Exercise - Introduction to Version Control

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

   Ans.

```
ttn@rahul-jain:~/Desktop/learning/learnings$ git push --hel
ttn@rahul-jain:~/Desktop/learning/learnings$ git --version
git version 2.17.1
```

2. Initialize a Git Repository

   Ans.

```
ttn@rahul-jain:~/Desktop$ mkdir version-control
ttn@rahul-jain:~/Desktop$ cd version-control/
ttn@rahul-jain:~/Desktop/version-control$ git init
Initialized empty Git repository in /home/ttn/Desktop/version
-control/.git/
```

3. Add files to the repository

   Ans.

```
ttn@rahul-jain:~/Desktop/version-control$ touch rahul.txt
ttn@rahul-jain:~/Desktop/version-control$ git add rahul.txt
ttn@rahul-jain:~/Desktop/version-control$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   rahul.txt
```

4. Unstage 1 file

   Ans.

```
ttn@rahul-jain:~/Desktop/version-control$ git rm --cached rahul.txt
rm 'rahul.txt'
ttn@rahul-jain:~/Desktop/version-control$ git status
On branch master

No commits yet

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

        rahul.txt

nothing added to commit but untracked files present (use "git add" to tra
ck)
```

5. Commit the file

   Ans.

```
ttn@rahul-jain:~/Desktop/version-control$ git add rahul.txt
ttn@rahul-jain:~/Desktop/version-control$ git commit -m "initial commit"
[master (root-commit) 227331f] initial commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 rahul.txt
```

6. Add a remote

   Ans.

```
ttn@rahul-jain:~/Desktop/version-control$ git remote add origin git@github.com
:rahulj236/Test_repo.git
ttn@rahul-jain:~/Desktop/version-control$ git status
On branch master
nothing to commit, working tree clean
```

7. Undo changes to a particular file.

Ans.

```
ttn@rahul-jain:~/Desktop/version-control$ vi rahul.txt
ttn@rahul-jain:~/Desktop/version-control$ git add .
ttn@rahul-jain:~/Desktop/version-control$ git commit
Aborting commit due to empty commit message.
ttn@rahul-jain:~/Desktop/version-control$ git commit -m "changes in file done"

[master 1768eeb] changes in file done
 1 file changed, 1 insertion(+)
ttn@rahul-jain:~/Desktop/version-control$ git revet HEAD~0
git: 'revet' is not a git command. See 'git --help'.

The most similar command is
        revert
ttn@rahul-jain:~/Desktop/version-control$ git revert HEAD~0
[master db131c9] Revert "changes in file done"
 1 file changed, 1 deletion(-)
ttn@rahul-jain:~/Desktop/version-control$ git status
On branch master
nothing to commit, working tree clean
```

8. Push changes to Github

   Ans.

```
ttn@rahul-jain:~/Desktop/version-control$ git push origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 240 bytes | 240.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:rahulj236/repo_test.git
 * [new branch]      master -> master
```

9. Clone the repository

   Ans.

```
ttn@rahul-jain:~/Desktop$ git clone git@github.com:rahulj236/repo_test.git
Cloning into 'repo_test'...
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@rahul-jain:~/Desktop$ ls
learning  repo_test  version-control
```

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

Ans.

```
ttn@rahul-jain:~/Desktop$ cd repo_test
ttn@rahul-jain:~/Desktop/repo_test$ ls
rahul.txt
ttn@rahul-jain:~/Desktop/repo_test$ vi rahul.txt
ttn@rahul-jain:~/Desktop/repo_test$ git add .
ttn@rahul-jain:~/Desktop/repo_test$ git commit -m "changes done at repo_test"
[master 65900ac] changes done at repo_test
 1 file changed, 1 insertion(+), 1 deletion(-)
ttn@rahul-jain:~/Desktop/repo_test$ git push origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 286 bytes | 286.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:rahulj236/repo_test.git
   7a09011..65900ac  master -> master
ttn@rahul-jain:~/Desktop/repo_test$ cd ~
ttn@rahul-jain:~$ cd Desktop
ttn@rahul-jain:~/Desktop$ cd version-control
ttn@rahul-jain:~/Desktop/version-control$ ls
rahul.txt
ttn@rahul-jain:~/Desktop/version-control$ git pull origin master
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From github.com:rahulj236/repo_test
 * branch            master     -> FETCH_HEAD
   7a09011..65900ac  master     -> origin/master
Updating 7a09011..65900ac
Fast-forward
 rahul.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

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

    Ans.

```
ttn@rahul-jain:~/Desktop/version-control$ vi rahul.txt
ttn@rahul-jain:~/Desktop/version-control$ git add rahul.txt
ttn@rahul-jain:~/Desktop/version-control$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   rahul.txt

ttn@rahul-jain:~/Desktop/version-control$ git diff --cached rahul.txt
diff --git a/rahul.txt b/rahul.txt
index 614e822..4e4d876 100644
--- a/rahul.txt
+++ b/rahul.txt
@@ -1 +1,2 @@
 this is another change doing this far
+hello rahul
```

12. Ignore a few files to be checked in

    Ans.

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

13. Create a new branch.

    Ans.

```
ttn@rahul-jain:~/Desktop/version-control$ git branch newbranch
ttn@rahul-jain:~/Desktop/version-control$ git branch
* master
  newbranch
```

14. Diverge them with commits

    Ans.

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

        new file:   ignoreFile1.txt
        modified:   rahul.txt

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:    ignoreFile1.txt

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

        .gitignore

ttn@rahul-jain:~/Desktop/version-control$ git commit -m "commit in master"
[master 7d10dc4] commit in master
 2 files changed, 1 insertion(+)
 create mode 100644 ignoreFile1.txt
```

```
ttn@rahul-jain:~/Desktop/version-control$ git commit -m "commit in master"
[master 7d10dc4] commit in master
 2 files changed, 1 insertion(+)
 create mode 100644 ignoreFile1.txt
ttn@rahul-jain:~/Desktop/version-control$ git checkout newbranch
Switched to branch 'newbranch'
ttn@rahul-jain:~/Desktop/version-control$ ls
file1.txt  file2.txt  rahul.txt
ttn@rahul-jain:~/Desktop/version-control$ vi rahul.txt
ttn@rahul-jain:~/Desktop/version-control$ git add rahul.txt
ttn@rahul-jain:~/Desktop/version-control$ git commit -m "commit in new branch"
[newbranch f03bb4c] commit in new branch
 1 file changed, 1 insertion(+)
ttn@rahul-jain:~/Desktop/version-control$ git s
shortlog      show          show-branch   stage         stash         status        submodule     subtr
ttn@rahul-jain:~/Desktop/version-control$ git status
On branch newbranch
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .gitignore

nothing added to commit but untracked files present (use "git add" to track)
```

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

    Ans.

```
ttn@rahul-jain:~/Desktop/version-control$ git checkout
ttn@rahul-jain:~/Desktop/version-control$ git branch
  master
* newbranch
ttn@rahul-jain:~/Desktop/version-control$ vi rahul.txt
ttn@rahul-jain:~/Desktop/version-control$ ls
file1.txt  file2.txt  rahul.txt
ttn@rahul-jain:~/Desktop/version-control$ rahul.txt
rahul.txt: command not found
ttn@rahul-jain:~/Desktop/version-control$ git add rahul.txt
ttn@rahul-jain:~/Desktop/version-control$ git commit -m "changes made on line
2"
[newbranch 9038683] changes made on line 2
 1 file changed, 1 insertion(+), 1 deletion(-)
ttn@rahul-jain:~/Desktop/version-control$ git checkout master
Switched to branch 'master'
ttn@rahul-jain:~/Desktop/version-control$ vi rahul.txt
ttn@rahul-jain:~/Desktop/version-control$ git add rahul.txt
ttn@rahul-jain:~/Desktop/version-control$ git commit -m "changes made in line
2 in master branch"
[master 2dbd1d9] changes made in line 2 in master branch
 1 file changed, 1 insertion(+), 1 deletion(-)
```

16. Try merging and resolve merge conflicts

    Ans.

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

Before:

```
this is another change doing this far
<<<<<<< HEAD
change in new branch doing again
=======
hello rahul changesndoing
>>>>>>> master
~
```

After :

```
this is another change doing this far
change in new branch doing again
hello rahul changesndoing
```

```
ttn@rahul-jain:~/Desktop/version-control$ git add .
ttn@rahul-jain:~/Desktop/version-control$ git commit -m "resolved merge conflict"
[newbranch 1599db3] resolved merge conflict
ttn@rahul-jain:~/Desktop/version-control$ git status
On branch newbranch
nothing to commit, working tree clean
```

```
ttn@rahul-jain:~/Desktop/version-control$ git merge master
Already up to date.
```

17. Stash the changes and pop them.

    Ans.

```
ttn@rahul-jain:~/Desktop/version-control$ git branch
  master
* newbranch
ttn@rahul-jain:~/Desktop/version-control$ vi rahul.txt
ttn@rahul-jain:~/Desktop/version-control$ git checkout master
error: Your local changes to the following files would be overwritten by checkout:
        rahul.txt
Please commit your changes or stash them before you switch branches.
Aborting
ttn@rahul-jain:~/Desktop/version-control$ git stash -p
diff --git a/rahul.txt b/rahul.txt
index 489095a..ef00763 100644
--- a/rahul.txt
+++ b/rahul.txt
@@ -1,3 +1,4 @@
 this is another change doing this far
 change in new branch doing again
-hello rahul changesndoing
+hello rahul changesndoinga
+stash command doing
Stash this hunk [y,n,q,a,d,e,?]? y

Saved working directory and index state WIP on newbranch: 1599db3 resolved merge conflict
```

Pop:

```
ttn@rahul-jain:~/Desktop/version-control$ git checkout master
Switched to branch 'master'
ttn@rahul-jain:~/Desktop/version-control$ git stash pop
Auto-merging rahul.txt
CONFLICT (content): Merge conflict in rahul.txt
```

Before:

```
this is another change doing this far
<<<<<<< Updated upstream
hello rahul changesndoing
=======
change in new branch doing again
hello rahul changesndoinga
stash command doing
>>>>>>> Stashed changes
~
```

After:

```
this is another change doing this far
hello rahul changesndoing
change in new branch doing again
hello rahul changesndoinga
stash command doing
```

Pop again:

```
ttn@rahul-jain:~/Desktop/version-control$ git stash pop
rahul.txt: needs merge
unable to refresh index
```

Stash show:

```
ttn@rahul-jain:~/Desktop/version-control$ git stash show
 rahul.txt | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
```

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.

In bashrc file:

```
# 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)\[\033[00m\]\$ '
else
PS1='\u@\h:\W $(parse_git_branch)\$ '
fi
unset color_prompt force_color_prompt
```

After:

```
ttn@rahul-jain:version-control (master)$ vi ~/.bashrc
ttn@rahul-jain:version-control (master)$ source ~/.bashrc
ttn@rahul-jain:version-control (master)$
```