objects: 4, done.  in the changes and pop them
Delta compression using up to 4 threads. your .bashrc file : color_prompt='yes'
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 396 bytes | 0 bytes/s, done.
Total 4 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To git@github.com:pushkarsinghTTN/AssignmentGit.git
 * [new branch]      branch1 -> branch1
```

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



```
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit
master
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ cat >>file1.cpp
//This is change made in branch1
^Z
[10]+  Stopped                  cat >> file1.cpp
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ git add *
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ git commit -m "change made through branch1"
[branch1 450d05a] change made through branch1
 1 file changed, 1 insertion(+)
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ git push origin branch1
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 343 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To git@github.com:pushkarsinghTTN/AssignmentGit.git
 7ec3bc9..450d05a branch1 -> branch1
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ cat >>file1.cpp
//This is change made in master
^Z
[11]+  Stopped                  cat >> file1.cpp
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ git add *
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ git commit -m "chnage made through master"
[master d3f00ed] chnage made through master
 1 file changed, 1 insertion(+)
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ git push origin master
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 339 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To git@github.com:pushkarsinghTTN/AssignmentGit.git
  ff3f8bf..d3f00ed master -> master
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$
```

## Q16. Try merging and resolve merge conflicts.

pushkar@Pushkar-Singh: ~/Desktop/AssignmentGit\$ git merge branch1 master  
Auto-merging file1.cpp  
CONFLICT (content): Merge conflict in file1.cpp  
Automatic merge failed; fix conflicts and then commit the result.  
pushkar@Pushkar-Singh: ~/Desktop/AssignmentGit\$

1 Answer

merge is used to bring two (or more) branches together.

124 a little example:

```
# on branch A:  
# create new branch B  
$ git checkout -b B  
$ hack hack  
$ git commit -am "commit on branch B"  
  
# create new branch C from A  
$ git checkout -b C A  
$ Hack Hack  
$ git commit -am "commit on branch C"  
  
so now there are three separate branches (namely A B and C) with different heads
```

to get the changes from B and C back to A, checkout A (already done in this example) and then use the merge command:

```
# create an octopus merge  
$ git merge B C
```

Related

- How to clone all remote branches in Git?
- How to resolve merge conflicts in Git?
- What is the difference between git pull and git fetch?
- How to undo git add before commit?
- Make an existing Git branch track a remote branch?
- How to undo the most recent commits in Git?
- How do I check out a remote Git branch?
- How do I delete a Git branch both locally and remotely?
- Best (and safest) way to merge a git branch into master
- How do I rename a local Git branch?

Join Stack Overflow today. Share knowledge and learn from others. Ask a Question

```
pushkar@Pushkar-Singh: ~/Desktop/AssignmentGit$ 0 upgraded, 2 newly installed, 0 to remove and 27 not upgraded.  
Need to get 6,199 kB of archives.  
After this operation, 30.0 MB of additional disk space will be used.  
Do you want to continue? [Y/n] y  
Get:1 http://in.archive.ubuntu.com/ubuntu xenial-updates/main amd64 vim-runtime all 2:7.4.1689-3ubuntul.2 [5,164 kB]  
Get:2 http://in.archive.ubuntu.com/ubuntu xenial-updates/main amd64 vim amd64 2:7.4.1689-3ubuntul.2 [1,036 kB]  
Fetched 6,199 kB in 7s (854 kB/s)  
Selecting previously unselected package vim-runtime.  
(Reading database ... 177984 files and directories currently installed.)  
Preparing to unpack .../vim-runtime_2%3a7.4.1689-3ubuntul.2_all.deb ...  
Adding 'diversion of /usr/share/vim/vim74/doc/help.txt to /usr/share/vim/vim74/doc/help.txt.vim-tiny by vim-runtime'  
Adding 'division of /usr/share/vim/vim74/doc/tags to /usr/share/vim/vim74/doc/tags.vim-tiny by vim-runtime'  
Unpacking vim-runtime (2:7.4.1689-3ubuntul.2) ...  
Selecting previously unselected package vim.  
Preparing to unpack .../vim_2%3a7.4.1689-3ubuntul.2_amd64.deb ...  
Unpacking vim (2:7.4.1689-3ubuntul.2) ...  
Processing triggers for man-db (2.7.5-1) ...  
Setting up vim-runtime (2:7.4.1689-3ubuntul.2) ...  
Setting up vim (2:7.4.1689-3ubuntul.2) ...  
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/vim (vim) in auto mode  
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/vimdiff (vimdiff) in auto mode  
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/rvim (rvim) in auto mode  
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/rview (rview) in auto mode  
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/vi (vi) in auto mode  
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/view (view) in auto mode  
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/ex (ex) in auto mode  
pushkar@Pushkar-Singh: ~/Desktop/AssignmentGit$ vim file1.cpp  
pushkar@Pushkar-Singh: ~/Desktop/AssignmentGit$ git commit  
U file1.cpp  
error: commit is not possible because you have unmerged files.  
hint: Fix them up in the work tree, and then use 'git add/rm <file>'  
hint: as appropriate to mark resolution and make a commit.  
fatal: Exiting because of an unresolved conflict.  
pushkar@Pushkar-Singh: ~/Desktop/AssignmentGit$ git add *  
pushkar@Pushkar-Singh: ~/Desktop/AssignmentGit$ git commit  
[branch1 333dba4] Merge branch 'master' into branch1 resolved.  
pushkar@Pushkar-Singh: ~/Desktop/AssignmentGit$ git push origin master  
Everything up-to-date  
pushkar@Pushkar-Singh: ~/Desktop/AssignmentGit$
```

## Q17. Stash the changes and pop them

```
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ cat >>file1.cpp
//Changes2
^Z
[15]+  Stopped                  cat >> file1.cpp
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ git status
On branch branch1
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:   file1.cpp

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

no changes added to commit (use "git add" and/or "git commit -a")
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ git stash
Saved working directory and index state WIP on branch1: 333dba4 Merge branch 'master' into branch1
HEAD is now at 333dba4 Merge branch 'master' into branch1
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ git status
On branch branch1
Changes to be committed:
  (use "git add <file>..." to update what will be committed)
    (use "git checkout -- <file>..." to discard changes in the other
     Untracked files:
       10. Add changes to one of the copies and pull the changes in the other.
         (use "git add <file>..." to include in what will be committed)
           12. Ignore a few files to be checked in
             .gitignore
               13. Create a new branch
                 14. Remove from all commits
nothing added to commit but untracked files present (use "git add" to track)
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ git stash list
stash@{0}: WIP on branch1: 333dba4 Merge branch 'master' into branch1 resolved.
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ git stash pop -yes
On branch branch1
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:   file1.cpp
```

### 17. Stash the changes and pop them

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

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (557d1a6578656ela72a7b6078edc448b3559caf7)
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ █
```

```

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

```

The screenshot shows a Linux desktop environment with a terminal window open in the top panel. The terminal history includes:

```

pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ vim ~/.bashrc
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ vim ~/.bashrc
pushkar@Pushkar-Singh:~/Desktop/AssignmentGit$ source ~/.bashrc
pushkar@Pushkar-Singh:AssignmentGit (branch1)$

```

The terminal prompt shows the user is in a Git repository named 'AssignmentGit' with a branch named 'branch1'. Below the terminal, a sidebar displays various application icons, and at the bottom, there are navigation links for 'Attendance' and 'Exercises'.